

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Aerotech AS32020C* at our website: ***Click HERE***

THE UNIDEX® 500 MOTION CONTROLLER AND WINDOWS SOFTWARE

OPERATION & TECHNICAL MANUAL
P/N: EDU150 (V1.3a)



AEROTECH, Inc. • 101 Zeta Drive • Pittsburgh, PA. 15238-2897 • USA
Phone (412) 963-7470 • Fax (412) 963-7459
Product Service: (412) 967-6440; (412) 967-6870 (Fax)

www.aerotechinc.com

If you should have any questions about the UNIDEX 500 or comments regarding the documentation, please refer to Aerotech online at:
http://www.aerotechinc.com.

For your convenience, a product registration form is available at our web site.

Our web site is continually updated with new product information, free downloadable software and special pricing on selected products.

The UNIDEX 500 PC-based motion controller, Toolkit, and MMI software packages are products of Aerotech, Inc.
Borland C is a product of Borland International, Inc.
IBM PC/AT bus is a registered trademark of International Business Machines, Inc.
Inductosyn is a registered trademark of Farrand Industries, Inc.
iSBX is a registered trademark of Intel Corporation.
MS-DOS, QuickBASIC and Windows are products of Microsoft Corporation.

The UNIDEX 500 Motion Controller and Windows Software
Operations and Technical Manual Revision History:

Rev 1.0	April 18, 1995
Rev 1.1	June 26, 1995
Rev 1.2	June 14, 1996
Rev 1.3	July 5, 2000
Rev 1.3a	November 17, 2000

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION 1-1
1.1. Overview of the UNIDEX 500 System 1-1
1.2. The UNIDEX 500 Family of Controllers 1-1
1.2.1. The UNIDEX 500 Base Model (U500) 1-2
1.2.2. The UNIDEX 500 Plus Model (U500PLUS) 1-3
1.2.3. The UNIDEX 500 Ultra Model (U500ULTRA) 1-3
1.3. Ordering Information 1-3
1.4. Options and Accessories 1-5
1.5. Safety Procedures and Warnings 1-6

CHAPTER 2: GETTING STARTED 2-1
2.1. Introduction 2-1
2.2. Unpacking the UNIDEX 500 System 2-1
2.3. Inspection of the UNIDEX 500 Control Board 2-2

CHAPTER 3: SOFTWARE INSTALLATION AND FUNDAMENTALS 3-1
3.1. Introduction 3-1
3.2. Minimum Hardware Requirements and Recommended System Configurations 3-1
3.3. Standard Installation Using Windows NT 4.0 or 95 3-2
3.4. Installing the UNIDEX 500 PC Board 3-3
3.5. Software Startup 3-4
3.5.1. U500 ISA Base Address Jumpers (JP4, JP5, JP6, JP7, JP8, and JP9) 3-4
3.5.2. Configuring the Software to Match the Hardware 3-5
3.5.3. Configuration of the Software for the ISA Board 3-6
3.5.4. Configuring the Software for the PCI Board 3-6
3.5.5. Initializing the U500 Board 3-7
3.6. Software Configuration Considerations 3-9
3.6.1. Configuring Key Parameters 3-9
3.7. Special Startup Considerations 3-11
3.7.1. Feedback Verification 3-15
3.7.2. Limit Verification 3-15
3.7.3. Preliminary Servo Loop Setup 3-15

CHAPTER 4: THE SOFTWARE INTERFACE 4-1
4.1. Introduction 4-1
4.2. The File Menu 4-2
4.2.1. The Project... Option of the File Menu 4-3
4.2.2. The Parameters... Option of the File Menu 4-3
4.2.3. The Firmware... Option of the File Menu 4-3
4.2.4. The Load Program... Option of the File Menu 4-3
4.2.5. The Reset Option of the File Menu 4-4
4.2.6. The Exit Option of the File Menu 4-4
4.3. The Edit Menu 4-4
4.3.1. The Project... Option of the Edit Menu 4-4
4.3.2. The Parameters... Option of the Edit Menu 4-5
4.3.2.1. Edit Parameters: The File Submenu 4-6

4.3.2.2. Edit Parameters: The Number, Value, and Axis Fields 4-6
4.3.3. System Options 4-7
4.3.4. The Program Option of the Edit Menu 4-14
4.4. The Tools Menu 4-15
4.4.1. The Diagnostics Option of the Tools Menu 4-15
4.4.2. The Demo Option of the Tools Menu 4-19
4.4.3. The Axis Tuning... Option of the Tools Menu 4-20
4.4.3.1. Axis Scope: The File Submenu 4-20
4.4.3.2. Axis Scope: The Plot Submenu 4-21
4.4.3.3. Axis Scope: The Trigger Submenu 4-22
4.4.3.4. Axis Scope: The Collect Submenu 4-23
4.4.3.5. Axis Scope: The Display Submenu 4-24
4.4.3.6. Axis Scope: The Axis Submenu 4-24
4.4.3.7. Axis Scope: The Units Submenu 4-25
4.4.3.8. Axis Scope: The Tools Submenu 4-25
4.4.4. The Joystick Digitizing Option of the Tools Menu 4-31
4.4.5. The "Diagnostics 2" Option of the Tools Menu 4-33
4.4.6. The Variable Watch Option of the Tools Menu 4-34
4.5. The Windows Menu 4-35
4.5.1. The Functions Option of the Windows Menu 4-35
4.5.1.1. The F2 Abort Button 4-36
4.5.1.2. The F3 Pause/Resume Button 4-36
4.5.1.3. The F4 Load Button and the Program Display Window 4-36
4.5.1.4. The F5 Jog Button 4-38
4.5.1.5. Password Protection 4-39
4.5.1.6. The F7 Diag Button 4-42
4.5.1.7. The F8 Fault Ack Button 4-42
4.5.1.8. The F9 Reset Button 4-42
4.5.2. The Status Option of the Windows Menu 4-43
4.5.3. The Jog Option of the Windows Menu 4-45
4.5.4. The Gain Adjust Option of the Windows Menu 4-45
4.5.5. The MDI Option of the Windows Menu 4-45
4.5.6. The MFO Option of the Windows Menu 4-46
4.5.7. The Axis Display Option of the Windows Menu 4-46
4.6. The Functions Menu 4-48
4.7. The Help Menu 4-49

CHAPTER 5: PARAMETERS 5-1
5.1. Introduction 5-1
5.2. The Edit Parameters Window 5-1
5.2.1. The Menu Bar of the Edit Parameters Window 5-2
5.2.2. The Parameter Tabs 5-2
5.2.3. The Scrollable Parameter Display Window 5-2
5.2.4. The Parameter Number Field 5-2
5.2.5. The Parameter Value Field 5-3
5.2.6. Axis Number Radio Buttons 5-4
5.3. The Position Tracking Tab 5-5
5.3.1. Keep Axis Position after Reset (Y/N) 5-6
5.3.2. Axis *n* Rollover Machine Steps for Axes 1-4 5-7

5.3.3. PSO Mailbox Dual_port RAM Base Address 5-8
5.3.4. PSO-HOST and PC Interface Address (U500 ISA - 16 bit MMI software only) 5-9
5.3.5. Plane *n* Metric System (Y/N) 5-10
5.3.6. Metric Mode Number of Decimal Digits (1-8) 5-11
5.3.7. English Mode Number of Decimal Digits (1-8) 5-12
5.4. The Advanced Motion Tab 5-13
5.4.1. Overview of Planes 5-14
5.4.2. Number of Contour Planes (1, 2, 4) 5-17
5.4.3. Axis *n* Map to Plane {1-4} as {X, Y, Z, U} 5-19
5.4.4. Axis *n* Gantry {Y/None}, Slave {1, 2, 3, 4} 5-20
5.4.5. Plane *n* Indexing Segment Time (1-20 ms) 5-23
5.4.6. Linear Type Ac/De (Y/N) 5-24
5.4.7. Clamp Feedrate (Program Steps/ms) 5-25
5.4.8. Corner Rounding Non-Ramp Time (1-32,000 ms) 5-26
5.4.9. Contouring Mode 5-28
5.4.10. A/D Channel *n* Joystick Deadband 5-29
5.4.11. A/D Channel *n* Joystick Center Position 5-30
5.4.12. Safe Zone Output Bit (0-8) 5-30
5.4.13. Option Board Setup Code 5-31
5.4.14. User Interrupt Setup Code 5-32
5.4.15. Abort On Input High (0, 1-16) 5-32
5.5. The Basic Motion Tab 5-33
5.5.1. MFO, 0 No MFO-pot, 1-255 Pot Offset 5-34
5.5.2. Plane *n* Contour Ramping Time (ms) 5-35
5.5.3. Contour Feedrate (Program Steps/ms) for Plane *n* 5-36
5.5.4. X, Y, Z, and U Axes' Point-to-point Feedrates (Program Steps/ms) for Plane *n* 5-36
5.6. The Homing and Limits Tab 5-37
5.6.1. The Home Cycle 5-38
5.6.2. Home Direction is CCW (Y/N) 5-40
5.6.3. Home Switch Normally Open (Y/N) 5-41
5.6.4. Power-on Home Feedrate (Machine Steps/ms) 5-42
5.6.5. Normal Home Feedrate (Machine Steps/ms) 5-43
5.6.6. Home Offset (Machine Steps) 5-43
5.6.7. Home Switch to Marker (Machine Steps) – U500 ISA Only 5-44
5.6.8. Home/Limit Switch Debounce (ms) 5-45
5.6.9. Limit Switch Normally Open (Y/N) 5-45
5.6.10. Switch to Mechanical Stop (Machine Steps) 5-46
5.6.11. CCW Software Limit (Machine Steps) 5-46
5.6.12. CW Software Limit (Machine Steps) 5-46
5.6.13. Enable Vel/Accel Feedforward During Home (Y/N) 5-47
5.6.14. Use Home Limit During Home Cycle (Y/N) 5-48
5.6.15. Safe Zone Limits (Machine Steps) 5-48
5.6.16. Home/Limit Switch Debounce (Steps) 5-49
5.6.17. Home Marker Search Speed (Machine Steps/ms) – U500 ISA Only 5-49
5.7. The Traps Tab 5-50
5.7.1. Max Ac/De (Machine Steps/ms/ms) 5-51
5.7.2. Top Feedrate (Machine Steps/ms) 5-52
5.7.3. Max Velocity Error (0-8,388,607) 5-53

5.7.4. Max Position Error (0-8,388,607) 5-54
5.7.5. Max Integral Error (0-8,833,607) 5-55
5.7.6. RMS Current Trap (0-100%) 5-56
5.7.7. RMS Current Sample Time (1-16,383 ms) 5-58
5.7.8. Clamp Current Output (0-100%) 5-59
5.7.9. Output Bit for AUX OUTPUT 5-60
5.7.10. Drive Fault Signal Active Low 5-61
5.7.11. AUX OUTPUT Active High 5-61
5.8. The Motor and Feedback Configuration Tab 5-62
5.8.1. Introduction to Motors and Feedback Configurations 5-63
5.8.2. Motor Feedback and Configuration Parameters 5-71
5.8.3. Primary/Position Feedback Channel 5-72
5.8.3.1. Encoder Gantry Mode 5-73
5.8.4. Secondary/Velocity Feedback Channel 5-75
5.8.5. Primary Feedback Setup Code 5-76
5.8.6. Secondary Feedback Setup Code 5-76
5.8.7. Drive (Motor) Type (0-DC Brush, 1-AC Brushless, 2/3-Stepper) 5-77
5.8.8. AC Brushless Motor Commutation Factor (Cycles/Rev) 5-78
5.8.9. Encoder Feedback (Steps/Rev/(*4)) 5-79
5.8.10. AC Brushless Motor Phase Offset (Degrees) 5-79
5.8.11. Stepper High Current (0-100%) 5-80
5.8.12. Stepper Low Current (0-100%) 5-80
5.8.13. Stepper Microstepping Resolution (Steps/Rev) 5-80
5.8.14. Stepper Encoder Verification (Y/N) 5-81
5.8.15. Stepper Encoder Speed (Microsteps/ms) 5-81
5.8.16. AC Brushless Motor Phase Advance Base Speed (Steps/ms) 5-82
5.8.17. AC Brushless Motor Base Speed Advance (Degrees) 5-83
5.8.18. AC Brushless Motor Phase Speed (steps/ms) 5-83
5.8.19. AC Brushless Motor Phase Speed Advance (Degrees) 5-83
5.8.20. Current Command Offset Parameters (mV) 5-83
5.8.21. Encoder Multiplication 5-83
5.9. The Servo Loops Tab 5-84
5.9.1. Notch Filter (Y/N) 5-85
5.9.2. Kpos (0-8,388,607)(Position Loop Gain) 5-85
5.9.3. Ki (0-8,388,607)(Velocity Loop Integrator) 5-85
5.9.4. Kp (0-8,388,607)(Velocity Loop Proportional Gain) 5-86
5.9.5. Vff (0-8,388,607)(Velocity Feed Forward) 5-86
5.9.6. Aff (0-8,388,607)(Acceleration Feed Forward) 5-86
5.9.7. Notch Filter Coefficient N0, N1, N2, D1, and D2 5-87
5.9.7.1. The Notch Filter 5-87
5.9.7.2. Notch Filter Example 5-89
5.9.7.3. The Second Order Low Pass Filter 5-91
5.9.8. Servo Loop Update Rate (1-100) x 0.25 ms 5-92
5.9.9. Servo Loop Type 5-92
5.9.10. Filter Time Constant 5-93
5.10. The Other Tab 5-94
5.10.1. Metric (x00) and English (x01) Conversion Factors 5-95
5.10.2. Positive (+) Move is Clockwise (Y/N) 5-98
5.10.3. Positive (+) Jog is Same Direction as + Move (Y/N) 5-99

5.10.4. Pause Enable in Freerun (Y/N) 5-99
5.10.5. MFO Enable in Freerun (Y/N) 5-100
5.10.6. Calibration Enable (Y/N) 5-100
5.10.7. In Position Deadband (Machine Steps) 5-101
5.10.8. Backlash (Machine Steps) 5-102
5.10.9. Joystick: High Speed (Machine Steps/sec) 5-103
5.10.10. Joystick: Low Speed (Machine Steps/sec) 5-103
5.10.11. Absolute Mode Scale (Machine Steps) 5-104
5.10.12. Orthogonality Correction Table Enabled (Y/N) 5-104
5.10.13. 2-D Error Mapping Enabled (Y/N) 5-105
5.10.14. Reverse Joystick Direction 5-105
5.11. The Faults Tab 5-106
5.11.1. Introduction to Fault Masks 5-106
5.11.2. Error Mask Fault 5-108
5.11.3. Disable Axis 5-108
5.11.4. Interrupt PC 5-108
5.11.5. AUX OUTPUT 5-109
5.11.6. Halt Queue 5-109
5.11.7. Abort Motion 5-109
5.11.8. Enable Brake 5-110
5.12. The General Tab 5-111
5.13. The Axis Tab 5-112

CHAPTER 6: TUNING SERVO LOOPS 6-1
6.1. Introduction 6-1
6.2. The Axis Scope Toolbars 6-2
6.2.1. Kp Proportional Gain 6-3
6.2.2. Ki Integral Gain 6-3
6.2.3. Kpos Position Gain 6-3
6.2.4. Vff Velocity Feedforward Gain 6-3
6.2.5. Aff Acceleration Feedforward Gain 6-3
6.3. Autotuning 6-4
6.3.1. Setting up an Excitation 6-4
6.3.2. Specifying Desired Performance 6-4
6.3.3. Bandwidth and Damping 6-4
6.3.4. Procedure 6-5
6.3.5. Dual Loop Systems 6-8
6.3.6. Guidelines and Limitations 6-8
6.4. Manual Tuning Procedure for Servo Loops 6-9
6.5. Tuning Tips 6-19
6.6. Tachometer Feedback Basics 6-20
6.6.1. In-Position Integrator 6-20
6.6.2. Velocity Feed Forward 6-20
6.6.3. Servo Parameter Setup for Tachometer Feedback 6-20
6.6.4. The Axis Scope Toolbars 6-21
6.6.4.1. Kpos Position Gain 6-21
6.6.4.2. Ki In-Position Integrator 6-21
6.6.4.3. Vff Velocity Feedforward Gain 6-21
6.6.4.4. Kp Proportional Gain 6-21
6.6.4.5. Aff Acceleration Feedforward Gain 6-21
6.7. Tuning Tachometer Loops 6-22

CHAPTER 7: PROGRAMMING COMMANDS 7-1
7.1. Introduction 7-1
7.2. Program Execution Levels 7-2
7.2.1. Board Level Execution 7-2
7.2.2. C Library Interface 7-2
7.2.3. C Library ASCII Interface 7-2
7.3. Mathematical Function Commands 7-3
7.3.1. Direct Variables (V0 through V255) 7-3
7.3.2. Indirect Variables (VV0 through VV255) 7-3
7.3.3. Functions 7-3
7.3.4. Operators and Evaluation Hierarchy 7-5
7.4. System Registers 7-6
7.4.1. Relative Position Registers 7-6
7.4.2. Absolute Position Registers 7-6
7.4.3. Real Time Feedback Position Registers 7-7
7.4.4. Real Time Commanded Position Registers 7-7
7.4.5. Understanding the Concept of Program Steps 7-8
7.4.6. A/D Channel Registers 7-8
7.5. System Inputs $INP and $IN0-$INF 7-10
7.6. iSBX I/O Interface (optional on U500 ISA only) 7-11
7.7. 4EN Expansion Board I/O Interface (optional on U500 ISA only) 7-14
7.8.1. ABORT 7-22
7.8.2. ACCELERATION 7-22
7.8.3. AC PL (ACcel PLane) 7-23
7.8.4. AFCO (Auto Focus) 7-23
7.8.5. AGAIN 7-27
7.8.6. Analog Vector Tracking Mode 7-28
7.8.7. AT (Autotune) 7-30
7.8.8. BOARD 7-31
7.8.9. BRAKE 7-32
7.8.10. CAL (Load Calibration File) 7-33
7.8.11. CALLDLL 7-34
7.8.12. CLOCKWISE and COUNTERCLOCKWISE CIRCULAR INTERPOLATION 7-37
7.8.13. CLRSCR (Clear Screen) 7-42
7.8.14. CM (Contouring Mode) 7-42
7.8.15. COM FREE 7-43
7.8.16. COM GET 7-43
7.8.17. COM INIT 7-45
7.8.18. COM RECEIVE 7-45
7.8.19. COM SEND 7-46
7.8.20. COM SET _RECEIVE_ 7-47
7.8.21. COM SET _STATUS_ 7-48
7.8.22. COM SET _TERMINATE_ 7-49
7.8.23. COM SET _TIMEOUT_ 7-50
7.8.24. CS (Command Scope) 7-51
7.8.25. Cutter Compensation Commands 7-52
7.8.26. CVI (Convert to Integer) 7-55
7.8.27. CYCLE 7-56
7.8.28. DAC (D/A Output) 7-57

7.8.29. DISABLE 7-58
7.8.30. DS (Display Servo Loop Data) 7-59
7.8.31. DWELL 7-60
7.8.32. DY (Dynamic Gain) 7-60
7.8.33. EC 7-61
7.8.34. ENABLE 7-63
7.8.35. ERROR 7-64
7.8.36. EXIT 7-65
7.8.37. FAULT ACKNOWLEDGE 7-66
7.8.38. FL (Filter Time Constant) 7-66
7.8.39. FREEDLL 7-67
7.8.40. FREERUN 7-68
7.8.41. GAIN 7-69
7.8.42. GEAR 7-70
7.8.43. GOTO 7-71
7.8.44. HALT 7-73
7.8.45. HOME 7-74
7.8.46. IF 7-75
7.8.47. IMMEDIATE 7-77
7.8.48. INDEX 7-78
7.8.49. INn (Read Inputs) 7-79
7.8.50. INTERRUPT 7-80
7.8.51. IO COMMAND 7-81
7.8.52. IOSET COMMAND 7-81
7.8.53. JOG 7-82
7.8.54. Label Marker (:) 7-82
7.8.55. LINEAR 7-83
7.8.56. LOADDLL 7-84
7.8.57. LOOP 7-85
7.8.58. LVDT 7-86
7.8.59. M0 (“M Zero”) 7-87
7.8.60. MCOMM (Motor Commutation) 7-88
7.8.61. MESSAGE 7-89
7.8.62. MR (Memory Read) 7-90
7.8.63. MSET (Motor Setup) 7-91
7.8.64. MW (Memory Write) 7-94
7.8.65. NEXT 7-95
7.8.66. OEn (Extended Output) 7-96
7.8.67. OUTPUT 7-97
7.8.68. PARALLEL 7-98
7.8.69. PRM (PARAMETER READ) 7-99
7.8.70. PAUSE 7-100
7.8.71. PLANE 7-101
7.8.72. PLC 7-102
7.8.73. POSITION COMMAND (IN POSITION) 7-105
7.8.74. PROGRAM 7-106
7.8.75. QUEUE 7-108
7.8.76. RAMP 7-110
7.8.77. REFERENCE 7-111
7.8.78. RETURN 7-111
7.8.79. ROTATE (Part Rotation) 7-112
7.8.80. ROUNDING 7-113

7.8.81. SCOPE 7-116
7.8.82. SCF (Overriding Scale Factor) 7-117
7.8.83. SKEY (Soft Keys) 7-118
7.8.84. SLAVE 7-119
7.8.85. SLEW 7-124
7.8.86. SOFTWARE 7-126
7.8.87. SPLINE 7-128
7.8.88. START 7-129
7.8.89. SUBROUTINE 7-130
7.8.90. SYNC 7-130
7.8.91. Target Tracking Commands (TE, TD, TP) 7-131
7.8.92. TRAJECTORY 7-132
7.8.93. TRIGGER 7-133
7.8.94. UMFO (Manual Feed Override) 7-133
7.8.95. VAR (Read/Write Variables) 7-134
7.8.96. VELOCITY 7-135
7.8.96.1. Correct Usage and Limitations of the Velocity Profiling Algorithm in Contour Mode 0 (CM0) 7-136
7.8.96.2. CM1 Contouring Mode 7-140
7.8.97. WAIT 7-144
7.8.98. WHILE/ENDWHILE 7-145

CHAPTER 8: U500 PCI PULSE OUTPUT 8-1

8.1. Hardware Configuration 8-1
8.2. Pulse Output Programming 8-2
8.2.1. Single PSO Mode 8-2
8.2.1.1. Array Download 8-5
8.2.1.2. Window Comparators 8-7
8.2.2. Dual Axis PSO Mode 8-10
8.3. U500PCI/PSO-PC Functionality 8-14

CHAPTER 9: SAMPLE PROGRAMS 9-1

9.1. Introduction 9-1
9.2. Incremental (Relative) Motion with Velocity Profiling 9-2
9.3. Absolute Motion with Velocity Profiling 9-5
9.4. CNC Demonstration Using Velocity Profiling, Linear, and Circular Interpolation 9-7
9.5. Corner Rounding 9-8
9.6. GEAR Demonstration of a Master Axis with Two Slave Axes 9-10
9.7. Interlocking Contour Planes 9-11
9.8. Splining 9-12
9.9. Programming Using Inputs 9-13
9.10. Part Rotation 9-14
9.11. Overriding Scale Factor 9-16
9.12. COM Functions 9-19

CHAPTER 10: PROGRAMMING TOOLS 10-1

10.1. Introduction 10-1
10.2. Summary of the UNIDEX 500 Quick Library Functions (32bit) 10-2
10.2.1. Overview 10-2

10.2.2. Device Driver Information 10-2
10.2.3. Registry Information 10-3
10.2.4. Quick Library Functions..... 10-4
10.2.5. Arguments and Return Values for the aerq_read_status Function..... 10-18
10.2.6. Programming Example..... 10-21
10.3. Summary of the UNIDEX 500 WAPI Functions (32 bit)..... 10-31
10.3.1. Overview..... 10-31
10.3.2. Device Driver Information 10-31
10.3.3. Registry Information 10-32
10.3.4. WAPI Library Functions..... 10-33
10.3.5. Arguments and Return Values for the WAPIAerReadStatus Function..... 10-40
10.3.6. Example file for U500 "WAPI" library "WIN50032.DLL" 10-43

CHAPTER 11: UTILITY PROGRAMS..... 11-1
11.1. Introduction 11-1
11.2. The AERVER.EXE Utility (32 bit software only)..... 11-1
11.3. The DSPDMP32.EXE Program (32 bit software only)..... 11-2

CHAPTER 12: TECHNICAL DETAILS..... 12-1
12.1. Technical Details Common between U500 PCI and ISA Cards..... 12-1
12.1.1. Encoder Signal Specifications..... 12-1
12.1.1.1. Differential Encoders..... 12-1
12.1.1.2. Single Ended Encoders 12-1
12.1.1.3. Software Setup for Single-ended Encoders..... 12-2
12.1.1.4. Aerotech Stepper Motors with the Home Marker Option..... 12-3
12.1.2. Encoder Signal Pinouts 12-6
12.1.3. Limit and Amplifier Fault Inputs..... 12-7
12.1.4. Current Command Output 12-8
12.1.5. Digital Input Bus Specifications..... 12-9
12.1.6. Output Bus Specifications..... 12-11
12.1.7. The Brake Output..... 12-12
12.1.8. Opto 22 I/O Bus..... 12-13
12.1.9. Main Connector Pinout of the UNIDEX 500..... 12-17
12.1.10. Opto 22 I/O Hall Inputs (P5) Pinouts..... 12-20
12.2. UNIDEX 500 ISA Technical Details 12-21
12.2.1. Test Points..... 12-21
12.2.2. Jumper Configurations 12-23
12.2.2.1. Marker Pulse Qualification Jumpers (JP28 – JP35) 12-24
12.2.2.2. Amplifier Enable Jumpers (JP36 – JP39) 12-25
12.2.2.3. PC Bus Interrupt Jumpers (JP10 – JP15)..... 12-27
12.2.2.4. Dual-Ported RAM Base Address Select Jumpers JP16 – JP24) 12-29
12.2.2.5. Base Address Jumpers (JP4 – JP9) 12-31
12.2.3. Emergency Stop Input..... 12-33
12.2.4. iSBX Connector Pinout (P4)..... 12-34
12.2.5. DSP Bus Pin Description (P3) 12-35
12.2.6. Encoder Interface between U500 and PC-PSO (P6) 12-36

12.2.7. Motorola DSP56002 OnCE Port (P8) 12-36
12.3. UNIDEX 500 PCI Technical Details 12-37
12.3.1. Emergency Stop Input 12-37
12.3.2. U500 PCI Analog Inputs 12-37
12.3.3. CLK/DIR Stepper Mode 12-38
12.3.4. Additional D/A Outputs 12-40
12.3.5. U500 PCI D/A Analog Outputs 12-40
12.3.6. U500 PCI Test Points 12-46
12.3.7. U500 Specifications 12-47
12.3.8. U500 PCI Jumpers 12-47
12.3.9. Connector Pinouts 12-49
12.3.9.1. P1 – Main Connector 12-49
12.3.9.2. P3 – Expansion Connector 12-50
12.3.9.3. P4 – 8X3 IO Conncetor 12-51
12.3.9.4. P5 - 16IN 8 OUT (OPTIONAL HALL INPUTS) 12-52
12.3.9.5. U500 PCI – P6 – Internal Encoder/Miscellaneous Interface 12-53
12.3.9.6. P8 – ½ Secondary Axis Interface 12-54
12.3.9.7. P9 – ½ Secondary Axis Interface 12-55

CHAPTER 13: TROUBLESHOOTING 13-1
13.1. Installation, Board Startup, and Communication Problems 13-2
13.2. Stepper Motors and Related Problems 13-3
13.3. Servo Related Problems 13-4
13.4. Problems Involving Fault Conditions 13-5
13.5. Homing Related Problems 13-7
13.6. General UNIDEX 500 Questions 13-8

APPENDIX A: GLOSSARY OF TERMS A-1

APPENDIX B: WARRANTY AND FIELD SERVICE B-1

APPENDIX C: PARAMETER SUMMARY C-1

APPENDIX D: REV_ BOARD TECHNICAL DETAILS D-1
D.1. Introduction D-1
D.2. Test Points D-1
D.3. Jumper Configurations D-3
D.4. Encoder Signal Specifications D-6
D.4.1. Differential Encoders D-6
D.4.2. Single Ended Encoders D-6
D.4.3. Software Setup for Single-ended Encoders D-7
D.4.4. Aerotech Stepper Motors with the Home Marker Option D-7
D.5. Encoder Signal Pinouts D-10
D.6. Amplifier Enable Outputs D-11
D.7. Main Connector Pinout of the UNIDEX 500 ..... D-12

APPENDIX E: SETTING UP AN AC BRUSHLESS MOTOR WITH THE UNIDEX 500 ..... E-1
E.1. Introduction E-1
E.2. Setup Procedure E-1

APPENDIX F: UNIDEX 500 APPLICATIONS F-1
F.1. Using the Encoder Position Capture Input - "UINT_N" F-1
F.2. Additional UINT features on the U500 PCI card. F-2
APPENDIX G: U500 CALIBRATION OPTION G-1
G.1. Introduction G-1
G.2. Axis Calibration G-2
G.3. Orthogonality Correction G-5
G.4. 2D Error Mapping G-8

∇ ∇ ∇

LIST OF FIGURES

Figure 1-1. The UNIDEX 500 System Diagram 1-1
Figure 1-2. The UNIDEX 500 Family of Controllers 1-2

Figure 3-1. Software Menu Available after Completed Installation 3-3
Figure 3-2. U500 Registry Editor 3-5
Figure 3-3. Board 1 (U500 Registry Editor) 3-6
Figure 3-4. The Edit Project File Popup 3-7
Figure 3-5. The Axis Position Window 3-8
Figure 3-6. Parameter Window 3-11
Figure 3-7. The Diagnostics Popup Window 3-12
Figure 3-8. The Diagnostics 2 Popup Window 3-13
Figure 3-9. The Axis Position Display (Choose Windows ➔ Axis Display ➔ Show) 3-14
Figure 3-10. Flowchart Summary of the Installation/Configuration Process 3-14

Figure 4-1. U500 Software Menu Structure 4-1
Figure 4-2. The Edit Project File Popup 4-5
Figure 4-3. The Edit Parameters Screen 4-6
Figure 4-4. The Defaults Tab of the MMI System Options Screen 4-7
Figure 4-5. Abort Options Tab of the MMI System Options Screen 4-11
Figure 4-6. LZR Setup Tab of the MMI System Options Screen 4-12
Figure 4-7. FPGA Setup Tab of the MMI System Options Screen 4-13
Figure 4-8. The Diagnostics Window 4-15
Figure 4-9. Forward Motion... Popup Window 4-22
Figure 4-10. Reverse Motion... Popup Window 4-23
Figure 4-11. Sample Rate Popup Window 4-23
Figure 4-12. Collect Submenu of the Axis Scope Screen 4-24
Figure 4-13. Display Submenu of the Axis Scope Screen 4-24
Figure 4-14. Axis Submenu of the Axis Scope Screen 4-25
Figure 4-15. Units Submenu of the Axis Scope Screen 4-25
Figure 4-16. Tools Submenu of the Axis Scope Screen 4-26
Figure 4-17. Cursors Tool Bar of the Axis Scope Screen 4-26
Figure 4-18. Status Tool Bar of the Axis Scope Screen 4-27
Figure 4-19. Control Tool Bar of the Axis Scope Screen 4-27
Figure 4-20. Gains Toolbar of the Axis Scope Screen 4-28
Figure 4-21. Frequency Response Toolbar of the Axis Scope Screen 4-28
Figure 4-22. FFT Toolbar of the Axis Scope Screen 4-29
Figure 4-23. U500 Joysticks 4-31
Figure 4-24. Joystick Digitizing Window 4-32
Figure 4-25. The Diagnostics 2 Window 4-33
Figure 4-26. The Variable Watch Window 4-34
Figure 4-27. Software Screen with All Windows Visible 4-36
Figure 4-28. Program Control Window after F4 Load of UNTITLED.PRG 4-37
Figure 4-29. The F5 Jog Window 4-39
Figure 4-30. System Password Prompt Window 4-40
Figure 4-31. System Password Window 4-40
Figure 4-32. The Software Status Bar 4-43
Figure 4-33. Status Cell Numbering 4-44

Figure 4-34. The Gain Adjust Popup 4-45
Figure 4-35. The MDI (Manual Data Interface) Window 4-45
Figure 4-36. The MFO Adjust Popup 4-46
Figure 4-37. The Axis Display Menu with Options 4-47
Figure 4-38. The Axis Display Window Showing Six Options 4-47
Figure 4-39. The Functions Menu 4-48
Figure 4-40. The Help Menu 4-49
Figure 4-41. The About Popup Window 4-50

Figure 5-1. The Edit Parameters Window 5-3
Figure 5-2. The Axis Positions Window With Position Information Cleared 5-6
Figure 5-3. Axis *n* Rollover in Rotary Stage Application 5-7
Figure 5-4. Programming Control Using a Single Plane 5-14
Figure 5-5. Programming Control Using Four Planes 5-15
Figure 5-6. Sample Programming Segment Showing the Use of Planes 5-16
Figure 5-7. Using *Home Offset* Parameter to Keep Gantry Aligned After Homing 5-22
Figure 5-8. Graphs of Linear and Sine Ramping Trajectories 5-24
Figure 5-9. Sample Motion Path Shown with and without Corner Rounding 5-26
Figure 5-10. Sample of Non-Ramp Time Overlap for Corner Rounding 5-26
Figure 5-11. Velocity Diagram of Non-Corner Rounding (G24) 5-27
Figure 5-12. MFO Potentiometer with and without Offsets 5-34
Figure 5-13. Contour Ramping (Acceleration/Deceleration) Time 5-35
Figure 5-14. Home Cycle 5-39
Figure 5-15. Typical Stage Showing CW and CCW Motor Rotation 5-40
Figure 5-16. Sample RMS Current Maximums 5-56
Figure 5-17. Motor and Encoder Rotation 5-64
Figure 5-18. Phase Advance Slope 5-82
Figure 5-19. U500 Filter Utility 5-87

Figure 6-1. UNIDEX 500 Servo Loop 6-2
Figure 6-2. Axis Scope Toolbars 6-2
Figure 6-3. The “Gains” and “Auto Tune” Toolbars. 6-5
Figure 6-4. Autotune Plot where “Steps” Has Been Set Too Low 6-6
Figure 6-5. Autotune Plot where “Steps” Has Been Set Too High 6-7
Figure 6-6. Autotune Plot Showing Proper Calibration 6-7
Figure 6-7. Flowchart of Overall Tuning Process 6-10
Figure 6-8. The Edit Parameters Window 6-11
Figure 6-9. Servo Loop Tab of the Edit Parameters Window 6-12
Figure 6-10. Maximize Button on the Axis Scope Window 6-12
Figure 6-11. Unacceptable Velocity Error 6-14
Figure 6-12. Acceptable Velocity Error (When Adjusting Kp) 6-14
Figure 6-13. Unacceptable Position Error (When Adjusting Ki) 6-15
Figure 6-14. Plot Showing an Appropriate Value for Kpos 6-16
Figure 6-15. Plot Showing Overall Effects When Kpos is High 6-17
Figure 6-16. Plot Showing Aff Adjustment 6-18
Figure 6-17. Tuning Plot of an AC Brushless Motor 6-19
Figure 6-18. Flowchart of Overall Tach Tuning Process 6-23
Figure 6-19. The Edit Parameters Window 6-24
Figure 6-20. Servo Loop Tab of the Edit Parameters Window 6-25
Figure 6-21. Cross-Section of the DS16020/16030 Amplifier 6-26
Figure 6-22. Amplifier Potentiometer Layout 6-26

Figure 6-23. Axis Scope Window 6-28
Figure 6-24. Plot Showing a Roughly Tuned Axis (When Adjusting Kpos) 6-29
Figure 6-25. Plot Showing the Removal of DC Offsets in the Position Error 6-30
Figure 6-26. O-scope Showing Current Feedback for One Move 6-30
Figure 6-27. Plot Illustrating Smoothness in the Position Error 6-31
Figure 6-28. Plot Showing Effects on Position Error (When Ki is too High) 6-32
Figure 6-29. Plot of the Position Error With Appropriate Ki Value 6-33
Figure 6-30. Position Error After Increasing Vff 6-34
Figure 6-31. Position Error Reduced to Within 10 Counts of Error Using Vff 6-34
Figure 6-32. Plot of Position Error When Vff is too High 6-35

Figure 7-1. Circle Center Adjustment 7-40
Figure 7-2. Startup Moves 7-53
Figure 7-3. Ending Moves 7-53
Figure 7-4. Cutter Compensation Example 7-55
Figure 7-5. CYCLE START Function 7-56
Figure 7-6. Sample Uses of the GOTO Command 7-72
Figure 7-7. Sine-Type Ramp 7-84
Figure 7-8. LVDT Sensor 7-87
Figure 7-9. PAUSE Function 7-100
Figure 7-10. Illustration of No Corner Rounding (G24) 7-113
Figure 7-11. Illustration of Corner Rounding (G23) 7-113
Figure 7-12. Optional UNIDEX 500 Joysticks (JBV Model Left, JI Model Right) 7-124
Figure 7-13. Plot of Velocity Without Velocity Profiling 7-136
Figure 7-14. Plot of Velocity With Velocity Profiling 7-136
Figure 7-15. Short Middle Move With No Velocity Profiling 7-137
Figure 7-16. Short Middle Move With Velocity Profiling 7-137
Figure 7-17. Plot Of Velocity With Velocity Profiling In Contour Mode 1. 7-138
Figure 7-18. Same Motion With Ramping Time Reduced 7-138
Figure 7-19. Circular Profiling With Long Ramp Time 7-139
Figure 7-20. Circular Profiling With Short Ramp Time 7-139
Figure 7-21. Two Axis Linear Move With Velocity Profiling 7-140
Figure 7-22. Velocity Profile for Non-tangential Vectors 7-141
Figure 7-23. Velocity Profile With Digital Filter 7-142
Figure 7-24. Velocity Profile Without a G9 Command at the End of the Sequence 7-143

Figure 8-1. Pulse Output Circuit Diagram 8-2
Figure 8-2. Single PSO Mode Block Diagram 8-12
Figure 8-3. Dual PSO Mode Block Diagram 8-13

Figure 9-1. Sample Path for Incremental (Relative) Motion Demonstration Using Velocity Profiling 9-2
Figure 9-2. Sample Path for Absolute Motion Example Using Velocity Profiling 9-5
Figure 9-3. Sample Path of Square with and without the Rounding Feature 9-8
Figure 9-4. Output from Splining Example 9-13
Figure 9-5. Output from Parts Rotation Example Program 9-15
Figure 9-6. Output from Overriding Scale Factor Example Program 9-18

Figure 12-1. Electrical Characteristics of a Single Ended Encoder Interface 12-2
Figure 12-2. Electrical Characteristics of an Aerotech Stepper Motor Home Marker Option 12-3
Figure 12-3. Electrical Characteristics of the UNIDEX 500 Limit Interface 12-8
Figure 12-4. Electrical Characteristics of the UNIDEX 500 Current Command Output 12-8
Figure 12-5. Electrical Characteristics of the UNIDEX 500 Input Bus Interface 12-10
Figure 12-6. Example Optocoupled Output Bus Interface 12-12
Figure 12-7. Electrical Characteristics of the UNIDEX 500 Brake Signal Output 12-13
Figure 12-8. Electrical Characteristics of the UNIDEX 500 Opto 22 Connections 12-16
Figure 12-9. Electrical Characteristics of the UNIDEX 500 Opto 22 Connections (page 2) 12-17
Figure 12-10. UNIDEX 500 PC Board Jumper Locations 12-23
Figure 12-11. Typical Connection of the U500 to an Opto-Isolator 12-27
Figure 12-12. Electrical Characteristics of the UNIDEX 500 Emergency Stop Interface 12-33
Figure 12-13. U500 PCI Emergency Stop Input 12-37
Figure 12-14. CLK/DIR Schematic 12-39
Figure 12-15. U500 PCI CLK/DIR Stepper Section 12-39
Figure 12-16. U500 PCI D/A Output Diagram 12-41
Figure 12-17. U500 PCI Connection to BB500 (Isolated) 12-42
Figure 12-18. U500 PCI Connection to BB500 (Non-Isolated) 12-43
Figure 12-19. U500 PCI Connection to BB501 (Isolated) 12-44
Figure 12-20. U500 PCI Connection to BB501 (Non-Isolated) 12-45
Figure 12-21. U500 PCI Board 12-56

Figure D-1. UNIDEX 500 PC Board Jumper Locations D-5
Figure D-2. Electrical Characteristics of a Single Ended Encoder Interface D-6
Figure D-3. Electrical Characteristics of an Aerotech Stepper Motor Home Marker Option D-8
Figure D-4. Electrical Characteristics of the UNIDEX 500 Amplifier Enable Output D-11

Figure E-1 U500 Diagnostic Window E-3

Figure F-1. U500PCI UINT-N Circuit F-3

Figure G-1. Sample ASCII Calibration File G-3
Figure G-2. Sample Single-Row Orthogonality Correction File G-6
Figure G-3. Sample Multi-Row Orthogonality Correction File G-6
Figure G-4. Multi-Row Orthogonality Correction Grid G-7
Figure G-5. Calibration File with Correction Point Data G-8
Figure G-6. 2D Correction Grid G-10
Figure G-7. 2D Correction Interpolation G-11

∇ ∇ ∇

LIST OF TABLES

Table 1-1. Available PC Bus-based Motion Controllers in the UNIDEX 500 Series 1-3
Table 1-2. Available Motor Drivers compatible with UNIDEX 500 and DR500 Amplifier Chassis 1-4
Table 1-3. Available DR500 Amplifier Chassis Options 1-4
Table 1-4. Options and Accessories Available for the UNIDEX 500 1-5

Table 3-1. Minimum Hardware Requirements and Recommendations 3-1
Table 3-2. Base Address Jumper Settings 3-5
Table 3-3. Key UNIDEX 500 Configuration Parameters 3-10
Table 3-4. Servo Loop Tuning Parameters 3-16

Table 4-1. File Menu Commands 4-2
Table 4-2. Edit Menu Commands 4-4
Table 4-3. Components of the "Edit Project File" Popup 4-5
Table 4-4. Pre-Defined Global Subroutine Labels 4-8
Table 4-5. Tools Menu Commands 4-15
Table 4-6. Software Status Diagnostics 4-16
Table 4-7. Axis Position Diagnostics 4-17
Table 4-8. Hardware Status Diagnostics 4-18
Table 4-9. Menu Items on the Axis Scope Screen 4-20
Table 4-10. File Submenu Options in Axis Scope 4-20
Table 4-11. Plot Submenu Options in Axis Scope 4-21
Table 4-12. Trigger Submenu Options in Axis Scope 4-22
Table 4-13. File Submenu Options of the FFT screen 4-29
Table 4-14. Plot Submenu Options of the Axis Scope Screen 4-29
Table 4-15. Windows Menu Commands 4-35
Table 4-16. Password Protected Access Level Options 4-41
Table 4-17. Cell Descriptions of the Status Bar 4-44
Table 4-18. Axis Display Options 4-47

Table 5-1. Position Tracking Parameters (Tab 0) 5-5
Table 5-2. Settings for Parameter 001 5-6
Table 5-3. Settings for Parameters 011, 012, 013, and 014 5-7
Table 5-4. Commonly Used Settings for the DPRAM Base Address 5-8
Table 5-5. Commonly Used Settings for the Host Base Address Parameter 016 5-9
Table 5-6. Settings for Parameters 020, 038, 056, and 074 5-10
Table 5-7. Parameter Associations between Planes, Measurement Units, and the Number of Decimal Digits 5-10
Table 5-8. Settings for Parameters 029, 047, 065, and 083 5-11
Table 5-9. Settings for Parameters 030, 048, 066, and 084 5-12
Table 5-10. Advanced Motion Parameters (Tab 1) 5-13
Table 5-11. Settings for Parameter 000 5-17
Table 5-12. Settings for Parameters 003, 004, 005, and 006 5-19
Table 5-13. Settings for Parameters 007, 008, 009, and 010 5-20
Table 5-14. Settings for Parameters 018, 036, 054, and 072 5-23
Table 5-15. Settings for Parameters 021, 039, 057, and 075 5-24
Table 5-16. Settings for Parameters 027, 045, 063, and 081 5-25

Table 5-17. Settings for Parameters 028, 046, 064, and 082 5-27
Table 5-18. Settings for Parameters 31, 49, 67, and 85 5-28
Table 5-19. Joystick Deadband Parameters 5-29
Table 5-20. Joystick Center Position Parameters 5-30
Table 5-21. Settings for Parameter 099 5-31
Table 5-22. Settings for Parameter 500 5-32
Table 5-23. Settings for Parameter 501 5-32
Table 5-24. Basic Motion Parameters (Tab 2) 5-33
Table 5-25. Settings for Parameter 002 5-34
Table 5-26. Settings for Parameters 019, 037, 055, and 073 5-35
Table 5-27. Settings for Parameters 022, 040, 058, and 076 5-36
Table 5-28. Point-to-point Feedrate Parameter Assignments and Settings 5-36
Table 5-29. Homing and Limits Parameters (Tab 3) 5-37
Table 5-30. Settings for Parameter x02 5-40
Table 5-31. Settings for Parameter x03 5-41
Table 5-32. Settings for Parameter x04 5-42
Table 5-33. Settings for Parameter x05 5-43
Table 5-34. Settings for Parameter x06 5-43
Table 5-35. Settings for Parameter x07 5-44
Table 5-36. Settings for Parameter x08 5-45
Table 5-37. Settings for Parameter x09 5-45
Table 5-38. Settings for Parameter x22 5-46
Table 5-39. Settings for Parameter x23 5-47
Table 5-40. Settings for Parameter x73 5-47
Table 5-41. Settings for Parameter x74 5-48
Table 5-42. Safe Zone Limit Parameters 5-48
Table 5-43. Settings for Parameter x77 5-49
Table 5-44. Settings for Parameter x81 5-49
Table 5-45. Trap Parameters 5-50
Table 5-46. Settings for Parameter x16 5-51
Table 5-47. Settings for Parameter x17 5-52
Table 5-48. Settings for Parameter x18 5-53
Table 5-49. Settings for Parameter x19 5-54
Table 5-50. Settings for Parameter x20 5-55
Table 5-51. Settings for Parameter x48 5-57
Table 5-52. Settings for Parameter x49 5-58
Table 5-53. Settings for Parameter x53 5-59
Table 5-54. Settings for Parameter x54 5-60
Table 5-55. Settings for Parameter x70 5-61
Table 5-56. Settings for Parameter x84 5-61
Table 5-57. Motor Feedback Parameters 5-62
Table 5-58. Summary of General Purpose Inputs 5-67
Table 5-59. Commutation Factors for 4, 6, and 8 Poles 5-68
Table 5-60. Factory Configuration for UNIDEX 500 RDP 5-68
Table 5-61. Feedback Channels, Types, and Additional Hardware 5-70
Table 5-62. Feedback Setup Codes, and the Associated Actions 5-70
Table 5-63. RDP Resolution and Setup Codes 5-70
Table 5-64. Motor Feedback and Configuration Parameters 5-71
Table 5-65. Settings for Parameter x38 5-72
Table 5-66. Encoder Gantry Mode Configuration Parameters 5-74
Table 5-67. Settings for Parameter x39 5-75
Table 5-68. Settings for Parameter x40 5-76

Table 5-69. Settings for Parameter x41 5-76
Table 5-70. Settings for Parameter x42 5-77
Table 5-71. Sample Commutation Factors for AC Brushless Motors 5-78
Table 5-72. Settings for Parameter x64 5-81
Table 5-73. Settings for Parameters x79 and x80 5-83
Table 5-74. Settings for Parameter x82 5-83
Table 5-75. Servo Loop Parameters (Tab 6) 5-84
Table 5-76. Settings for Parameter x24 5-85
Table 5-77. Settings for Parameter x62 5-92
Table 5-78. Bits of Parameter x78 and PID Loop Configuration 5-92
Table 5-79. Settings for parameter x83 5-93
Table 5-80. Other (Miscellaneous) Parameters (Tab 7) 5-94
Table 5-81. Relationship Between Number of Decimal Digits Parameters and the Number of Programming Steps per Programming Unit 5-96
Table 5-82. Settings for Parameter x11 5-98
Table 5-83. Settings for Parameter x12 5-99
Table 5-84. Settings for Parameter x13 5-99
Table 5-85. Settings for Parameter x14 5-100
Table 5-86. Settings for Parameter x35 5-101
Table 5-87. Settings for Parameter x37 5-102
Table 5-88. Settings for Parameter x50 5-103
Table 5-89. Settings for Parameter x51 5-103
Table 5-90. Settings for Parameter x52 5-104
Table 5-91. Settings for Parameter x71 5-104
Table 5-92. Fault Parameters (Tab 8) 5-106
Table 5-93. Fault Mask Bit Descriptions 5-107

Table 6-1. Initial Servo Parameter Values 6-12
Table 6-2. Servo Gain Values 6-21
Table 6-3. Initial Servo Parameter Values - Tach Tuning 6-25

Table 7-1. Programming Conventions Used in This Manual 7-1
Table 7-2. Single Character Arguments for Programming 7-2
Table 7-3. Supported Functions 7-4
Table 7-4. Mathematical Operators and Their Evaluation Hierarchy 7-5
Table 7-5. Relative Position Registers 7-6
Table 7-6. Absolute Position Registers 7-6
Table 7-7. Real Time Feedback Position Registers 7-7
Table 7-8. Real Time Commanded Position Registers 7-7
Table 7-9. U500 ISA A/D Channel Registers 7-8
Table 7-10. U500 PCI A/D Channel Registers 7-9
Table 7-11. ISBX Control word Configurations 7-12
Table 7-12. 4EN I/O Connector Interface 7-14
Table 7-13. U500 Programming Command Summary 7-15
Table 7-14. Optional Arguments 7-25
Table 7-15. Analog Input Locations 7-26
Table 7-16. Arguments for Analog Command 7-28
Table 7-17. Comparison Operators 7-76
Table 7-18. Motor Phase Labels and Hall States 7-92
Table 7-19. Arguments for In Position Command Syntax 7-105
Table 7-20. Setting for the ‘flags’ argument of the Slave Command 7-120

Table 8-1. Opto Coupler Options 8-1
Table 8-2. Arguments for the FIRE Command 8-3
Table 8-3. Arguments for Array Command 8-6
Table 8-4. Arguments for Window Command 8-7
Table 8-5. Arguments for Dual Command 8-10
Table 8-6. PSO Commands 8-14

Table 10-1. Software Development Platforms and Libraries 10-1
Table 10-2. Quick Library Functions 10-4
Table 10-3. Arguments and Return Values for the aerq_read_status Function 10-18
Table 10-4. WAPI Library Functions 10-33
Table 10-5. Arguments and Return Values for the WAPIAerReadStatus Function 10-40

Table 12-1. Termination Resistor Configuration for Axis 1 Encoders 12-4
Table 12-2. Termination Resistor Configuration for Axis 2 Encoders 12-4
Table 12-3. Termination Resistor Configuration for Axis 3 Encoders 12-5
Table 12-4. Termination Resistor Configuration for Axis 4 Encoders 12-5
Table 12-5. Encoder Signals and Pinouts 12-6
Table 12-6. Limit and Amplifier Fault Inputs 12-7
Table 12-7. Current Command Output Signals and Pin Locations 12-9
Table 12-8. UNIDEX 500 Inputs and Locations 12-10
Table 12-9. UNIDEX 500 Output Pinouts 12-11
Table 12-10. UNIDEX 500/Opto 22 Connection Information 12-14
Table 12-11. Main Connector Pinouts for the UNIDEX 500 ISA 12-18
Table 12-12. Analog Input Locations for the UNIDEX 500 ISA 12-19
Table 12-13. Pinouts for Opto 22 I/O Hall Inputs (P5) 12-20
Table 12-14. Motor Related Test Points 12-21
Table 12-15. PC Interface to the UNIDEX 500 12-22
Table 12-16. DSP 56002 Chip Test Points 12-22
Table 12-17. Miscellaneous Test Points 12-22
Table 12-18. Marker Pulse Qualification Jumper Settings 12-24
Table 12-19. Amplifier Enable Jumper Settings 12-26
Table 12-20. PC Bus Interrupt Jumper Settings 12-28
Table 12-21. Dual Ported RAM Base Address Select Jumpers Settings 12-29
Table 12-22. Miscellaneous Jumpers 12-30
Table 12-23. Base Address Jumper Settings 12-32
Table 12-24. External Voltages and Resistances for the Emergency Stop Input 12-34
Table 12-25. iSBX Connector Pinouts 12-34
Table 12-26. DSP Bus Pinouts 12-35
Table 12-27. Pinouts for the Encoder Interface to the PC-PSO 12-36
Table 12-28. OnCE Port Pinouts (P8) 12-36
Table 12-29. U500 PCI Emergency Stop Electrical Specifications 12-37
Table 12-30. Allowable Combinations for U500 BASE/PLUS 12-38
Table 12-31. Allowable Combinations for U500 ULTRA 12-38
Table 12-32. U500 PCI CLK/DIR Stepper Mode Jumpers 12-40
Table 12-33. Non-Isolated System Jumpers 12-40
Table 12-34. Isolated System Jumpers (external power supply required) 12-41
Table 12-35. D/A Output Specifications 12-41
Table 12-36. U500 PCI Test Points 12-46
Table 12-37. U500 Specifications: Current Consumption 12-47
Table 12-38. U500 PCI Jumpers 12-47

Table 12-39. P1 – Main Connector Pinouts 12-49
Table 12-40. P3 – Expansion Connector Pinouts 12-50
Table 12-41. P4 – 8X3 IO Connector Pinouts 12-51
Table 12-42. P5 – 16 In 8 Out (Optional Hall Inputs) 12-52
Table 12-43. U500 PCI – P6 – Internal Encoder/Misc. Interface Pinouts 12-53
Table 12-44. P8 – ½ Secondary Axis Interface Pinouts 12-54
Table 12-45. P9 – ½ Secondary Axis Interface Pinouts 12-55

Table 13-1. Troubleshooting for Common Installation, Startup, and Communication Problems 13-2
Table 13-2. Troubleshooting for Stepper Motors (and Related) Problems 13-3
Table 13-3. Troubleshooting for Servo Related Problems 13-4
Table 13-4. Troubleshooting for Problems Involving Fault Conditions 13-5
Table 13-5. Troubleshooting for Homing Related Problems 13-7

Table C-1. General Parameters C-1
Table C-2. Parameters Dedicated to Planes 1, 2, 3, and 4 ..... C-4
Table C-3. Axis Parameters C-5

Table D-1. Motor Related Test Points D-1
Table D-2. PC Interface to the UNIDEX 500 ..... D-2
Table D-3. DSP 56001 Chip Test Points D-2
Table D-4. Miscellaneous Test Points D-3
Table D-5. PC Base Address Jumper Settings D-3
Table D-6. Jumper Configurations D-4
Table D-7. Termination Resistor Configuration for Axis 1 Encoders D-8
Table D-8. Termination Resistor Configuration for Axis 2 Encoders D-9
Table D-9. Termination Resistor Configuration for Axis 3 Encoders D-9
Table D-10. Termination Resistor Configuration for Axis 4 Encoders D-9
Table D-11. Encoder Signals and Pinouts D-10
Table D-12. Amplifier Enable Output Locations D-11
Table D-13. Main Connector (P1) Pinouts for the UNIDEX 500 ..... D-12

Table E-1. Hall State Table E-4

Table F-1. Pin Locations of UINT_N Signals F-1

Table G-1. Sample Calibration Data G-4

∇ ∇ ∇

DECLARATION OF CONFORMITY

Manufacturer's Name and Address:

Aerotech, Inc.
101 Zeta Drive
Pittsburgh, PA 15238-2897

Declares that the product:

Product Name: UNIDEX 500

Conforms to the following product specifications:

EMC: EN 55011:1991 Class B Emissions
EN 50082-1: 1992 Immunity
IEC 801-2: 1984
IEC 801-3: 1984
IEC 801-4: 1988

and complies with EC directive 89/336/EEC.

Pittsburgh, PA
June, 1996

Dave Kincel _____
Quality Assurance Manager

Alex Weibel _____
Engineer Certifying Compliance

General notes concerning the test setup:

This product was tested at Washington Laboratories, LTD in Gaithersburgh, MD on December 12,1995. The report is WLL 2986F.

The UNIDEX 500 was tested in a CE–compliant, class B personal computer and it controlled two DC motors, one brushless AC motor and one stepper motor through a DR500 Drive Chassis. The following modifications ensure compliance with EMC directive 89/336/ECC.

- Add two ferrites (P/N: TDK ZCAT3035-13304 or the equivalent) to the DIO and OP500 cables. Locate a ferrite on each end of both cables.
- Install a 25–pin Sub–D filter (P/N: SCI-56-725-001 or the equivalent) on the DR500 axis limit connectors that interface to each motor.
- Add one ferrite (P/N: TDK ZCAT3035-13304 or the equivalent) and two toroids to the DR500 power cord.

- Install a Schaffner filter (P/N: FN2080-10-06 or the equivalent) on the DR500 AC power input.
- Bond shields of all cables to DR500 chassis.
- Add ferrite (P/N: Steward 28B-029-0A0 or the equivalent) to each motor cable at the DR500.

Failure to follow the described procedures may cause excessive emissions or reduced immunity.

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: OVERVIEW

This chapter contains an overview of the UNIDEX 500 motion controller as well as a sample system diagram. This chapter also contains precautionary notes about installing and using the UNIDEX 500 motion controller.

CHAPTER 2: GETTING STARTED

This chapter contains information about the components of the UNIDEX 500 system, unpacking and inspecting the equipment, and minimum hardware and software requirements for proper operation.

CHAPTER 3: SOFTWARE INSTALLATION AND FUNDAMENTALS

This chapter contains information about the U500 Interface software that is supplied with the UNIDEX 500 system. Included is an explanation of how to install the Interface software for proper operation, starting the Interface software, using a mouse, alternate keyboard commands, and configuring the software for proper operation.

CHAPTER 4: THE SOFTWARE INTERFACE

This chapter provides a complete list of menu items and options that are provided through the UNIDEX 500 software interface. Sample screens are illustrated for all options.

CHAPTER 5: PARAMETERS

This chapter provides information that helps the operator to understand and configure the parameters within the UNIDEX 500 system. Appropriate parameter configuration optimizes the UNIDEX 500 for an application. This chapter includes detailed discussions of all software parameters, motor and feedback configurations, and other topics related to the operation and configuration of the UNIDEX 500 system.

CHAPTER 6: TUNING SERVO LOOPS

This chapter provides information about servo loops and proper tuning techniques.

CHAPTER 7: PROGRAMMING COMMANDS

This chapter supplies information required in understanding the UNIDEX 500 programming environment. This chapter also includes an in-depth discussion of individual programming commands.

CHAPTER 8: UNIDEX 500 PCI PULSE OUTPUT

This chapter provides information about the pulse output commands.

CHAPTER 9: SAMPLE PROGRAMS

This chapter contains sample applications, highlighting UNIDEX 500 features, parameter settings and sample programs.

CHAPTER 10: PROGRAMMING TOOLS

This chapter introduces programming tools such as C and QuickBASIC functions that allow the operator to create a customized operator interface rather than using the Toolkit or MMI interface.

CHAPTER 11: UTILITY PROGRAMS

This chapter lists the utility programs that are included with the U500 installation. Each utility is explained in detail, including the purpose of the utility, the syntax for invoking the utility, necessary inputs, expected results and any technical notes or helpful details.

CHAPTER 12: TECHNICAL DETAILS

This chapter supplies a variety of technical specifications for the UNIDEX 500. These specifications include test points, jumper configurations, encoder signal specifications, pinouts, outputs, bus specifications, and others.

CHAPTER 13: TROUBLESHOOTING

This chapter provides a reference tool if problems with the UNIDEX 500 arise.

APPENDIX A: GLOSSARY OF TERMS

Appendix A contains a list of definitions of terms used in this manual.

APPENDIX B: WARRANTY AND FIELD SERVICE

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

APPENDIX C: PARAMETER SUMMARY

Appendix C contains a reference of all UNIDEX 500 parameters.

APPENDIX D: REV_BOARD TECHNICAL DETAILS

Appendix D contains information on test points, jumpers and other technical details for Revision _ PC boards, an earlier production version of the UNIDEX 500. The remainder of the manual pertains to the latest production version of the U500, Revision C.

APPENDIX E: SETTING UP AN AC MOTOR WITH THE UNIDEX 500

Appendix E contains a procedure for setting up AC brushless motors with the U500.

APPENDIX F UNIDEX 500 APPLICATIONS

Appendix F contains information regarding specific U500 applications.

APPENDIX G UNIDEX 500 CALIBRATION OPTIONS

Appendix G contains reference information about the three types of corrections available with the U500: axis calibration correction, orthogonality correction, and 2D error mapping.

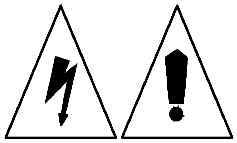
INDEX

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 may appear in one of four possible styles: standard serif font (e.g., 3-1) for normal references, boldface font (e.g., **3-1**) for references to illustrations, italic font (e.g., *3-1*) for references to tables, or boldface italic font (e.g., ***3-1***) for references to illustrations within tables.

MANUAL CONVENTIONS

Throughout this manual the following conventions are used:

- Use of "n" within a program block signifies that any axis (X, Y, Z or U) or drive (1, 2, 3, or 4) may be inserted
- Capitalized letters within a command indicate the minimum entry for that command (e.g., DIsable)
- The terms UNIDEX 500 and U500 are used interchangeably throughout this manual
- *Italic font* is used to illustrate syntax and arguments for programming commands
- Underline letters refer to an <ALT> - letter keystroke
- Keys such as Shift, Ctrl, Alt and Enter are enclosed in brackets (e.g., <Shift>, <Ctrl>, <Alt> and <Enter>) to distinguish them from individual keystrokes.
- The text <Enter> and the symbol ↵ are used interchangeably throughout this manual to indicate that the Enter/Return key on the keyboard is to be pressed.
- Hexadecimal numbers are listed using a preceding "0x" (for example, 0x300, 0x12F, 0x01EA, etc.,) to distinguish them from decimal numbers
- An "x" preceding a parameter number represents the axis number (1, 2, 3 or 4) for the corresponding axis (X, Y, Z or U, respectively). Therefore, parameter x38 (the Primary Feedback Channel), for example, actually corresponds to four distinct parameters:
  - 138 for the Primary Feedback Channel of axis X
  - 238 for the Primary Feedback Channel of axis Y
  - 338 for the Primary Feedback Channel of axis Z, and
  - 438 for the Primary Feedback Channel of axis U.
- The terms <Enter> key and <Return> key are used interchangeably throughout this document when referring to the keyboard.
- Within the index, a bold locator page number (e.g., Components, **1-1**) indicates that the reference is part of an illustration. An italic locator page number (e.g., OP500 Cable Pinouts, *5-7*) indicates that the reference is part of a table. Text references are shown in a standard serif font (e.g., Software Setup, 3-1).



- Graphic icons or keywords may appear in the outer margins to provide visual references of key features, components, operations or notes.
- Danger and/or Warning symbols (see left) appear in the outer margins next to important precautions. Failure to observe these precautions could result in serious injury and/or damage to the equipment.
- The following statements apply wherever a Warning or Danger symbol appears within this manual. Failure to observe these precautions could result in serious injury to those performing the procedures and/or damage to the equipment.

> To minimize the possibility of electrical shock and bodily injury, make certain that all of the electrical power switches are in the off position prior to making any electrical connections.

> To minimize the possibility of electrical shock and bodily injury when any electrical circuit is in use, ensure that no person comes in contact with the circuitry.

> When this controller is installed within a system, mechanical motion will occur. Care must be exercised that all personnel remain clear of any moving parts.

> To minimize the possibility of bodily injury, make certain that all electrical power switches are in the off position prior to making any mechanical adjustments.

- This manual uses the symbol "∇ ∇ ∇" to indicate the end of a chapter.

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.

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. Readers are encouraged to critique the manual and offer their feedback by completing the form and either mailing or faxing it to Aerotech.

∇ ∇ ∇

CHAPTER 1: INTRODUCTION

In This Section:

- Overview of the UNIDEX 500 System 1-1
- The UNIDEX 500 Family of Controllers 1-1
- Ordering Information.....1-3
- Options and Accessories.....1-5
- Safety Procedures and Warnings.....1-6

1.1. Overview of the UNIDEX 500 System

The UNIDEX 500 (or U500) system is a combination of the U500 PC bus-based motion control card, the Windows-based Toolkit or MMI interface software, and any number of optional accessories. The U500 system integrates with amplifiers, positioning stages, and these accessories to form a complete, programmable, customized control system that is suitable for a wide range of motion control applications. A typical system is illustrated in Figure 1-1.

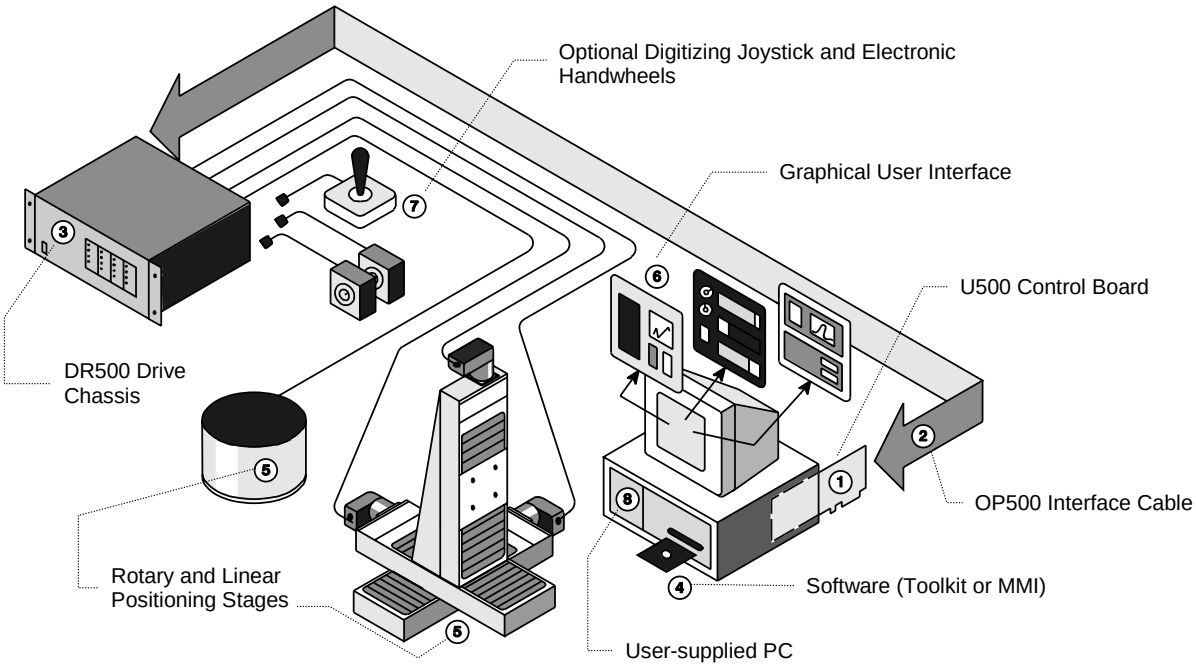


Figure 1-1. The UNIDEX 500 System Diagram

1.2. The UNIDEX 500 Family of Controllers

There are two versions of the UNIDEX 500 card that support two different bus architectures, the ISA bus and the PCI bus. Each version of the UNIDEX 500 control card is available in three models (Base, Plus and Ultra) to provide a level of motion control that will match any application. A U500 board is identified using a small label that is located on the component side of the board itself. This label will show the model of the U500 board. Refer to Figure 1-2.

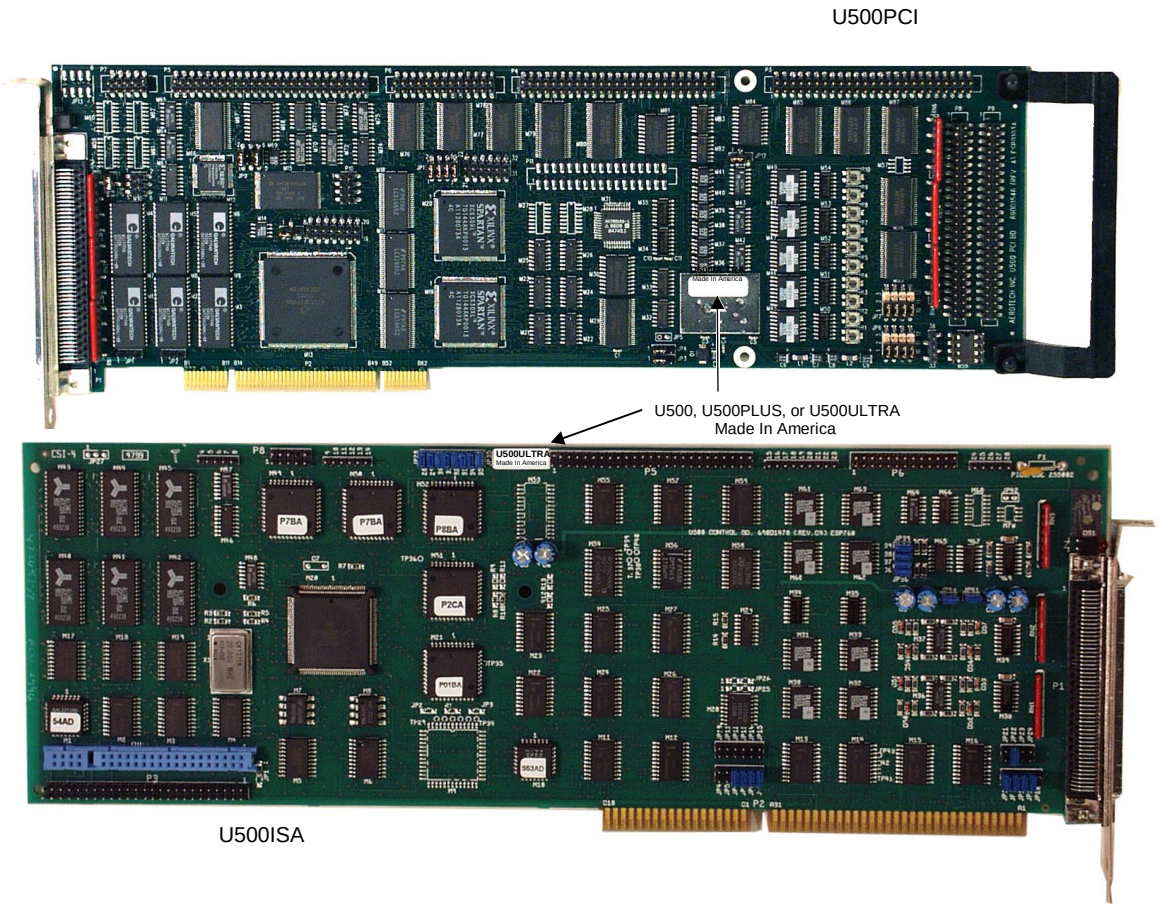


Figure 1-2. The UNIDEX 500 Family of Controllers

1.2.1. The UNIDEX 500 Base Model (U500)

The UNIDEX 500 base model is the basic version of the U500 family. A base model UNIDEX 500 is identified with a small label that reads "UNIDEX 500". This label is located on the component side of the PC board. Refer to Figure 1-2.

The U500 is capable of performing the following functions:

- Programmable multitasking
- Point-to-point type motion
- Absolute or incremental positioning
- Constant velocity motions
- Velocity profiled motions
- Time-based motions (arriving at a set location in a desired amount of time)
- Freerun (unsynchronized motion)
- Linear interpolation of 2, 3, or 4 axes
- Electronic gearing
- Gantry modes
- Autotuning
- Data Collection

1.2.2. The UNIDEX 500 Plus Model (U500PLUS)

The UNIDEX 500 Plus model is the version of the U500 family to use if the operator requires basic motion control functions, error mapping, and the ability to program circular-type moves. The Plus model UNIDEX 500 is identified with a small label that reads "UNIDEX 500PLUS". This label is located on the component side of the PC board. Refer to Figure 1-2.

The U500PLUS adds the following functions to those of the base model:

- Circular interpolation (any two pairs, up to 360°)
- Helical interpolation (circular motion of one axis pair plus one or two axes of linear motion)
- Increased RAM
- Axis calibration and error mapping (per axis)

1.2.3. The UNIDEX 500 Ultra Model (U500ULTRA)

The UNIDEX 500 Ultra model is the most versatile version of the U500 family of controllers. This version combines standard motion functions with spline table positioning. In addition, this version also is equipped with an Intel-compatible iSBX expansion port (required when using any U500 option) and an expansion bus port that provides direct access to the digital signal processing (DSP) bus. The Ultra model UNIDEX 500 is identified with a small label that reads "UNIDEX 500ULTRA". This label is located on the component side of the PC board. Refer to Figure 1-2.

The U500ULTRA adds the following functions to those of the Plus model:

- Cubic spline interpolation
- Cutter radius compensation
- Part rotation and scaling
- Increased RAM
- Output commands linked to spline table points.
- On-board pulse synchronized output (PSO) laser firing (available on the PCI version only).

1.3. Ordering Information

Table 1-1 lists the motion controllers in the U500 series that are available from Aerotech, Inc. For complete ordering information, refer to the Aerotech Motion Control Catalog.

Table 1-1. Available PC Bus-based Motion Controllers in the UNIDEX 500 Series

Controller	Description
U500BASE / U500BASE-PCI	1-4 axes, PC bus-based, point-to-point motion, absolute and incremental positioning, constant velocity motion, velocity profiled motions, time-based motions, free run motion, linear interpolation of 2-4 axes, and electronic gearing by ratio
U500PLUS / U500PLUS-PCI	adds circular/helical interpolation, more RAM, and axis calibration and error mapping capabilities
U500ULTRA/ U500ULTRA-PCI	adds spline table motion, electronic gearing, more RAM, iSBX and local bus expansion ports.

Table 1-2 lists the Aerotech motor drivers that can be used when connecting the UNIDEX 500 to the DR500 amplifier drive chassis.

Table 1-2. Available Motor Drivers compatible with UNIDEX 500 and DR500 Amplifier Chassis

Motor Driver	Description
AM16007C / AM8007C (Microstepping)	Microstepping: 80V/160V, 6A RMS cont., 20 kHz PWM, 3U height stepping motor driver
DS16020C (DC Brush)	DC Brush: 160V, 10A cont. 20A peak, 20 kHz PWM, 3U height, DC servo motor driver
DS16030C (DC Brush)	DC Brush: 160V, 15A cont. 30A peak, 20 kHz PWM, 3U height, DC servo motor driver
AS32020 / AS32030 /	160V, 10/15A RMS cont., 20/30 A peak, 20 kHz PWM, 3U height, brushless motor driver
AS3005 (Brushless)	30V, 5A pk, 3U height, linear brushless motor driver

Table 1-3 lists the DR500 amplifier chassis options that are available from Aerotech for the UNIDEX 500 series motion controllers:

Table 1-3. Available DR500 Amplifier Chassis Options

<table>
<tr><th colspan="5">Part Number: DR500x-y-vv-vv/</th></tr>
<tr><td>DR500</td><td>X</td><td>Y</td><td>VV</td><td>VV</td></tr>
<tr><td>Base</td><td>Package</td><td>Input</td><td>Vbus1</td><td>Vbus2</td></tr>
<tr><td rowspan="5">DR500</td><td rowspan="2">S = Desktop</td><td>A = 115VAC</td><td colspan="2">0 = no bus</td></tr>
<tr><td>B = 230VAC</td><td colspan="2">30 = 30VDC</td></tr>
<tr><td rowspan="3">R = Rack Mount</td><td>C = 100VAC</td><td colspan="2">40 = 40VDC</td></tr>
<tr><td>D = 208VAC</td><td colspan="2">80 = 80VDC</td></tr>
<tr><td colspan="2">160 = 160VDC</td></tr>
</table>

EXAMPLE:

DR500R-A-40-80/

1.4. Options and Accessories

A variety of options and accessories may be purchased with the UNIDEX 500 to enhance its standard operation. Table 1-4 lists the Aerotech options and accessories that can be used with the UNIDEX 500 Series motion controllers. Brief descriptions of each option follow.

Table 1-4. Options and Accessories Available for the UNIDEX 500

Accessories	Description
DR500	External drive chassis for use with the U500
BB500	Breakout module for the U500 when not using DR500 chassis
HW500	3.6 inch handwheel assembly and cable (25-pin male “D”)
HW500-1	3.6 inch handwheel assembly and cable (9-pin male “D”)
HW500-2	3.6 inch handwheel assembly with flying leads
JBV-xx / JI	Joystick with digitizing capability
OP500	Interconnection cable from controller to the DR500 chassis, BB500, and BB501
PSO-PC	Programmable, PC bus-based, position synchronized, laser firing control card used to provide output signals based on the positions of up to three axes (available on the U500ULTRA only, on-board for U500ULTRA-PCI)
RDP-PC	Four-channel, PC bus-based, resolver-to-digital converter card to receive inductosyn or resolver feedback (available on the U500ULTRA and U500ULTRA-PCI only)
SBX-ENC1	Fifth axis encoder channel (daughter board fits on the iSBX site of the U500 board, U500ULTRA and U500ULTRA-PCI only). Additional SBX- options are available
BRK/BPS	Fail-safe brake control logic
PB8	Opto 22 interface board that provides 8 inputs or 8 outputs
PB16	Opto 22 interface board that provides 8 inputs and 8 outputs
PB24	Opto 22 interface board that provides 16 inputs and 8 outputs
SHUNT500	Regulates either (or both) of the two possible bus power supplies of the DR500
BB501	Breakout module to interface with the BA Series amplifiers when not using DR500 chassis
DIOSR/DRC-xx	Input/output extension cable (also required with AC brushless motor operation as Hall effect inputs)

1.5. Safety Procedures and Warnings

The following statements apply wherever the Warning or Danger symbol appears within this manual. Failure to observe these precautions could result in serious injury to those performing the procedures and/or damage to the equipment.



To minimize the possibility of electrical shock and bodily injury, make certain that all of the electrical power switches are in the off position prior to making any electrical connections.



To minimize the possibility of electrical shock and bodily injury when any electrical circuit is in use, ensure that no person comes in contact with the circuitry.



When this controller is installed within a system, mechanical motion will occur. Care must be exercised that all personnel remain clear of any moving parts.



To minimize the possibility of bodily injury, make certain that all electrical power switches are in the off position prior to making any mechanical adjustments.

∇ ∇ ∇

CHAPTER 2: GETTING STARTED

In This Section:

- Introduction 2-1
- Unpacking the UNIDEX 500 System 2-1
- Inspection of the UNIDEX 500 Control Board 2-2

2.1. Introduction

This chapter steps the operator through unpacking the U500, static precautions, and board inspection techniques. The user should read this section before attempting to install the UNIDEX 500 hardware. Hardware installation, hardware configuration, and software installation are discussed in the chapters that follow.

2.2. Unpacking the UNIDEX 500 System

Before unpacking any components, visually inspect the containers of the U500 system for any evidence of shipping damage. If any such damage exists, notify the shipping carrier immediately.

All electronic equipment is wrapped in antistatic material. Make certain that the antistatic material is not damaged during unpacking.



Remove the packing list from the UNIDEX 500 container. Make certain that the items listed on the packing slip are contained within the package. The following items should be found in every UNIDEX 500 system:

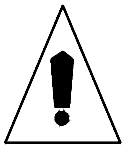
- The UNIDEX 500 Motion Controller and Software User's Manual
- The UNIDEX 500 PC bus-based controller (base, Plus or Ultra)
- Toolkit or MMI software (on double-sided, high-density floppy diskettes or CD)
- UNIDEX 500 packing slip (listing products shipped with the order)

The following list of additional items may be included with the UNIDEX 500 system, depending on the options and accessories that have been specified:

- The DR500 drive chassis (amplifier chassis with power supply)
- The OP500 interface cable (to connect the UNIDEX 500 to the DR500)
- The BB500 or BB501 breakout module (if the DR500 drive chassis is not used)
- Motor connector cables (to connect the motors to the DR500 drive chassis)
- JBV joystick and cable (with digitizing capability)
- Handwheel assembly and cable.

2.3. Inspection of the UNIDEX 500 Control Board

Before touching the UNIDEX 500 control board, be sure to observe the electrostatic discharge precautions listed below.



WARNING

The U500 board is sensitive to static electricity. To greatly reduce the possibility of board damage due to electrostatic discharge, adhere to the following precautions.

1. Do not remove the UNIDEX 500 PC board from the antistatic bag until it is ready to be installed. When removing a card from a system, immediately place the card in an antistatic bag.
2. Make certain that anyone handling the board (or any associated components) is wearing a properly grounded static strap.
3. When handling the UNIDEX 500 control board, hold the card by its edges and the mounting brackets. Avoid touching board components and the edge connectors that plug into the expansion slots.
4. Do not slide the UNIDEX 500 control board over any surface.
5. Avoid plastic, Styrofoam or vinyl in the work area.
6. Static charge buildup may be removed from an object by touching the object to a properly grounded piece of metal.

∇ ∇ ∇

CHAPTER 3: SOFTWARE INSTALLATION AND FUNDAMENTALS

In This Section:

- Introduction 3-1
- Minimum Hardware Requirements and Recommended System Configurations 3-1
- Standard Installation Using Windows NT 4.0 or 95..... 3-2
- Installing the UNIDEX 500 PC Board 3-3
- Software Startup 3-4
- Software Configuration Considerations..... 3-9
- Special Startup Considerations..... 3-11

3.1. Introduction

This chapter explains how to install and run the 32-bit MMI software for the UNIDEX 500 system. The chapter also discusses preliminary considerations prior to enabling an axis such as setting the parameters and tuning the servo loop. Important software configuration information is included.

3.2. Minimum Hardware Requirements and Recommended System Configurations

Minimum hardware requirements and recommended system configurations for the UNIDEX 500 are shown in Table 3-1.

Table 3-1. Minimum Hardware Requirements and Recommendations

Equipment	Minimum	Recommended
Computer (microprocessor)	IBM PC AT and PS/2 80486 or 100% compatibles	Pentium or higher
Computer Memory	8 MB of memory (conventional & extended)	16 MB of memory (conventional & extended)
Graphics Display	EGA/VGA	EGA/VGA
Hard Disk Space	18 MB	20 MB or more
Mouse	Any mouse supported by the computer	Any mouse supported by the computer
Disk Drives	3 ½" double-sided, high density	4X CDROM
Operating Systems	Windows 95 or higher Windows NT 4.0 or higher *	Windows 95 or higher Windows NT 4.0 or higher *

* The U500 MMI is not compatible with versions previous to Windows NT Version 4.0.

3.3. Standard Installation Using Windows NT 4.0 or 95

To install the software, the operator must follow the steps listed below.

1. Start Microsoft Windows NT 4.0, 95, or 98.
2. Place the installation CD in your CD-ROM drive.
3. Select Run from the Start menu.
4. If MMI was purchased, run the setup program (setup.exe) from the Winnt_95 subdirectory on the CD.
5. If MMI was not purchased, run the setup program (setup.exe) from the root directory. This will install the Toolkit software.
6. Select ISA or PCI installation when prompted. (To determine which type of card you have, hold the card with the front of the card facing you [the side with the larger components] and with the connector that plugs into the PC pointing down. The ISA card has the silver 100 pin connector P1 on the right side of the board. The PCI card has the same connector on the left side of the board).

   The program group called U500 MMI is created. Refer to Figure 3-1.
7. Power down the PC and insert the U500 card.



The Software disks may also contain additional files such as customized parameter (.PRM) files, configuration (.CFG) files, and calibration (.CAL) files. These files (if they are included) use the six digit customer order number as the file name (for example, 123456.PRM, 123456.CFG, and 123456.CAL). The installation program automatically creates a default project (named 123456.PRJ in this example) if these files are found on the Software disk. In addition, these files are copied to the root directory (C:\U500, for example) as well as the \DEFAULTS subdirectory. (The \Defaults subdirectory is not created in the 32-bit software.)



The U500PCI card requires software version 5.11 or higher and must use the firmware file “U500PCI.JWP”.

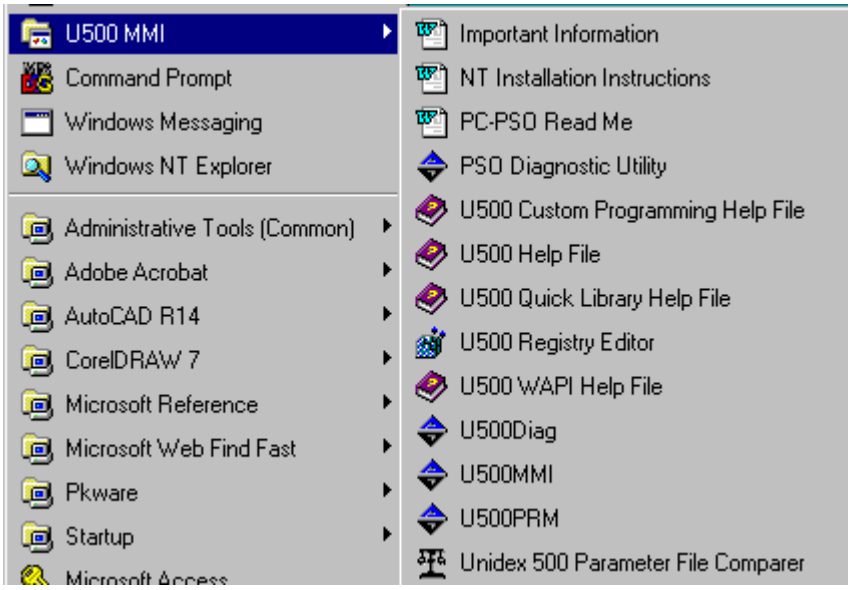


Figure 3-1. Software Menu Available after Completed Installation

3.4. Installing the UNIDEX 500 PC Board

The UNIDEX 500 control board is a full-sized AT/PCI card that is installed into any of the PC's unused expansion slots.

The UNIDEX 500 PC control board may not fit in some smaller models of PCs.



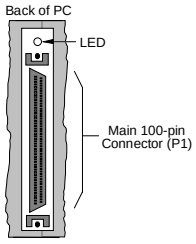
The procedure for installation of the UNIDEX 500 PC board is outlined in the steps that follow.

1. Turn OFF the power to the computer system unit and unplug the unit's power cord from the power source.

There is a risk of electric shock. Make certain that the computer system's power switch is in the OFF position and the power cord is disconnected before opening the computer's cabinet.



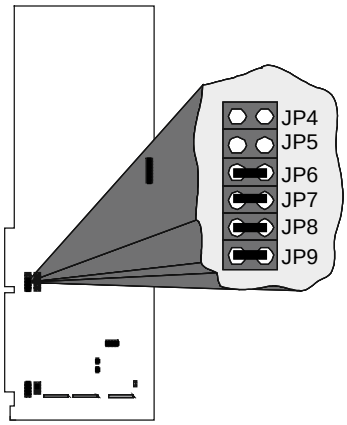
2. Open the computer cabinet. (Refer to the PC's User Manual for directions on opening the cabinet.)



3. Select an unused full-sized expansion slot on the computer motherboard.
4. Locate the bracket of the selected expansion slot. Remove the screw and pull the bracket out of the expansion slot.
5. Observing anti-static safeguards, line up the UNIDEX 500 PC board with the expansion slot and guide rails. Lower the board into the slot until each of its edge connectors rests on an expansion slot receptacle. Using evenly distributed pressure, push the board straight down until it is fully inserted into the expansion slot.
6. Secure the board to the chassis by reinstalling the bracket screw that was removed in step 4.

3.5. Software Startup

3.5.1. U500 ISA Base Address Jumpers (JP4, JP5, JP6, JP7, JP8, and JP9)



Input/Output (I/O) base addresses for the UNIDEX 500 are assigned in hexadecimal address ranges. The UNIDEX 500 control board occupies 16 consecutive memory locations in the input/output (I/O) channel memory of the PC. The UNIDEX 500 control board is factory configured for address 0x300-0x30F. The UNIDEX 500 interface software is also set to this default address. If the UNIDEX 500 control board does not initialize properly or exhibits sporadic operation, there may be another board in the system computer that is set to the same address. Use device manager that comes with the operating system to analyze which addresses are used, then try another UNIDEX 500 address (remember to select a new address in the Startup software).

Follow these steps to view the map addresses in Windows NT:

1. Select Administrative Tools from the Programs Menu
2. Select the Windows NT Diagnostics Program
3. Select the Resources Tab
4. Select the I/O Port Button to view the map addresses

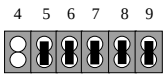
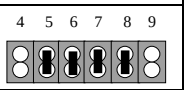
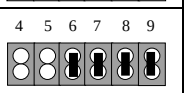
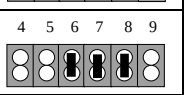
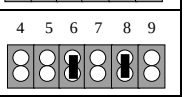
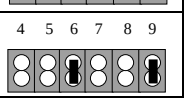
To view the map addresses in Windows 95/98:

1. Select ‘System’ from the Control Panel
2. Select Device Manager
3. Highlight ‘Computer’ and select the Properties button
4. Select the I/O button to view the map addresses

The address of a UNIDEX 500 board is set from jumpers JP4-JP9. These jumpers are located near the P2 connector of the control board. The pins of each jumper in the series (JP4 through JP9) are either connected (“IN”) or disconnected (“OUT”) to create a unique base address. The combinations of base address jumper settings are shown in Table 3-2.

Each UNIDEX 500 control board must have a distinct address in the same PC.

Table 3-2. Base Address Jumper Settings

PC I/O Base Address	JP4	JP5	JP6	JP7	JP8	JP9	Settings
200 - 20F	OUT	IN	IN	IN	IN	IN	
210 - 21F	OUT	IN	IN	IN	IN	OUT	
300 - 30F (default)	OUT	OUT	IN	IN	IN	IN	
310 - 31F	OUT	OUT	IN	IN	IN	OUT	
350 - 35F	OUT	OUT	IN	OUT	IN	OUT	
360 - 36F	OUT	OUT	IN	OUT	OUT	IN	

3.5.2. Configuring the Software to Match the Hardware

The U500 software must be configured to match the hardware settings and bus type. Open the "U500 Registry Editor" program located in the U500 MMI group. See Figure 3-2.

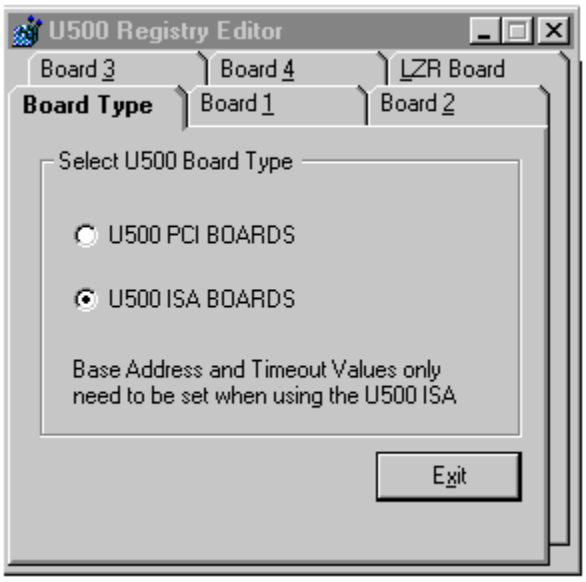


Figure 3-2. U500 Registry Editor

Select the correct board type, ISA or PCI. If this selection is changed, reboot the PC.

3.5.3. Configuration of the Software for the ISA Board

The UNIDEX 500 software must be configured to the same base address as the jumper on the U500 board (jumpers JP4-JP9). Use the "U500 Registry Editor" program located in the U500 MMI group. See Figure 3-3.

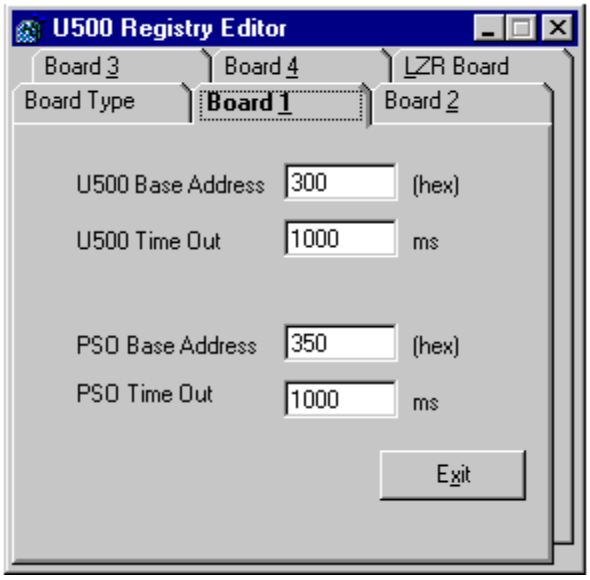


Figure 3-3. Board 1 (U500 Registry Editor)

Set the U500 Base Address for Board 1 to the appropriate hex address (default = 300). If you are using a PSO laser firing card or multiple U500 cards, set these addresses appropriately. If you do not have any additional Aerotech cards, put a 0 in all other base address text boxes. The TimeOut value should be set to a default of 1000. If any changes have been made to these registry values, the PC must be re-booted.

3.5.4. Configuring the Software for the PCI Board

There are no address settings for the PCI board. The hardware follows the "Plug and Play" standard. Windows NT will automatically configure the software to match the U500 PCI board. If you are using Win 95 or Win 98, after installing the UNIDEX 500 PCI card, the operating system will display a message "New Hardware Found". Take the following steps to properly install the device driver for the UNIDEX 500 PCI card:

Win 95:

1. The "Update Device Driver Wizard" will appear with the message "Windows was unable to locate a driver for this device". Click the "Other locations" button.
2. Insert the U500 Software Installation CD and enter "drive:\win95" (i.e. e:\win95) in the text box and select OK.

3. Select Finish.

4. A message will appear, “Insert disk, unable to find u500pci.vxd”. Enter “drive:\win95” in the text box and select OK.

5. Select Finish, and restart the computer.

In the device properties box, the U500PCI device will display a “system failure” message. Ignore this, the driver should be functioning properly.

Win 98:

1. The “Add New Hardware Wizard” will appear, click the Next button.

2. “What do you want Windows to do” message will be displayed, select “Search for the best driver for your device”.

3. Select the box “Specify a location” for where Windows will search. Insert the U500 Software Installation CD and enter “drive:\win98” (i.e. d:\win95). Select OK.

4. Select Finish, and restart the computer.

PCI and ISA versions of the UNIDEX 500 Controller cannot be mixed.



3.5.5. Initializing the U500 Board

Once the device driver is properly configured, start the UNIDEX 500 MMI or Toolkit software. If a message stating “Could not open board or driver not running”occurs, please refer back to the “Configuring the Software” section or the Troubleshooting section. Before resetting the U500 card, make sure the project file is set up correctly. Select “Project” from the “Edit” menu of the main MMI/Toolkit screen. The window in Figure 3-4 will be displayed.

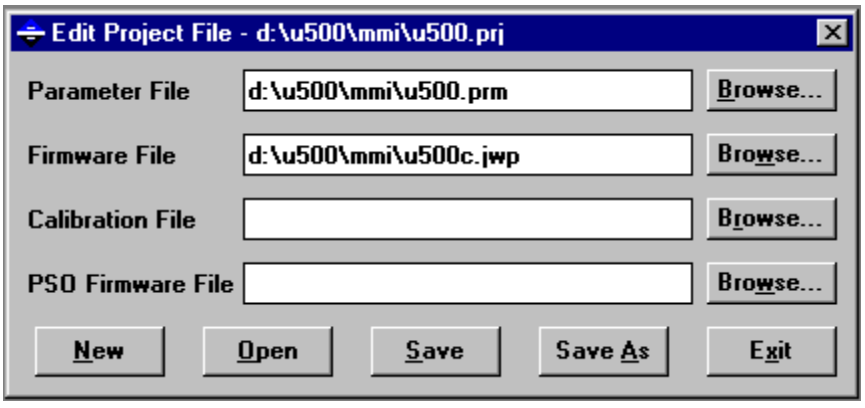


Figure 3-4. The Edit Project File Popup

The parameter file should default to U500.PRM. If your system has been configured at Aerotech, the parameter file will be the customer order number, i.e. 101234.PRM. If you are using an ISA card, the correct firmware file to be used is "U500C.JWP". If you are using a PCI card, the correct firmware file is "U500PCI.JWP". Make the correct changes and save the project file. Next hit the F9-Reset key. This downloads the software to the U500 board and the card should be ready for use. Following a Reset, the software should display the Axis Position window, as shown in Figure 3-5.

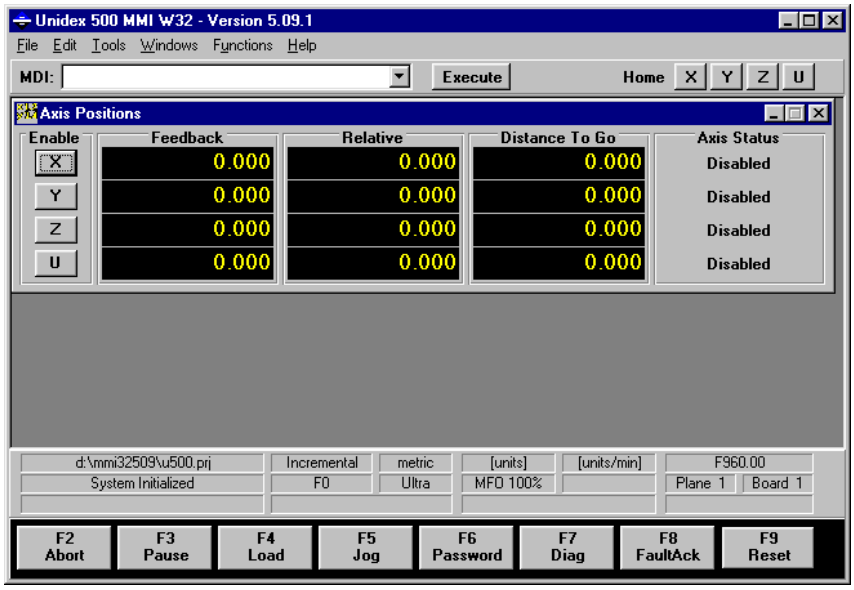
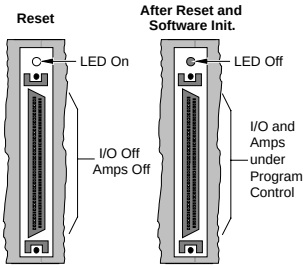


Figure 3-5. The Axis Position Window



U500 PCI:

The green LED, when illuminated, indicates that the controller is running. When off, the controller is in the reset state.

If the LED does not come ON or if it stays ON following software initialization, the device driver is not running or the incorrect firmware file has been specified. The correct firmware file is "U500PCI.JWP". Also insure that the board is seated properly in the PC and the installation procedure was followed correctly.

U500 ISA:

When the PC with the U500 is first powered-up, the U500's LED (visible from the rear of the system) should be ON. This will disable the amplifiers and set the output bus to the high impedance state. The LED will stay ON until the system is initialized, then the LED will turn OFF and remain OFF. During subsequent system software resets, the LED should come ON briefly and then turn OFF. The red LED, when lit, means that the U500 ISA is in reset (stop state).

If the LED flashes continuously, please make sure the correct firmware for the specific revision of the U500 board is being used, otherwise contact Aerotech's Service Department.

If the LED does not come ON or if it stays ON following software initialization, the I/O address is probably not set correctly. Refer to the Troubleshooting section of this manual for help.

3.6. Software Configuration Considerations

Prior to any motion control or programming, the UNIDEX 500 software must be properly configured. Although the software has many parameters, only a small group must be configured to get started. This section highlights only those parameters that are most important in order to form a basic operating foundation for the control system. Eventually, fine-tuning must be done using all of the essential parameters listed later in this manual.

For a detailed discussion of the parameters listed in this section (and others) refer to the individual parameter listings found in Chapter 5: Parameters.

3.6.1. Configuring Key Parameters

This section highlights the key parameters that are most important in forming a basic operating foundation for the control system. Fine-tuning must be done using all of the essential parameters listed in Chapter 5. This foundation is intended to provide a starting point for both the system and operator.

Table 3-3 lists the key parameters discussed above. This table includes a subset of parameters from the following parameter groups: position tracking, basic and advanced motion, motor feedback, servo loops, homing, and limits, and faults/traps/masks. Parameter numbers, descriptions, and default values are listed in the table.

Axis-related parameters begin with the letter "x" (e.g., x17). This is a convenient way of representing an axis parameter for each of the four axes (1-4). The value of the *Top feedrate* parameter (x17) is found in parameters 117, 217, 317 and 417 for axes 1-4, respectively.



Table 3-3. Key UNIDEX 500 Configuration Parameters

Parameter Group	Parameter Number	Description	Default Value(s)
1. Advanced Motion	007	Axis 1 gantry (y/none), slave (2,3,4)	None
	008	Axis 2 gantry (y/none), slave (1,3,4)	None
	009	Axis 3 gantry (y/none), slave (1,2,4)	None
	010	Axis 4 gantry (y/none), slave (1,2,3)	None
0. Position Tracking	011, 012, 013, 014	Axis n rollover machine steps (n = 1, 2, 3, or 4)	0
	020, 038, 056, 074	Plane n Metric System (y/n) (n = 1, 2, 3, or 4)	Yes
	029, 047, 065, 083	Metric mode number of decimal digits (1-8) (planes 1-4)	3
	030, 048, 066, 084	English mode number of decimal digits (1-8) (planes 1-4)	4
7. Other	x00	Metric conversion factor	1.0
	x01	English conversion factor	1.0
3. Homing/Limits	x02	Home direction is CCW (y/n)	Yes
	x03	Home switch normal open (y/n)	Yes
	x09	Limit switch normal open (y/n)	Yes
7. Other	x11	Positive (+) Move is CW (y/n)	Yes
4. Traps	x17	Top feedrate (machine steps/ms)	440.000000
	x18	Max velocity error (0-8,388,607)	1000
	x19	Max position error (0-8,388,607)	4000
	x20	Max integral error (0-8,388,607)	655,360
6. Servo Loops	x25	Kpos (0-8,388,607)	50
	x26	Ki (0-8,388,607)	5000
	x27	Kp (0-8388607)	100,000
	x28	Vff (0-8388607)	256
	x29	Aff (0-8388607)	0
5. Motor Feedback	x38	Primary/Position feedback channel	1, 2, 3, or 4
	x39	Secondary/Velocity feedback channel	0
	x42	Drive (motor) type (0-DC Brush, 1-AC Brushless, 2/3-Stepper)	0
	x44	Encoder Feedback (steps/rev/(*4))	4000

The values for the parameters listed in Table 3-3 are changed from the Parameters... option of the Edit menu. When this option is selected, the Edit Parameters popup window is displayed (see Figure 3-6). Parameters are organized into logical groups that are displayed on "tabs" in the Edit Parameters popup window. The General and Axis tabs collectively contain all of the UNIDEX 500 parameters. The other tabs (numbers 0-8) contain subsets of these parameters organized into functional groups (e.g., basic motion, faults, position tracking, etc.).

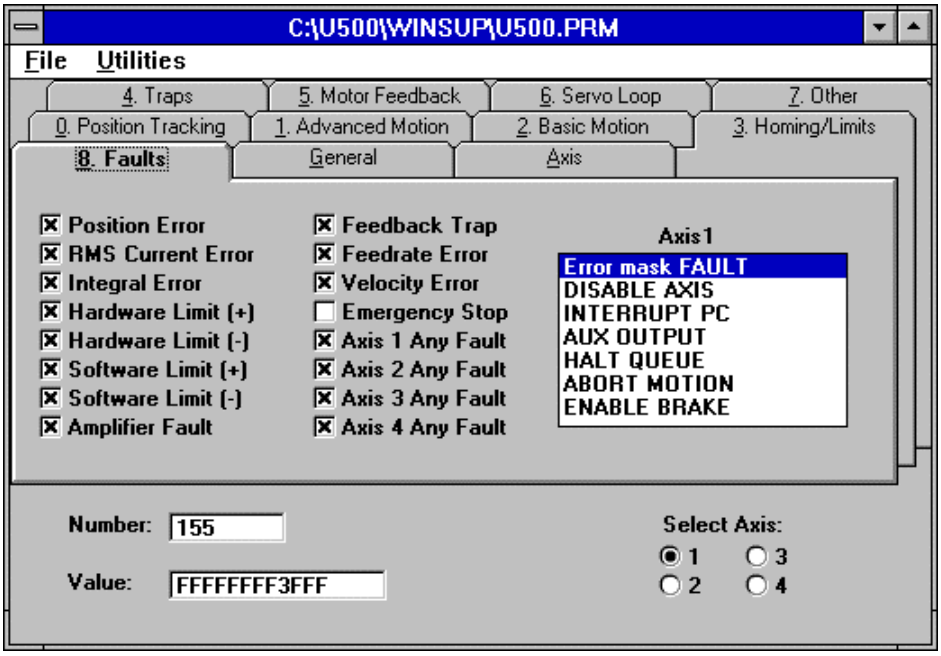


Figure 3-6. Parameter Window

For a detailed description of all U500 parameters, refer to Chapter 5: Parameters. For illustrations of all software screens and menu options, refer to Chapter 4: The Software Interface.

3.7. Special Startup Considerations

It is recommended that several functions be verified prior to enabling an axis for motion. To facilitate this verification process, a diagnostics window is provided in the software package. This diagnostics screen (shown in Figure 3-7) displays hardware (limits, I/O, etc.) and servo-related information (traps, machine position, etc.) dynamically (in real time). It is useful for system setup and debug purposes. The diagnostics window is accessed from the Diagnostics option of the Tools menu. For more information about accessing and using the diagnostics window, refer to Chapter 4: The Software Interface.

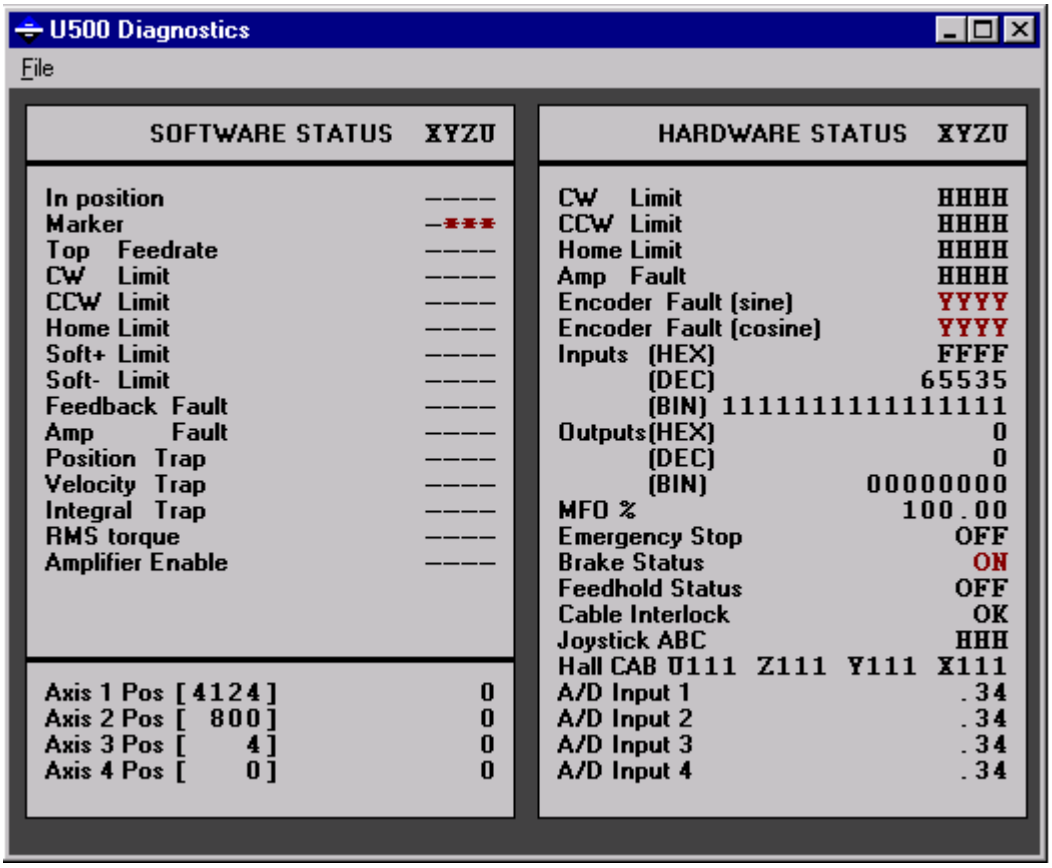


Figure 3-7. The Diagnostics Popup Window

When using the U500 PCI, a second diagnostic window is available to determine the status of axis 5-8, the additional I/O and the additional A/D inputs. This second Diagnostic window (Figure 3-8) is accessible from the File Menu of the Diagnostic window shown above.

The UNIDEX 500 software also contains a tracking display window that shows position data in real time. The position is displayed in programmable units that the operator may define (e.g., mm, in, etc.). The card must be initialized to display the Axis Positions window. The real-time position display window is illustrated in Figure 3-9.

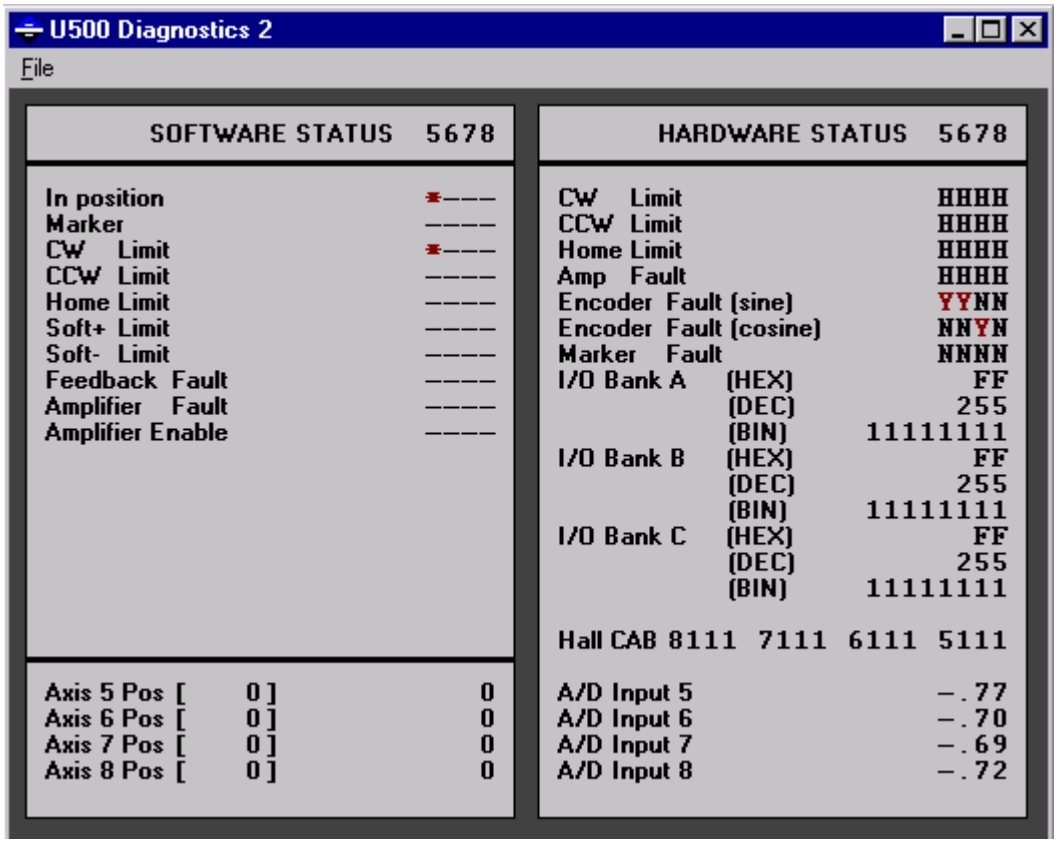


Figure 3-8. The Diagnostics 2 Popup Window

Make certain that all system traps and faults are enabled prior to initiating any axis movement. For more information, refer to the faults, traps, and mask parameters in Chapter 5.



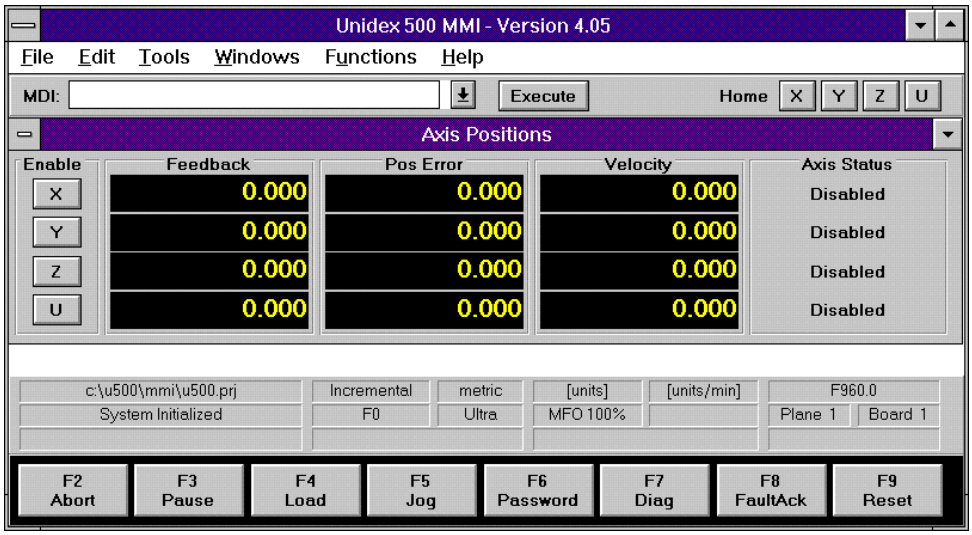


Figure 3-9. The Axis Position Display (Choose Windows ➔ Axis Display ➔ Show)

Figure 3-10 is a flowchart that summarizes the installation process from installing the software through preliminary servo loop setup.

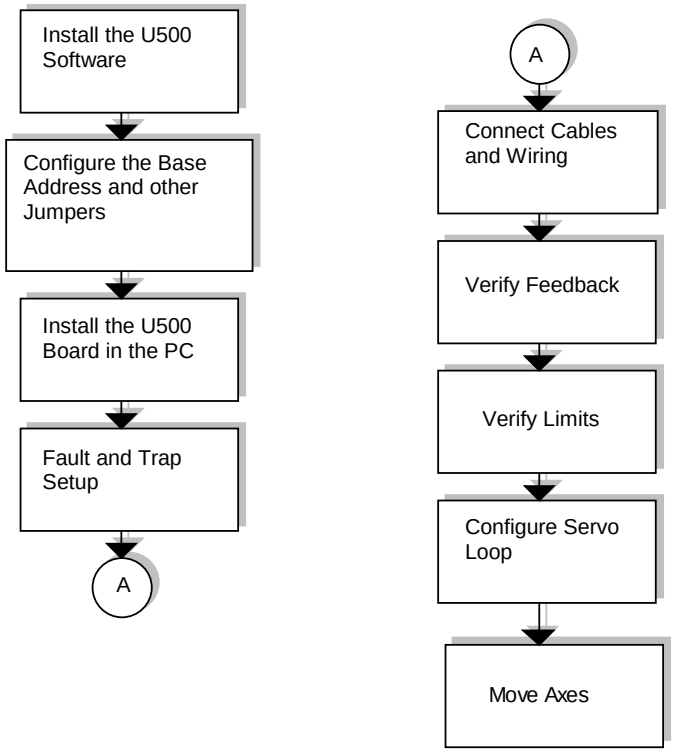


Figure 3-10. Flowchart Summary of the Installation/Configuration Process

3.7.1. Feedback Verification

Before enabling any axis, it is important to verify the feedback from the motors. The U500 card must be initialized. Display the diagnostic screen and refer to the Axis Position display in the lower left corner. The tracking display should be stable while the axis is stationary, although slight movement with high-resolution systems is normal.

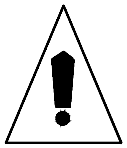
Feedback phasing may be verified by manually turning the motor. Clockwise/counter-clockwise rotation should produce an increasing/decreasing display in the diagnostic window. If not, feedback phasing is incorrect.

Motor direction (clockwise or counter-clockwise) is always referenced by "looking into" the shaft end of the motor. For linear motors, motion away from the motor wires should result in increasing feedback.



If the feedback does not perform as noted above, refer to the introduction to Motor and Feedback Configuration (in Chapter 6) and Chapter 13 (Troubleshooting).

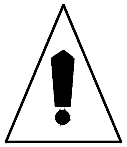
Make certain that the UNIDEX 500 is appropriately configured for the type of motor being driven.



3.7.2. Limit Verification

Limit verification is extremely important in the startup of the UNIDEX 500 system. Improperly configured limits can cause damage to system components and can pose safety hazards to operators and others. Limit verification requires the operator to disable each axis, manually engage each limit, and then check the state of that limit input using the diagnostic window (refer to Figure 3-7). The diagnostic window will display the state of limit inputs as either an "H" (for "high") or an "L" (for "low"). Normally closed limit switches go from low to high when activated. Normally open limit switches go from high to low when activated. If no change is observed, the limit system is faulty. Refer to Chapter 5 of this manual for more information about limit parameters.

To prevent the possibility of personal injury or possible damage to the equipment, do not enable the axes until the limits are working properly.



3.7.3. Preliminary Servo Loop Setup

In the most general sense, control loops are systems that create output signals based on (1) input signals and (2) a series of servo gains that define the output over a variety of input criteria. These gains must be individually tailored to every unique application. The process of manipulating these servo gains to provide the most desirable response is called *servo tuning*. In UNIDEX 500 systems, servo loops are tuned using servo gain parameters.

In the UNIDEX 500 system, there are five tuning parameters associated with each axis. Each set of servo tuning parameters must be properly configured before the associated axis can be enabled. The five servo loop tuning parameters are listed in Table 3-4.

Table 3-4. Servo Loop Tuning Parameters

Abbreviation	Description
Kpos (0-8,388,607)	Position Loop Gain
Ki (0-8,388,607)	Integral Gain
Kp (0-8,388,607)	Proportional Gain
Vff (0-8,388,607)	Velocity Feed Forward
Aff (0-8,388,607)	Acceleration Feed Forward



Servo loop tuning should be done with the motors fully loaded. Inertia, momentum, gravity, friction, and other forces effect the response of the system.

Preliminary servo loop setup consists of enabling the axis and tuning the servo loop for the desired performance. Tuning screens are illustrated in Chapter 4: The Software Interface. Information on UNIDEX 500 parameters can be found in Chapter 5: Parameters. The loop tuning process is explained in detail in Chapter 6: Tuning Servo Loops. Refer to the appropriate sections for more information.



If Aerotech stages are ordered with your UNIDEX 500 controller, the servo loop gain parameters will already be set from the factory.

∇ ∇ ∇

CHAPTER 4: THE SOFTWARE INTERFACE

In This Section:

- Introduction....4-1
- The File Menu....4-2
- The Edit Menu....4-4
- The Tools Menu....4-15
- The Windows Menu....4-35
- The Functions Menu....4-48
- The Help Menu....4-49

4.1. Introduction

This chapter explains the options that are available through the MMI software interfaces for the UNIDEX 500 motion controller. The U500 comes with Toolkit software. Additional functionality is provided by the MMI software interface (if purchased). Through the software interface, the operator can configure the U500 system, perform manual motion control, and create/edit/run motion control programs.

The Toolkit software is included with the U500. Additional functionality is provided by the MMI only if purchased by the customer.

This chapter provides a complete list of menu items and commands that are provided through the U500 software interface. Sample screens are illustrated to provide an understanding of the menu structure (Figure 4-1) and software interface.

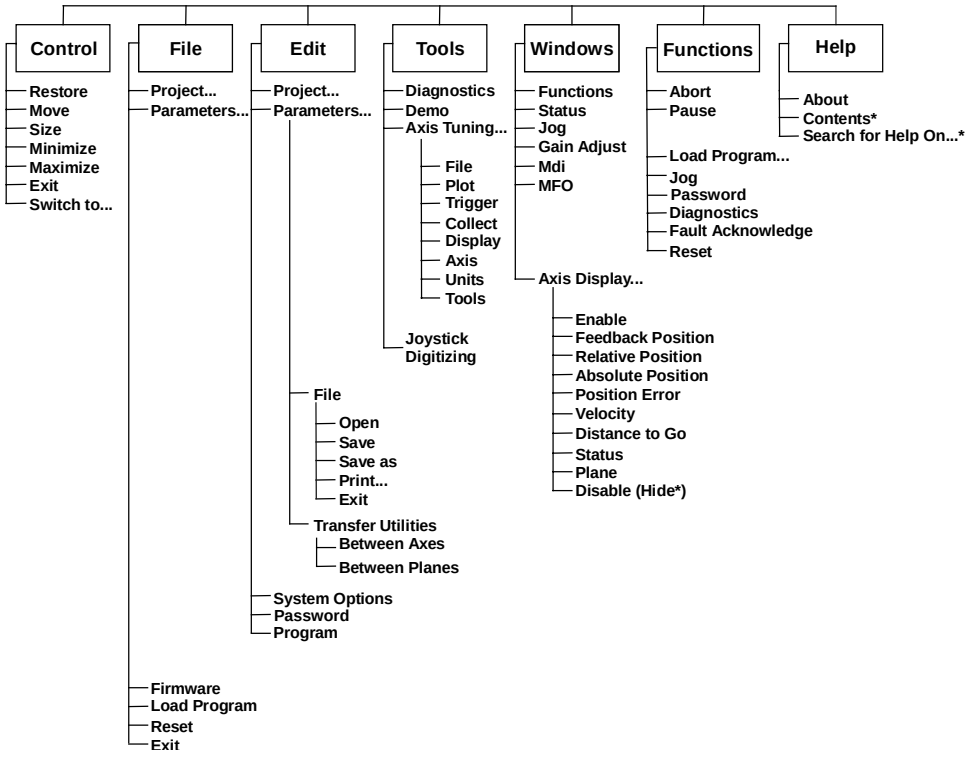


Figure 4-1. U500 Software Menu Structure

4.2. The File Menu

The File menu is a pull-down menu of the U500 software that contains the commands listed in Table 4-1. The underscored letters in the table indicate "short cut keys." Once the pull-down menu has been activated, hitting an underscored key activates the command.

Table 4-1. File Menu Commands

Command	Description
Project...	Opens an existing project (.PRJ) file
Parameters...	Opens an existing parameters (.PRM) file
Firmware...	Opens an existing firmware (.JWP) file
Load Program...	Loads an existing parts program (.PRG) file
Reset	Loads configuration (.CFG), firmware (.JWP) and parameter (.PRM) files and then resets software
Exit	Quits the MMI program

Parameter settings and firmware are all stored in files. These files use standard DOS naming conventions and have the extensions .PRM and .JWP, respectively. It is convenient to group a parameter file, a firmware file, a calibration file, and a PSO firmware file into a single unit called a *project*. The manipulation of a project (a single .PRJ file containing the names of the .PRM, .JWP, and possibly a .CAL and .FRM file) is easier and more efficient than manipulating its three associated components. The U500 software supports up to four project files, one for each installed controller.

4.2.1. The Project... Option of the File Menu

The Project... option of the File menu is used to open (and to prepare to load) an existing project. When this option is selected, the operator selects an existing project file name (with extension .PRJ). After a project name is chosen, the project is loaded into the system's memory and becomes the current project. This selection overrides the project file selection in the system options window.

Opening a project using the Project... option of the File menu does not automatically reset the UNIDEX 500. The U500 must be initialized (using the F9 Reset button or the Functions menu) for the new configuration and parameter files to take effect.



4.2.2. The Parameters... Option of the File Menu

The Parameters... option of the File menu is used to load an existing parameter file into memory. When this option is selected, the operator selects an existing parameter file name (with extension .PRM). This selection overrides the project file's parameter file setting.

Opening a parameter file using the Parameters... option of the File menu does not automatically reset the UNIDEX 500. The U500 must be initialized (using the F9 RESET button or the Functions menu) for the new parameter file to take effect.



4.2.3. The Firmware... Option of the File Menu

The Firmware... option of the File menu is used to load the UNIDEX 500 firmware file into memory. When this option is selected, the operator selects the firmware file name (with extension .JWP). This selection overrides the project file firmware setting.

Opening a firmware file using the Firmware... option of the File menu does not automatically reset the UNIDEX 500. The U500 must be initialized (using the F9 Reset button or the Functions menu) for the new firmware file to take effect.



4.2.4. The Load Program... Option of the File Menu

The Load Program... option of the File menu is used to load an existing motion program into memory. When this option is selected, the operator selects the program file name (with extension .PRG). After the program file name is chosen, the program run screen is displayed.

4.2.5. The Reset Option of the File Menu

The Reset option of the File menu is used to initialize the UNIDEX 500 system. Selecting this option has the same effect as selecting the F9 Reset soft key at the bottom of the main screen, selecting the Reset option from the Functions menu, or pressing the F9 function key on the keyboard.

Initialization is a process in which the current parameter (.PRM) file is sent down into the memory of the U500 board from the PC. In addition, the actual control program (the *firmware* file .JWP) is also sent down to the U500 board. The last step of the initialization process is the restarting of the U500 firmware program.



Some menu commands and display features are not available if the U500 control board is not reset (i.e., initialized). An attempt to invoke such commands and/or features will cause a "System Not Initialized" popup box to be displayed.

4.2.6. The Exit Option of the File Menu

The Exit option of the File menu is used to stop the software and close the main window.



If a motion control program is executing when the Exit option is selected, buffered commands will be executed by the UNIDEX 500 until the queue is empty.

4.3. The Edit Menu

The Edit menu is a pull-down menu of the software that contains the commands listed in Table 4-2.

Table 4-2. Edit Menu Commands

Command	Description
Project...	Create a new project, edit the file names of an existing project, open or save a project
Parameters...	Edit parameter values
System Options	Associates a project name to the U500 board, enables/disables the Auto Initialize feature, and accepts the name of a program (.PRG) file to be executed automatically upon initialization
Password	Sets up passwords and security levels for the software.
Program...	Motion control program editor

4.3.1. The Project... Option of the Edit Menu

The Project... option of the Edit menu is used to create a new project, edit the file names of an existing project, or open or save a project file. When this option is selected, the "Edit Project File" popup is displayed (refer to Figure 4-2). The components of the "Edit Project File" popup are listed and described in Table 4-3.

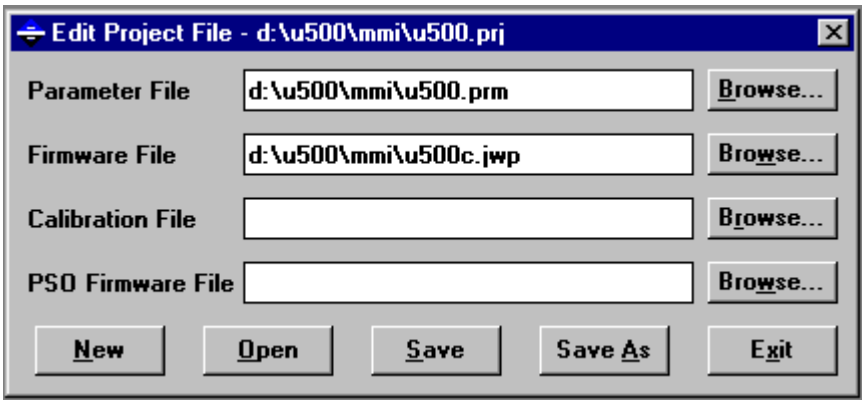


Figure 4-2. The Edit Project File Popup

Table 4-3. Components of the "Edit Project File" Popup

Component	Function
Parameter File	Specifies the full path and file name of the parameter (.PRM) file for the project
Firmware File	Specifies the full path and file name of the firmware (.JWP) file for the project. See note below.
Calibration File	Specifies path and file name of the calibration file (.CAL). The .CAL file contains stage positioning correction data. If there is no .CAL file, leave this box blank. See Appendix G for more information on the .CAL file.
PSO Firmware File (U500 ISA Only)	Specifies path and file name of the PSO firmware file (.FRM). This file must be specified if a PSO laser-firing card is being used. The default file is C:\U500\MMI\PCPSO.FRM.

There are 3 different firmware files: U500.JWP, U500C.JWP, and U500PCI.JWP. The original U500.JWP file is for use with Rev. – of the U500 ISA board. This revision of the board was no longer manufactured after 1997. This revision was replaced with Rev. C and requires the U500C.JWP firmware file. The easiest way to identify the board revision level is to look at the type of jumpers on the board. Rev – cards have metal, gold plated jumpers. Rev C cards have blue plastic jumpers. If a PCI version of the card is being used, the firmware file must be specified to be U500PCI.JWP.

4.3.2. The Parameters... Option of the Edit Menu

The Parameters... option of the Edit menu is used to edit parameter values for the UNIDEX 500 system. When this option is selected, the edit parameters screen is displayed. This screen contains 11 parameter tabs (that categorize parameters by their function). Each parameter is available on two different tabs (i.e., parameters on the Position Tracking tab are on the General tab). This screen is displayed in Figure 4-3. The components of this screen are discussed in the sections that follow.

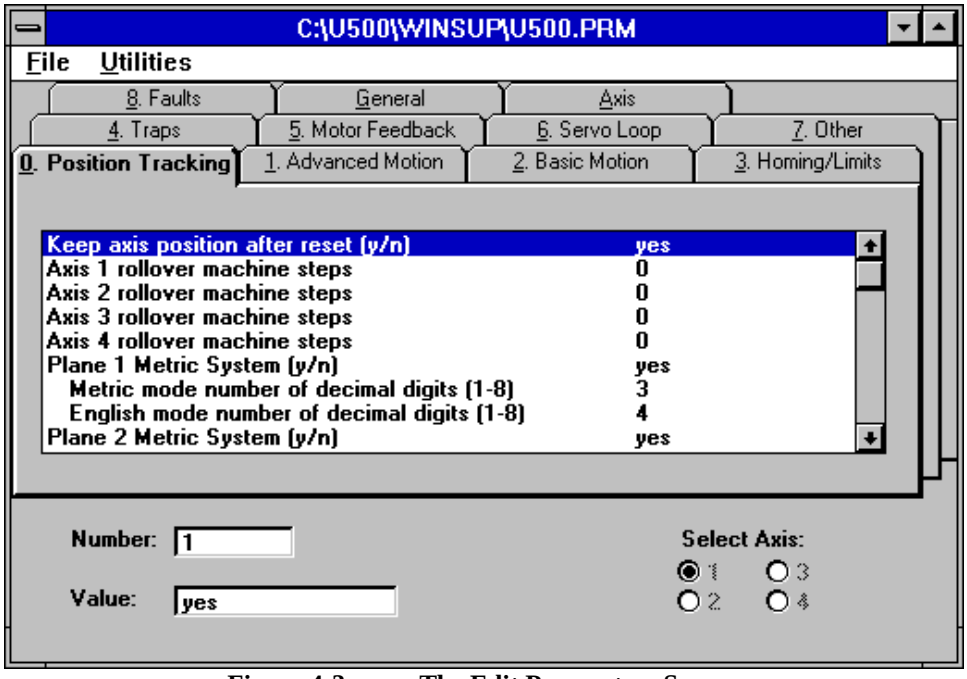


Figure 4-3. The Edit Parameters Screen

4.3.2.1. Edit Parameters: The File Submenu

The File submenu of the "Edit Parameters" screen contains options to open an existing parameter (.PRM) file, save changes to the current .PRM file, save to a new .PRM file name, and print parameters.

The Print... ➢ option provides two types of print options: the "By Tab Group" option and the "By Parameter Number" option. The "By Tab Group" option prints a 6-page report of the parameter numbers, parameter names, and current values of the numbered parameters tabs (i.e., tabs 0: Position Tracking through 8: Faults). The "By Parameter Number" option prints the same information, only in a different format (a sequential list of General parameters followed by a sequential list of Axis parameters).

4.3.2.2. Edit Parameters: The Number, Value, and Axis Fields

The "Number:", "Value:" and "Select Axis:" fields of the edit parameter screen are used to locate parameter names (based on parameter numbers), select axes, and view/change parameter values. These three screen components are located at the bottom of the Edit Parameters Screen. Refer to Figure 4-3.

The Number: field displays the parameter number of the parameter that is currently highlighted in the parameter tab. When a parameter is selected from a parameter tab, its parameter number is displayed in this field. This field can also be used to display the parameter when only a parameter number is known. This is accomplished by placing the parameter number in the Number: field and pressing the <Enter> key on the keyboard. If a valid parameter number is entered, the associated tab will become the focus, and the corresponding parameter will be highlighted.

The Value: field contains the value of the selected parameter. To change the value of a selected parameter, the operator must place the cursor in the Value: field. A value can then be entered using the keyboard. When the value in this field is the desired value, the operator must press the <Enter> key to change the parameter value. Entering “d” in the value field box can produce the default value of the parameter.

The Select Axis radio buttons are used to specify the desired axis parameter to be viewed/edited. Only one axis radio button may be selected at a given time.

4.3.3. System Options

The MMI interface has an expanded System Options screen. To access this screen, select the System Options item from the Edit menu. The screen is shown in Figure 4-4.

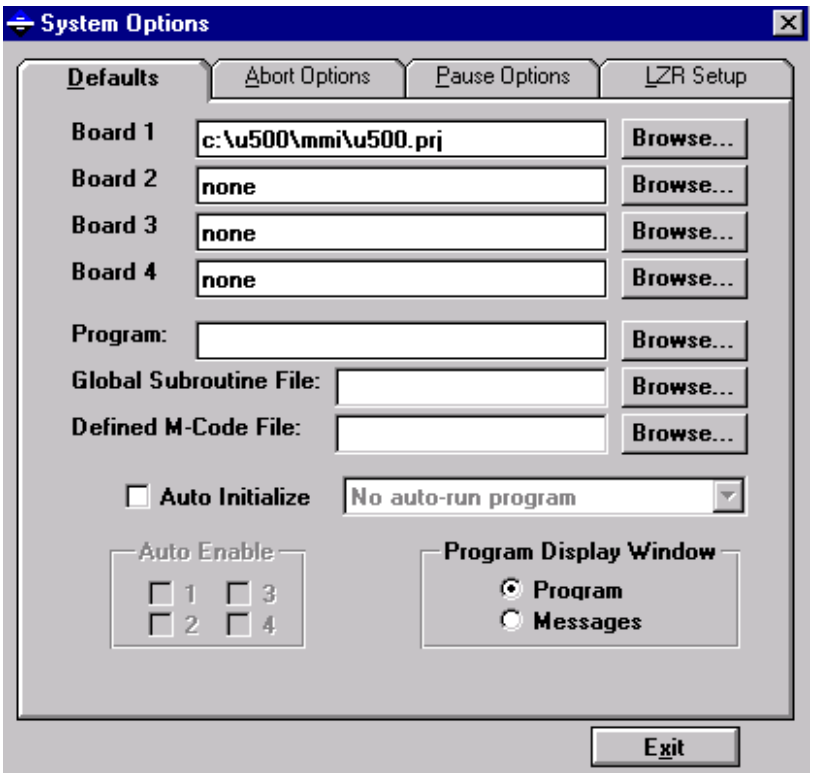


Figure 4-4. The Defaults Tab of the MMI System Options Screen

The System Options screen opens at the “Defaults” index tab. The four text boxes labeled “Board...” are available to select the default project (.PRJ) for each UNIDEX 500 board in the system. The “Program:” text box defines a default program (.PRG) for the system. The “Auto Initialize” check box, when checked, will initialize all system U500 boards according to the default projects when the software is first started.

The “Auto Enable” check boxes will enable the selected axes if Auto Initialize is selected. If a program is specified the auto-run program drop box is enabled. It has 3 choices: no auto-run program, auto-run program, or silent auto-run program. Auto-run program brings up the program execution window and runs the program in the normal execution mode.

Silent auto-run program does not display the program execution window. The program will execute in the background. The “Program Display Window” option radio buttons select whether the program code or the messages display are visible in the program execution window. If the program to be run contains more than 32000 lines, the message button must be selected.

The “Global Subroutine” text box is available to specify a file that contains subroutines that can be accessed from any UNIDEX 500 program that is being executed. Refer to the “Programming commands” section of this manual for the proper syntax of subroutines. An example of a global subroutine follows.

In global subroutine file:

```
:LABEL1
ME DI "HELLO"
RETURN
```

In U500 program:

```
EN X Y
G1 X10
SU :LABEL1  ;CALLS GLOBAL SUBROUTINE :LABEL1
EXIT
```

The Global subroutine file can also be used to list subroutines that will re-define the actions performed by certain MMI function keys and buttons (see Table 4-4). These subroutines will not affect any commands that are executed in a parts program or entered from the MDI line.

Table 4-4. Pre-Defined Global Subroutine Labels

MMI Function Key / Button	Subroutine Label(s)
ABORT	:_ABORT
ENABLE X, Y, Z, U	:_ENABLE1, :_ENABLE2, :_ENABLE3, :_ENABLE4 :_DISABLE1, :_ DISABLE2, :_ DISABLE3, :_ DISABLE4
FAULTACK	:_FAULTACK
HOME X, Y, Z, U	:_HOME1, :_HOME2, :_HOME3, :_HOME4
PAUSE	:_PAUSE_ON, :PAUSE_OFF

If these subroutines are used in the global subroutine file, the code in the subroutine will replace the normal action performed by that specific button. This subroutine can be useful in either adding functionality to a button, or by preventing that button from performing any function at all. If these subroutines are commented or not included in the global subroutine file, the MMI button will perform its usual function.

Refer to following example global subroutines:

```
:_ABORT     ; ABORT subroutine
abort
me di "Motion Aborted"
return

:_FAULTACK  ;  FAULT  ACKNOWLEDGE  SUBROUTINE  –
abort
faultack
me di "Fault Cleared"
return

:_HOME1     ; HOME AXIS 1 SUBROUTINE –
ou 1        ; set an output
enable x    ; enable x
home x      ; home x
return

:_HOME2     ; HOME AXIS 2 SUBROUTINE –
ou 2        ; set an output
me di "Home Axis 2"
enable y
ref y       ; perform a marker search on the y
            ; axis
return

:_HOME3     ; HOME AXIS 3 SUBROUTINE – no action
return

:_HOME4     ; HOME AXIS 4 SUBROUTINE – no action
return

; The following “enable” subroutines
; will execute when the axis is disabled, and
; the MMI Axis enable button is activated

:_ENABLE1   ; ENABLE AXIS 1 SUBROUTINE –
enable x    ; enable the x axis
ou 8        ; set output bits
return

:_ENABLE2   ; ENABLE AXIS 2 SUBROUTINE –
enable y    ; enable the y axis
ou 9        ; set output bits
return

:_ENABLE3   ; ENABLE AXIS 3 SUBROUTINE –
return      ; no action

:_ENABLE4   ; ENABLE AXIS 4 SUBROUTINE –
return      ; no action

; The following “disable” subroutines
; will execute when the axis is enabled, and
; the MMI Axis enable button is activated
```

```
:_DISABLE1  ; DISABLE AXIS 1 SUBROUTINE –
disable x
return

:_DISABLE2  ; DISABLE AXIS 2 SUBROUTINE –
me di "Disabled Axis 2"
di y
return
```

```
:_DISABLE3        ; DISABLE AXIS 1 SUBROUTINE –
return            ; no action

:_DISABLE4        ; DISABLE AXIS 1 SUBROUTINE –
return            ; no action

:_PAUSE_ON        ; PAUSE ON SUBROUTINE
pause enable      ; this will pause motion
ou 137            ; set output bits
me di "Pause"     ; message display
return

:_PAUSE_OFF       ; PAUSE ON (RESUME) SUBROUTINE
pause disable     ; resume motion
ou 255            ; set output bits
me di "Resumed"   ; message display
return
```

The Defined M-Code file text box specifies a file containing the definitions for an M-Code. An M-Code definition is basically a direct text replacement, however the locations where the replacement can take place are limited. The M-Code is designed to retrieve inputs and set outputs, but any command can be used as a replacement command. However, M-Codes can only be used as a separate command, within a variable statement, or as an argument in an IF statement. An example of a defined M-Code file follows.

In M-Code file:

```
M23 "OU 1"
M24 "$IN0"
```

In U500 program:

```
IF M24=1 :LABEL   ;SAME AS IF $IN0=1 :LABEL
M23               ;SAME AS OU 1
V0=M24/2          ;SAME AS V0=$IN0/2
```

Examples of both M-code and global subroutine files are installed in the U500\mmi\example subdirectory.

The “Abort Options” tab is shown in Figure 4-5. The Abort and Pause option tabs are similar. These tabs contain options for each of the eight output bits. Each bit can be turned on (pulled low), turned off (high impedance), or left unchanged. The Abort tab sets the output values to be changed when the Abort button is hit. The Pause tab sets the output values to be changed when the system is paused. When the pause is removed (pause button hit again), the outputs return to the state previous to the initial pause.

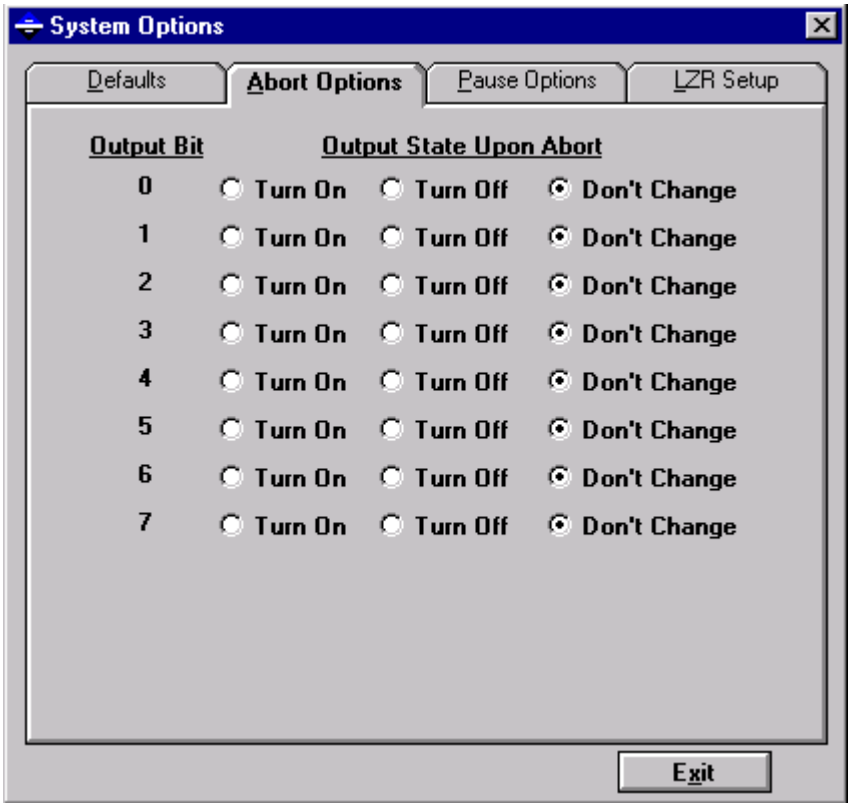


Figure 4-5. Abort Options Tab of the MMI System Options Screen

The LZR Setup options tab is shown in Figure 4-6. The LZR Setup tab is used if an LZR2100 laser interferometer card and/or a LZR1100 Environmental Compensation Unit accompany the UNIDEX 500 card. This tab sets the calibration file for the LZR1100 and/or the setup file for LZR2100. The base address for the LZR2100 laser interferometer card can be configured during the U500 software installation or by using the U500 Registry editor utility.

The LZR Options are as follows:

- "Do Not Use LZR Functions"

  This option disables LZR functionality.

- "Use the LZR2100 with the LZR1100"

  This option allows use of the PC-based LZR2100 board

- "Use the U500 with the LZR1100"

This option allows the LZR1100 Environmental Compensation unit to interface directly with A/D inputs on the U500 board. Refer to Chapter 7 for information on the "EC" command that is used with this option.

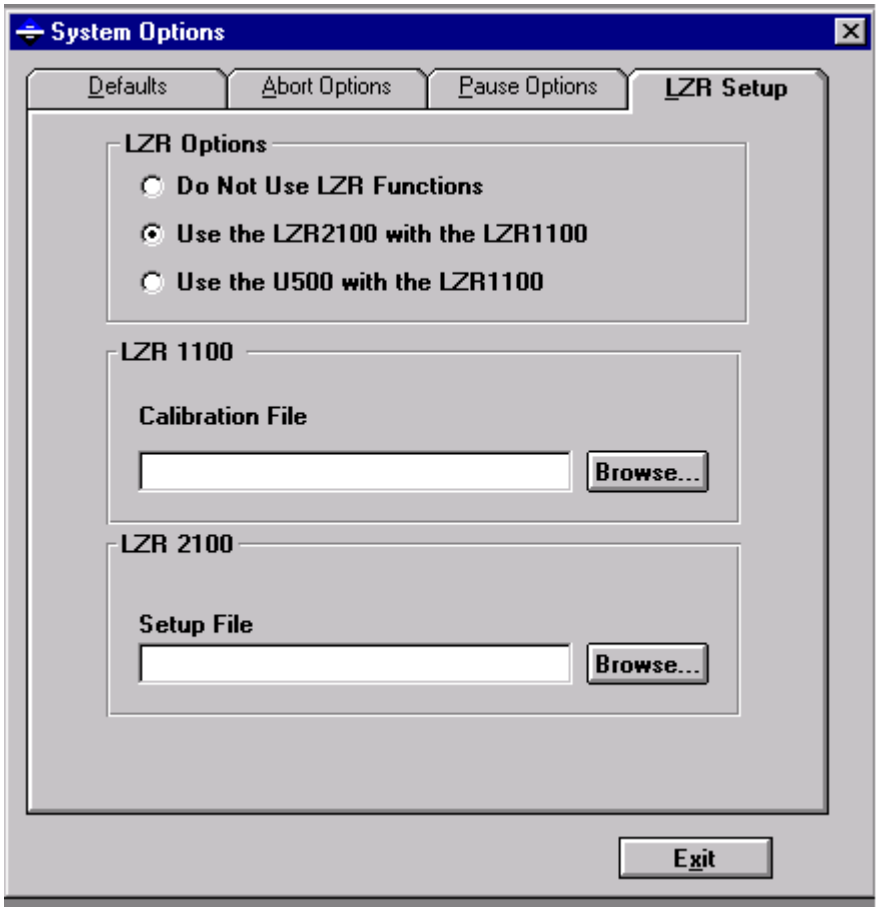


Figure 4-6. LZR Setup Tab of the MMI System Options Screen

The FPGA Setup options tab is shown in Figure 4-7. The FPGA Setup tab is used only with the U500 PCI. FPGA #1 can currently only be configured for encoder feedback operation. FPGA #2 is only available with the U500 PCI Ultra and currently allows the user to choose from the following three options:

Encoder 5-8	allows the U500 PCI to accept encoder feedback on channels 5- 8 for dual loop servo control
Single PSO	the U500 PCI will track the position of a single axis encoder channel. It can then generate an output pulse at a specified number of encoder counts. See Chapter 8 for more information on PSO functions.
Dual PSO	U500 PCI can track on any combination of two axes and will generate an output pulse at the vector distance specified. The U500PCI must also be factory configured to support this option. See Chapter 8 for more information on PSO functions.

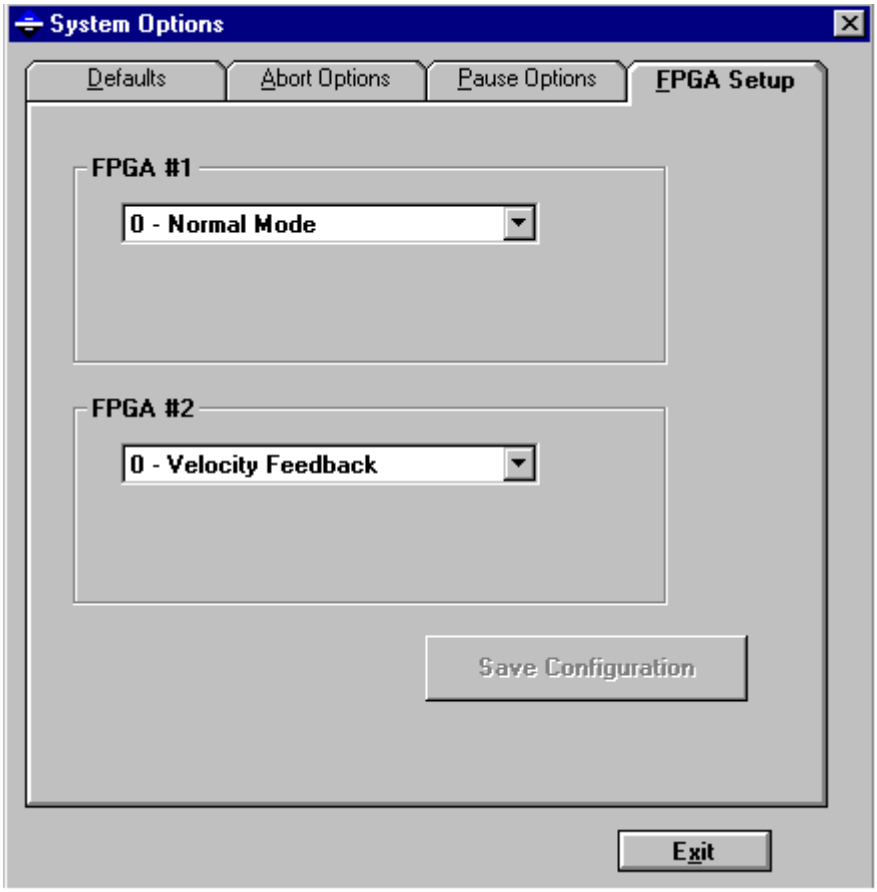


Figure 4-7. FPGA Setup Tab of the MMI System Options Screen

4.3.4. The Program Option of the Edit Menu

The Program option of the Edit menu is used to view/edit/create a motion control program for the UNIDEX 500 system. The program editor appears as a window within the main screen.



When the program editor is displayed, the options in the menu bar change. During program edits, the menu bar displays the File and Edit menus.

4.4. The Tools Menu

The Tools menu is a pull-down menu that contains the commands listed in Table 4-5.

Table 4-5. Tools Menu Commands

Command	Description
Diagnostics	Display diagnostics window
Demo	Enable demo mode for software "operation" without a U500 board
Axis Tuning...	Used to configure gain (and other) parameters to provide efficient axis motion
Joystick Digitizing	Used to create a program by recording positions using the joystick

4.4.1. The Diagnostics Option of the Tools Menu

The Diagnostics option of the Tools menu is used to display a dynamic window of software and hardware status fields for each of the four axes (X, Y, Z, and U). The Diagnostics window displays axis positions, analog-to-digital (A/D) input values, manual feed override, hardware/software faults, limits, traps, etc. The Diagnostics window is illustrated in Figure 4-8.

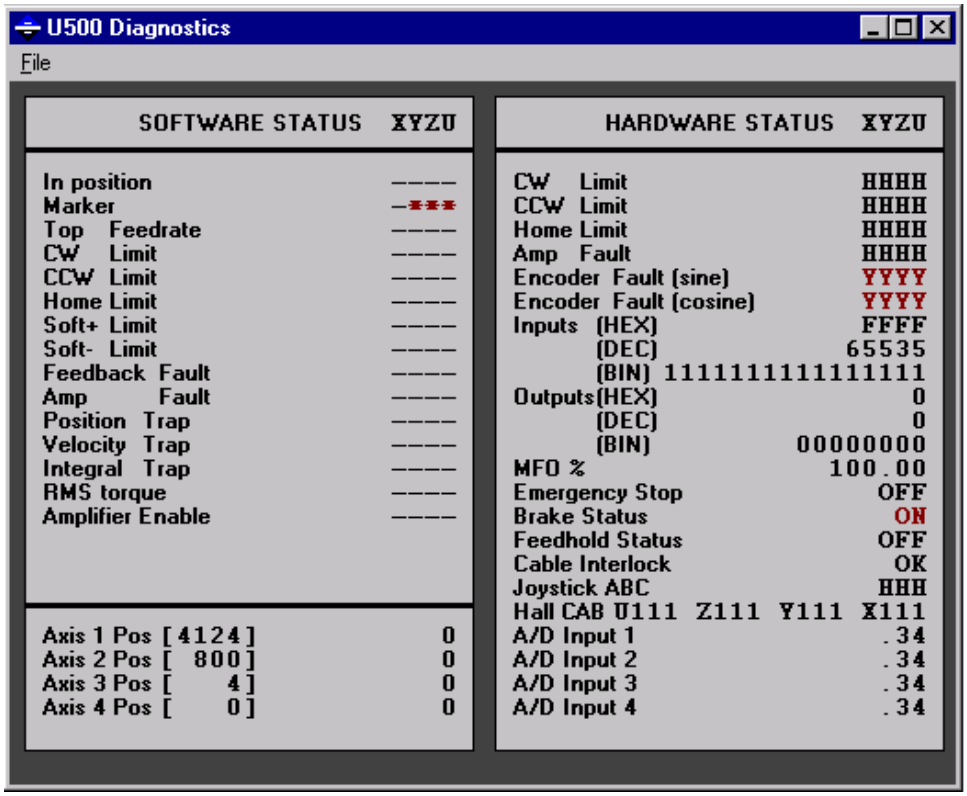


Figure 4-8. The Diagnostics Window

The Diagnostics window has a menu bar that contains a single menu -- the File menu. The File menu contains the Exit option that is used to close the Diagnostics window.

Hitting the <ENTER> key also closes the diagnostics window. When using the U500 PCI, the File menu also contains the Diagnostics 2 option that is used to provide diagnostic information for axes 5-8, the A/D Inputs 5- 8 and the additional 24 I/O lines that are available. Refer to Section 4.4.5. for more information concerning the Diagnostics 2 screen.

The Software Status portion of the Diagnostics screen contains information that is controlled by a parameter setting and requires that the axis be enabled. Normal conditions are displayed as a dash (-), and fault/trap/limit conditions are displayed as an asterisk (*). Each set of status flags contains four individual indicators, each corresponding to an axis from left to right (for example, **In position** - * - - indicates that axes 1, 3, and 4 have not reached their respective commanded positions and axis 2 has reached its commanded position).

The axis position portion of the Diagnostics screen contains the current positions (in machine steps) for each axis. This display will contain the corrections made by axis calibration, orthogonality correction, and backlash compensation. The corresponding value of the encoder counter or resolver to digital converter is shown in square brackets to the left.

The Hardware status portion of the Diagnostics screen displays the status of hardware related attributes. Examples of these attributes: the current CW and CCW limit status for each axis (L=low limit, H=high limit), the encoder fault status (Y=yes, N=no) for SIN and COS signals for each of the four encoders, input status, output status, etc. The hardware status portion also contains analog-to-digital input values for inputs 1-4.



Faults can be cleared (i.e., acknowledged) by pressing the F8 Fault Ack button.

The components of the Diagnostics screen are described in Table 4-6, Table 4-7, and Table 4-8.

Table 4-6. Software Status Diagnostics

Field	Description
In position	Indicates whether or not the axis has reached its commanded position (* = in position)
Marker	Indicates when the encoder marker is active (* = found)
Top Feedrate	Indicates if the current feedrate exceeds the top feedrate *(machine steps/ms)* parameter, x17 (* = feedrate exceeded x17)
CW Limit	Indicates if the CW hardware travel limit has been exceeded (* = CW limit switch was made and axis is enabled)
CCW Limit	Indicates if the CCW hardware travel limit has been exceeded (* = CCW limit switch was made and axis is enabled)
Home Limit	Indicates if the Home limit switch has been made (* = limit)
Soft CW Limit	Indicates if the CW software limit *(machine steps)* (x23) has been exceeded (* = x23 has been exceeded)

Table 4-6. Software Status Diagnostics (continued)

Field	Description
Soft CCW Limit	Indicates if the CCW software limit *(machine steps)* (x22) has been exceeded (* = x22 has been exceeded)
Feedback Trap	Indicates that the feedback signal from the feedback device has been lost (* = signal has been lost)
Amplifier Trap	Indicates that the amplifier is in a fault condition (* = amplifier in fault)
Position Trap	Indicates that the position error exceeds the *Max position error (0-8,388,607)* (x19) (* = current position is > value in x19)
Velocity Trap	Indicates that the velocity error exceeds the *Max velocity error (0-8,388,607)* (x18) (* = current velocity is > value in x18)
Integral Trap	Indicates that the integral error exceeds the *Max integral error (0-8,388,607)* (x20) (* = current integral error > value in x20)
RMS Torque	Current Trap. Indicates that the present output current exceeds the current defined by parameter x48 (*RMS current trap [0-100%]*).
Amplifier Enable	Indicates the current status of the axis (* = axis is enabled, - = axis is disabled)

Table 4-7. Axis Position Diagnostics

Field	Description
Axis 1 Pos	Current position of axis 1 in machine steps
Axis 2 Pos	Current position of axis 2 in machine steps
Axis 3 Pos	Current position of axis 3 in machine steps
Axis 4 Pos	Current position of axis 4 in machine steps

The axis position fields contain an additional value that is enclosed in brackets. This value is a hexadecimal number that shows the absolute position of the feedback device. In the case of encoders, the number displayed in brackets is not very useful and should be ignored. However, for resolvers, this number represents the absolute position of the resolver from the R/D hardware. The absolute position can range from 0x00 to 0xFFFF.



Table 4-8. Hardware Status Diagnostics

Field	Description
CW Limit	Indicates the current hardware input level of the CW limit input (H = "high" signal, L = "low" signal)
CCW Limit	Indicates the current hardware input level of the CCW limit input (H = "high" signal, L = "low" signal)
Home Limit	Indicates the current hardware input level of the Home input (H = "high" signal, L = "low" signal)
Amplifier Fault	Indicates that an amplifier is in a fault condition (H = "high" fault signal; L = "low" fault signal)
Encoder Fault (Sine)	Indicates that the U500 detected that the SIN and $\overline{\text{SIN}}$ signals are in the same state
Encoder Fault (Cosine)	Indicates that the U500 detected that the COS and $\overline{\text{COS}}$ signals are in the same state
Inputs (HEX)	Indicates the states of the 16 digital inputs in hexadecimal format from 0000 to FFFF
Inputs (DEC)	Indicates the states of the 16 digital inputs in decimal format from 0 to 65,535
Inputs (BIN)	Indicates the states of the 16 digital inputs in binary format from 0000 0000 0000 0000 to 1111 1111 1111 1111
Outputs (HEX)	Indicates the states of the 8 digital outputs in hexadecimal format from 00 to FF
Outputs (DEC)	Indicates the states of the 8 digital outputs in decimal format from 0 to 255
Outputs (BIN)	Indicates the states of the 8 digital outputs in binary format from 0000 0000 to 1111 1111
MFO %	Indicates the current manual feedrate override percentage from 0% to 199%
Emergency Stop	Indicates the current emergency stop status (On or Off)
Brake Status	Indicates the current brake status (On or Off)
Feedhold Status	Indicates the current feedhold status (On or Off) ("On" means that a pause condition has occurred)
Cable Interlock	Indicates if an OP500 cable interlock error has occurred (Yes or No). Yes indicates that pins 1 and 100 of the U500 are not connected (e.g., the cable between the U500 and DR500 is loose or disconnected)

Table 4-8. Hardware Status Diagnostics (continued)

Field	Description
Joystick ABC	Indicates the current input status of joystick buttons A, B and C. The format is ABC, where A is status of A button (L = press, H = no press), and B is status of B button (L = press, H = no press). If C is pressed, both position A and B go low (e.g., LLH)
Hall CAB	Applicable only with AC brushless motors. Indicates the state of Hall sensors. See Motor Setup command
A/D Input 1	Indicates the direct analog/digital converter voltage (0-5 V) for input 1
A/D Input 2	Indicates the direct analog/digital converter voltage (0-5 V) for input 2
A/D Input 3	Indicates the direct analog/digital converter voltage (0-5 V) for input 3
A/D Input 4	Indicates the direct analog/digital converter voltage (0-5 V) for input 4

4.4.2. The Demo Option of the Tools Menu

The Demo option of the Tools menu is used as a diagnostic tool to allow access to virtually all of the software screens and menus without having to install a UNIDEX 500 board in the PC. When this option is selected, a check mark appears next to "Demo" in the menu. This allows the operator to navigate freely through the software menus without experiencing error messages that keep the operator from continuing to certain screens.

Even with demo mode selected, there are a small number of screens that are not accessible if a UNIDEX 500 board is not installed in the PC. Such screens include the axis tuning screens and others.



When this option is selected, it toggles the current state of the demo mode. A check mark next to "Demo" in the menu indicates that demo mode is active. An absence of a check mark indicates that demo mode is disabled.

4.4.3. The Axis Tuning... Option of the Tools Menu

The Axis Tuning... option of the Tools menu is used to display Axis Scope, a display and loop tuning utility. When this option is selected, the Axis Scope screen is displayed. The Axis Scope screen contains a menu bar with loop tuning and display options. These menus are listed and described in Table 4-9. Information about servo loop tuning is discussed in Chapter 6: Tuning Servo Loops.

Table 4-9. Menu Items on the Axis Scope Screen

Command	Description
File	Save/load plot (.PLT) files, save ASCII files, exit
Plot	Specifies plot options and which functions to plot
Trigger	Data collection method, motion and control options
Collect	Defines the number of points to be collected
Display	Defines the number of points to be displayed
Axis	Specifies the axis (1, 2, 3 or 4) to be displayed
Units	Specifies distance and time units for the display
Tools	Enables/disables the cursor, status, control, and gains tools (menu bars with handy features)

4.4.3.1. Axis Scope: The File Submenu

The File submenu of the Axis Scope screen contains options that allow the operator to save binary plot (.PLT) files, save plot information as ASCII text (.TXT) files, load a previously saved plot (.PLT) file, and exit (return to the main screen). An option also allows screen plots to be sent to a printer. Submenu options are listed in Table 4-10.

Table 4-10. File Submenu Options in Axis Scope

Command	Description
Save	Saves plot results to current binary plot (.PLT) file
Save As...	Saves plot results to a new binary plot (.PLT) file
Save ASCII	Saves plot results as an ASCII text (.TXT) file
Load...	Loads (from disk into memory) a previously saved binary plot (.PLT) file and displays it on the Axis Scope screen
Print	Sends screen plots to the printer
Exit	Closes the Axis Scope screen

4.4.3.2. Axis Scope: The Plot Submenu

The Plot submenu of the Axis Scope screen contains options that allow the operator to specify plot options and which functions to plot. Submenu options are listed in Table 4-11.

Table 4-11. Plot Submenu Options in Axis Scope

Command	Description
Velocity Feedback	Plots velocity feedback of the selected axis
Velocity Command	Plots commanded velocity to the selected axis
Velocity Error	Plots velocity error of the selected axis
Position Feedback	Plots position feedback of the selected axis
Position Command	Plots commanded position to the selected axis
Position Error	Plots position error of the selected axis
Torque	Plots the output torque of the selected axis
Analog Input	Plots the analog input values of the selected axis
Integrator	Plots control loop integrator value for the axis
Inputs	Plots the status of one or more of the 16 inputs.
Outputs	Plots the status of one or more of the 8 outputs.
Zero Line	Aligns the plot to the vertical zero line
Refresh	Clears the display and replots the data

The first nine options in the Plot submenu allow the operator to specify one or more functions to be plotted over time. The nine options are Velocity Feedback, Velocity Command, Velocity Error, Position Feedback, Position Command, Position Error, Torque, Analog Input, and Integrator. The displayed values for velocity and position are dependent on the settings in the units menu. The torque is displayed in volts output by the Digital-to-Analog converter's range of -10 to +10V. The analog input ranges from 0 to 5V for the standard U500 ISA 8 bit A/D, -10 to 10V for the U500 ISA optional 12 bit A/D, and –10 to 10V for the U500 PCI. One or more of these functions can be selected as the vertical axis (or axes) of the plots in the display window.

The next three options, Torque, Analog Input, and Integrator cannot be displayed simultaneously. The desired signal must be selected (checked) before sampling the data. The last three options in the Plot submenu, the Zero Line option, the Refresh option, and the Overlap option, allow the operator to customize the look of the plot display.

4.4.3.3. Axis Scope: The Trigger Submenu

The Trigger submenu of the Axis Scope screen contains options that allow the operator to specify a data collection method (Collect One Set of Data or Collect Data Continuously), motion options (Single Step Motion, Auto Step Motion, Forward Motion, Reverse Motion), and control options (Stop, Abort, and Sample Rate). Submenu options are listed in Table 4-12.

When selected, the Forward Motion... and Reverse Motion... options display popups that allow the user to define the motion command to be used for forward and reverse axis motions, respectively. These popup boxes are displayed in Figure 4-9 and Figure 4-10.

Table 4-12. Trigger Submenu Options in Axis Scope

Command	Description
Collect One Set of Data	Collects one set of data (as specified in Collect submenu), plot the results, and stops. No motion is commanded
Collect Data Continuously	Collects one set of data (as specified in Collect submenu), plots the results, and starts over again. No motion is commanded
Single Step Motion	Performs commanded motion one step at a time (alternately sends the forward motion and the reverse motion and collects data during each cycle)
Auto Step Motion	Performs motion continuously (automatically, same as single step only continuous)
Forward Motion...	Prompts operator to specify a desired forward step motion command (e.g., G1 X1 F1000)
Reverse Motion...	Prompts operator to specify a desired reverse step motion command (e.g., G1 X1 F1000)
Stop	Stops axis motion after the current data set has been collected and then updates the plot
Abort	Stops axis motion immediately (aborts motion) and the plot is not updated
Sample Rate	Prompts the operator to specify how often a sample is read (given in milliseconds [ms])

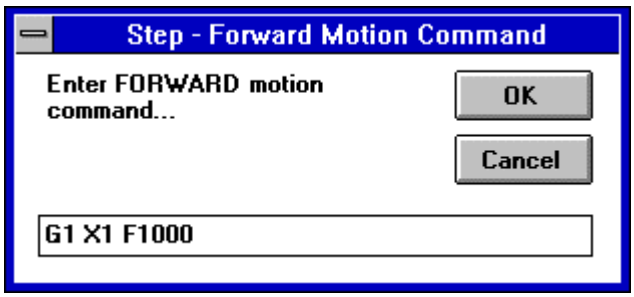


Figure 4-9. Forward Motion... Popup Window

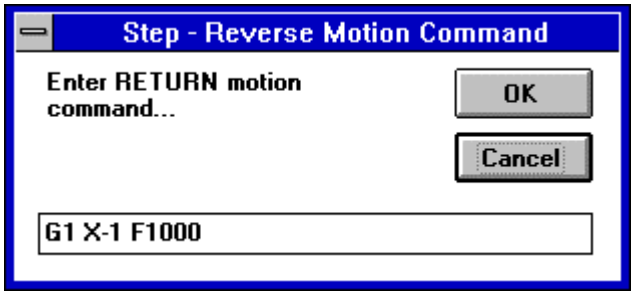


Figure 4-10. Reverse Motion... Popup Window

When the Sample Rate option is selected, the software displays the Scope Sample Timebase popup window. From this popup the operator enters the frequency at which samples are to be taken. This value is given in milliseconds (ms) and defaults to a value of 1 ms. This popup window is illustrated in Figure 4-11.

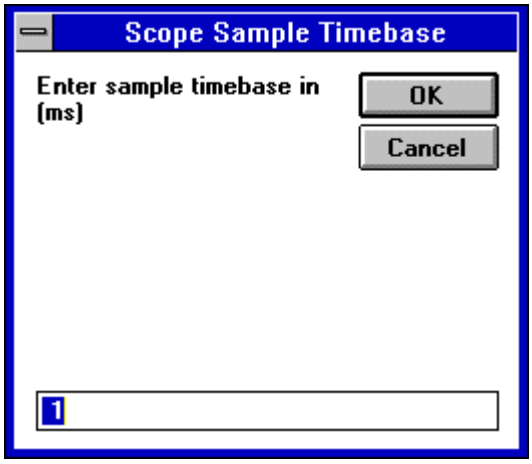


Figure 4-11. Sample Rate Popup Window

4.4.3.4. Axis Scope: The Collect Submenu

The Collect submenu of the Axis Scope screen specifies the number of data points to be collected in a single set. Once these points are collected, they are displayed on the Axis Scope screen according to the settings of other submenu items. The Collect submenu contains five data set sizes: 100, 250, 500, 1000, and 2500 points (the Base model can only collect 1000 points). Only one of these data sizes can be selected at a given time. The Collect submenu is illustrated in Figure 4-12.

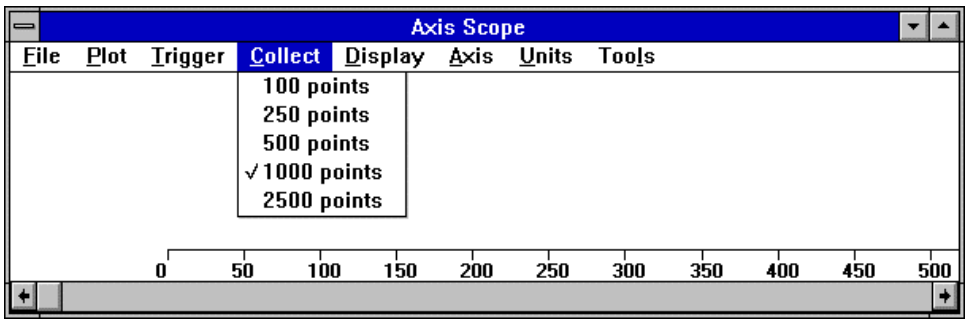


Figure 4-12. Collect Submenu of the Axis Scope Screen

4.4.3.5. Axis Scope: The Display Submenu

The Display submenu of the Axis Scope screen is similar to the Collect submenu. Display specifies the number of data points that are displayed or plotted. Like the Collect submenu, the Display submenu contains five data set sizes: 100, 250, 500, 1000 and 2500 points. Only one of these data sizes can be selected at a given time. The Display submenu is illustrated in Figure 4-13.

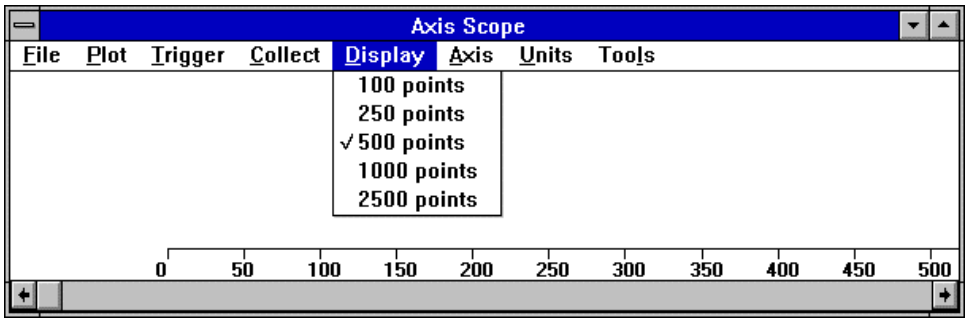


Figure 4-13. Display Submenu of the Axis Scope Screen

4.4.3.6. Axis Scope: The Axis Submenu

The Axis submenu of the Axis Scope screen specifies which axis to display. The functions shown in the Plot menu refer to the current axes as selected by this submenu. This submenu supports axes 1, 2, 3, and 4. When selected, the axis name has a check mark to its left. The Axis submenu is illustrated in Figure 4-14. (The UNIDEX 500 Base model can only collect data for a single axis. The Plus and Ultra can collect all four axes of data).

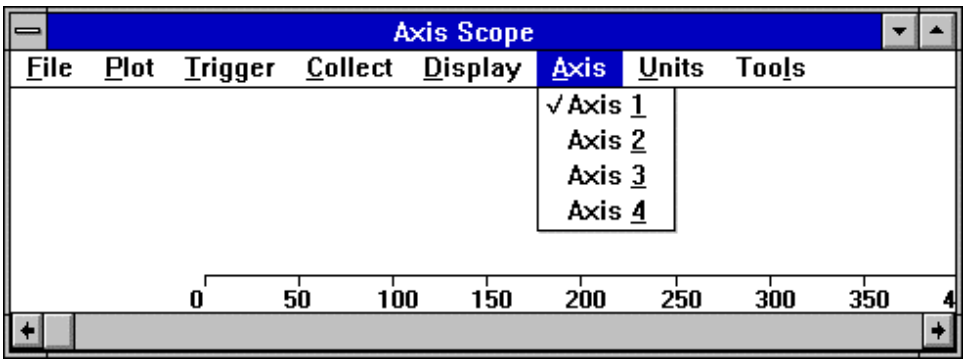


Figure 4-14. Axis Submenu of the Axis Scope Screen

4.4.3.7. Axis Scope: The Units Submenu

The Units submenu of the Axis Scope screen specifies the distance and time measurement units for the plot display. Distance units can be displayed in any one of the following units: Machine Steps, millimeters (mm), microns (mm / 1000), Inches, and thousandths of inches (Inches / 1000). Time units can be displayed in either Seconds or milliseconds (Seconds / 1000). The Units submenu is illustrated in Figure 4-15.

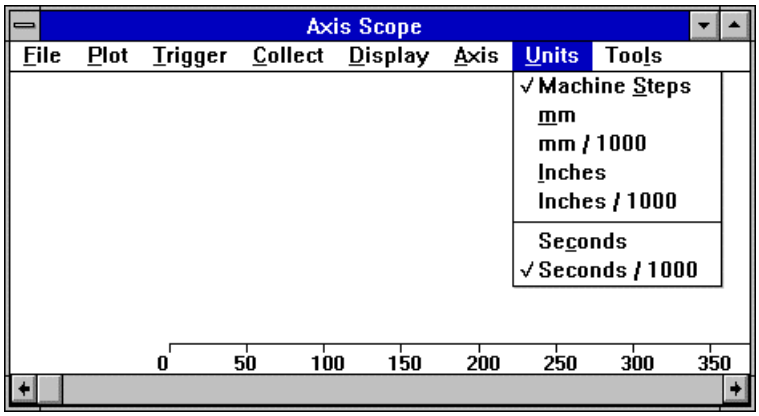


Figure 4-15. Units Submenu of the Axis Scope Screen

4.4.3.8. Axis Scope: The Tools Submenu

The Tools submenu of the Axis Scope screen enables/disables the display of four tool bars on the Axis Scope screen. The options of this submenu are: Cursors, Status, Control, Gains, Autotune, Frequency Response, and FFT. Selecting an option from this submenu toggles the display of the associated tool bar. When a tool bar is being displayed, a check mark appears to the left of the associated option in the Tools submenu. The Tools submenu is illustrated in Figure 4-16.

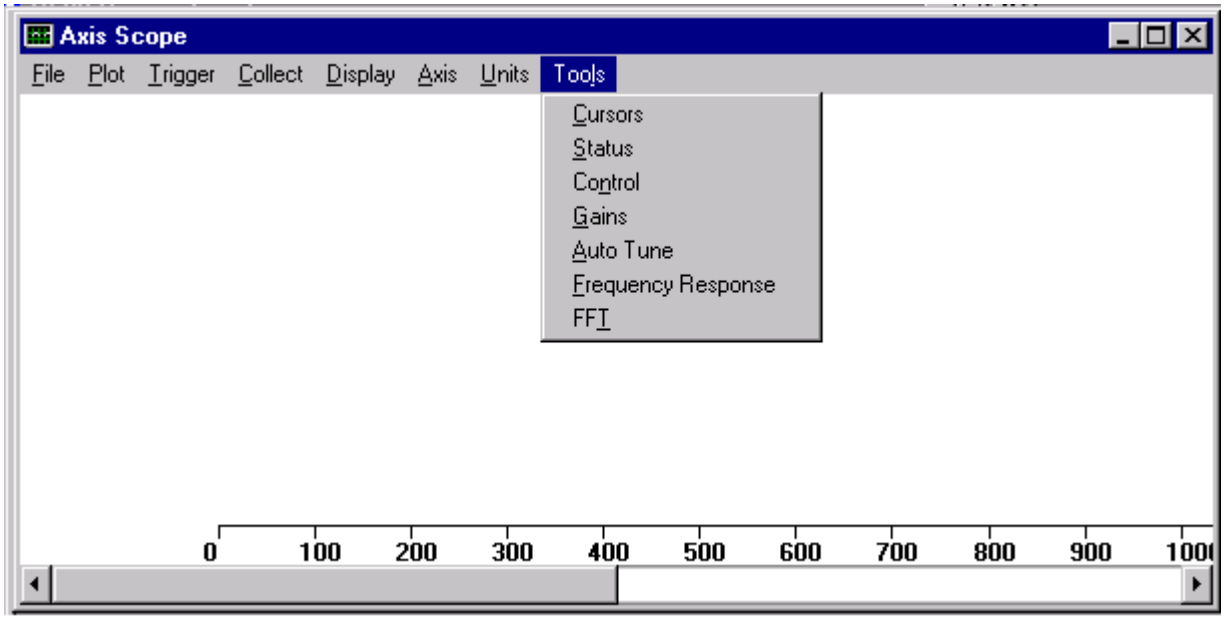


Figure 4-16. Tools Submenu of the Axis Scope Screen

The Cursors option is used to display/hide the Cursors tool bar. This tool bar contains features that assist the operator in determining time differences between points on the plot as well as frequency information. The Cursors tool bar is illustrated in Figure 4-17.

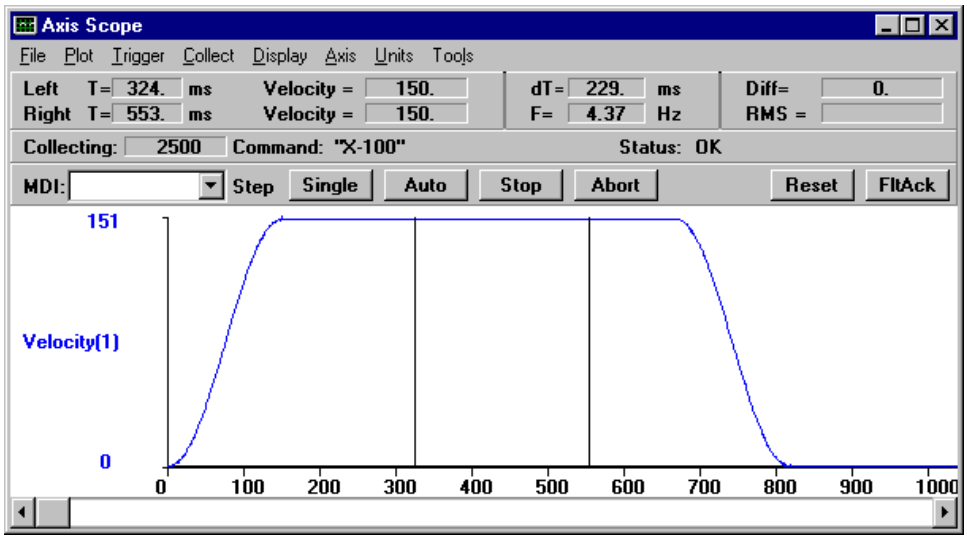


Figure 4-17. Cursors Tool Bar of the Axis Scope Screen

The Status option is used to display/hide the Status tool bar. This tool bar contains fields that display the current status of the data collection process and axis fault information. The tool bar is illustrated in Figure 4-18.

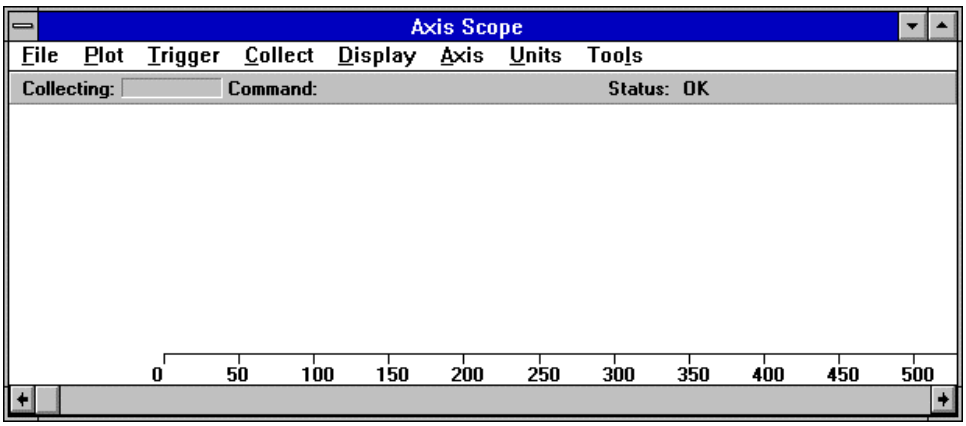


Figure 4-18. Status Tool Bar of the Axis Scope Screen

The Control option is used to display/hide the Control tool bar. This tool bar contains features such as an MDI command box, program step buttons, and control buttons. The Control tool bar is illustrated in Figure 4-19.

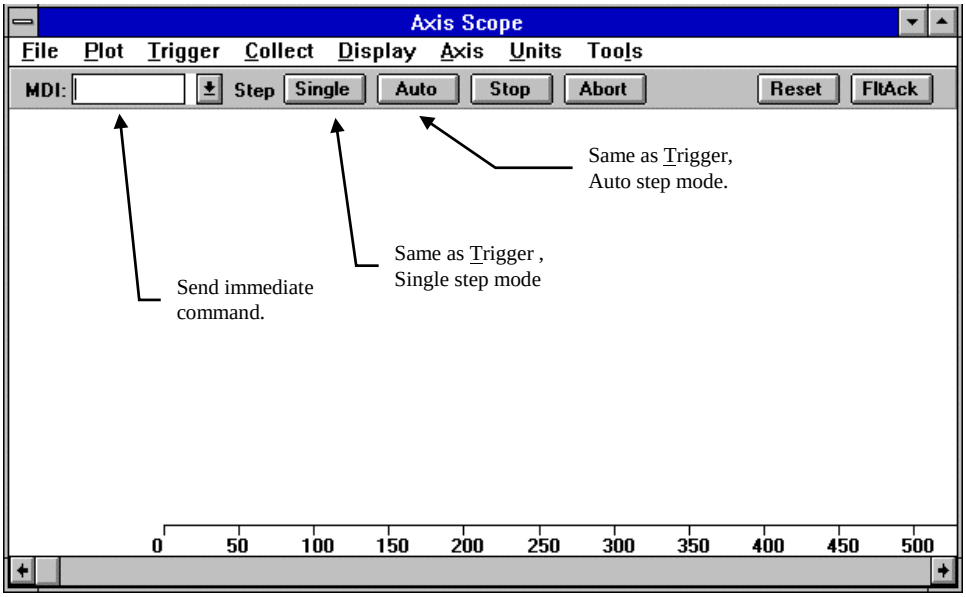


Figure 4-19. Control Tool Bar of the Axis Scope Screen

The Gains option is used to display/hide the Gains tool bar. This tool bar contains an axis selection button as well as loop tuning gains fields for easy parameter access. The Gains tool bar is illustrated in Figure 4-20.

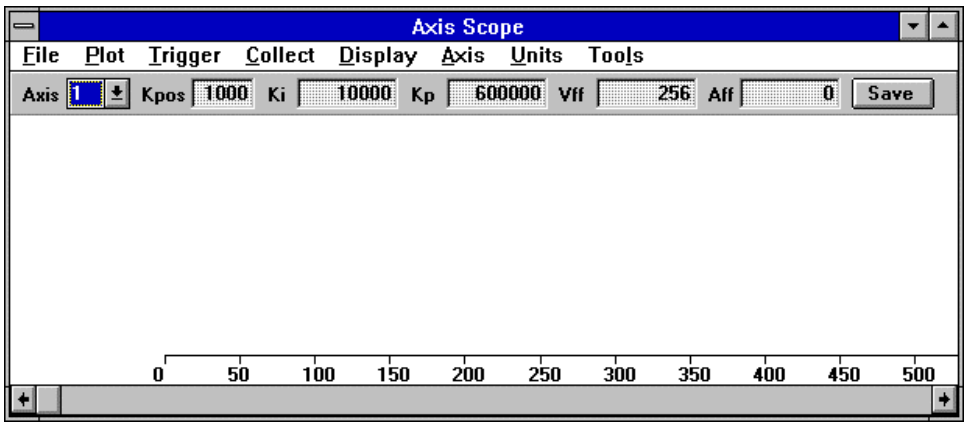


Figure 4-20. Gains Toolbar of the Axis Scope Screen

The Autotune option is used to display/hide the autotune tool bar. See Section 6-7 for a description on how to use autotuning.

The Frequency Response is used to display/hide the frequency response toolbar. The Frequency Response toolbar is shown in Figure 4-21. The frequency response is used to return amplitude and phase response of the system at multiple frequencies. The axis will be excited with a sinusoidal command of the distance entered and starting frequency entered in the Freq (Hz) box. The number of sinusoidal commands at one frequency is set by the cycles box. The number of different frequencies excited is set in the Freqs box. The frequencies are increased by 1 Hz for each subsequent frequency.

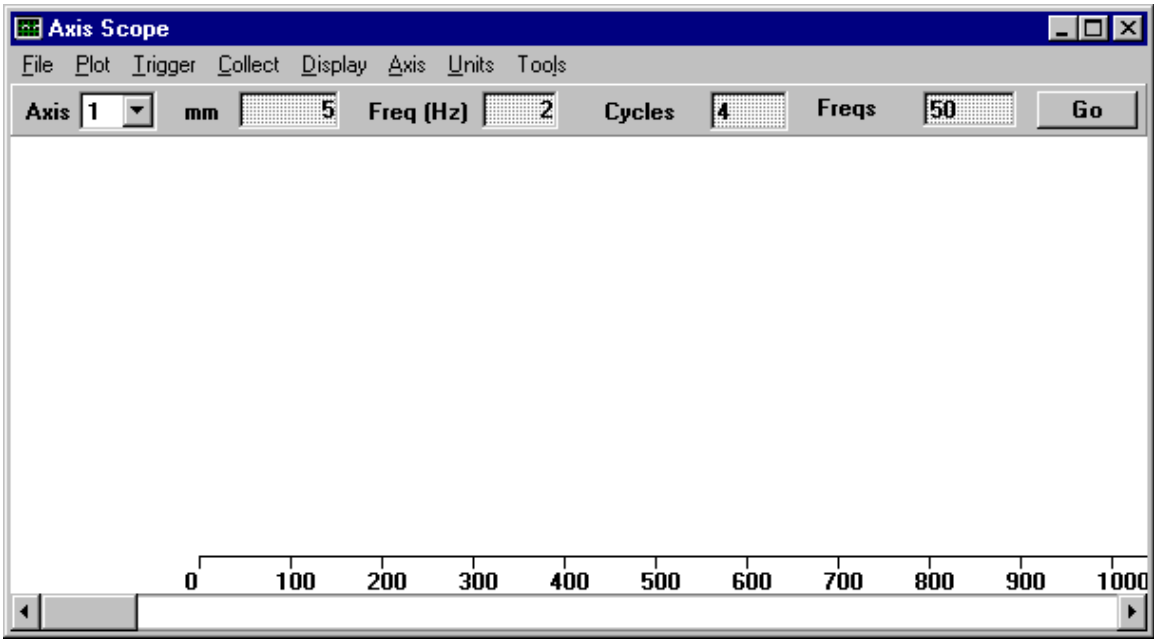


Figure 4-21. Frequency Response Toolbar of the Axis Scope Screen

The FFT option in the Axis scope window is used to display/hide the FFT window. This window can be used to apply the Fourier Transform to a set of data collected in the Axis Scope window. The FFT is used to convert the time-based input data into its frequency components. For example, the FFT can be used to identify a resonant frequency or a 60 Hz ground loop that is affecting the system.

The FFT window is shown below (Figure 4-22).

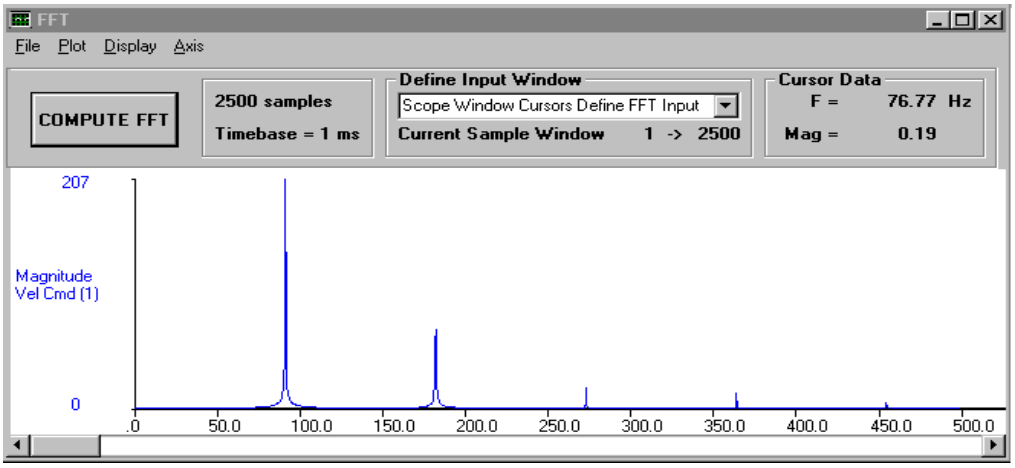


Figure 4-22. FFT Toolbar of the Axis Scope Screen

The File submenu (Table 4-13) of the FFT screen contains the following options that allow the operator to save plot information as ASCII text (.TXT) files, output the screen plots, and exit (return to the Axis Scope Window).

Table 4-13. File Submenu Options of the FFT screen

Command	Description
Save ASCII	Saves plot results as an ASCII text (.TXT) file
Print	Sends Screen plots to the printer
Exit	Closes the FFT window

The Plot Submenu (Table 4-14) of the Axis Scope screen contains options that allow the operator to specify the input data and the type of plot.

Table 4-14. Plot Submenu Options of the Axis Scope Screen

Command	Description
Velocity Feedback	Computes the FFT of the velocity feedback of the selected axis.
Velocity Command	Computes the FFT of the commanded velocity to the selected axis.
Velocity Error	Computes the FFT of the velocity error of the selected axis

Table 4-14. Plot Submenu Options of the Axis Scope Screen (continued)

Command	Description
Position Feedback	Computes the FFT of the position feedback of the selected axis
Position Command	Computes the FFT of the comanded position to the selected axis
Position Error	Computes the FFT of the position error of the selected axis
Torque	Computes the FFT of the output torque of the selected axis
Analog Input	Computes the FFT of the analog input values of the selected axis.
Integrator	Computes the FFT of the control loop integrator value for the selected axis
Log Plot	Aligns the plot to the vertical zero line

The first nine options in the Plot submenu (Table 4-14) allow the operator to specify the data from the Axis Scope window that will be analyzed. The FFT of one of these data sets can be selected as the vertical axis of the plot in the display window. The desired signal must be selected (checked) before computing FFT. The last option in the Plot submenu, the Log Plot option, allows the operator to plot the $\log_{10}$ values of the FFT magnitude.

The Display submenu of the FFT window specifies a range of data points that are displayed. The Display submenu contains five data set sizes: 5%, 10%, 25%, 50%, 75%, and 100%. Only these data sizes can be selected at a given time.

The Axis submenu of the FFT window specifies which axis to display. The functions shown in the Plot menu refer to the current axis as selected by this submenu. This submenu supports axes 1, 2, 3, and 4. When selected, the axis name has check mark to its left.

The "Compute FFT" command button will calculate the FFT of the scope window data that was selected using the Plot submenu. By default, the FFT uses the entire range of data that was collected in the scope window. However, the "Define Input Window" option box provides the option to use the right and left Axis Scope window cursors to define an input range for the FFT.

The FFT window cursors can be used to obtain the magnitude and frequency of a single point. The equations used to calculate the FFT data are:

$$y_p = \sum_{k=0}^{n-1} x_k[\cos(2\pi kp/n) + j*\sin(2\pi kp/n)]$$

X_k – kth time domain

Y_p – pth frequency domain sample

N - total number of samples

The magnitude data that is displayed in the FFT window is the vector sum of each frequency domain sample.

$$m_p = (Re(Y_p)^2 + Im(y_p)^2)^{1/2}$$

The Nyquist frequency of the data is determined by the time-base used to collect the data. For example, the sample rate in the Axis Scope window was 1ms (1kHz), then the Nyquist frequency is 500 Hz.

4.4.4. The Joystick Digitizing Option of the Tools Menu

Joystick digitizing permits the user to teach the controller a set of points and store them for future use. Through this option the user gains the ability to work around or trace odd shapes with the joystick, thus creating a shape or program for the axis (or axes) to follow.

The joystick digitizing menu option is only available when at least one axis is enabled and the software does not have the program execution screen opened. If the Joystick Digitizing menu item is shown in a lighter font (e.g., Joystick Digitizing as opposed to Joystick Digitizing), then digitizing is unavailable and the program execution window must be closed and at least one axis enabled. Also, the joystick (Figure 4-23) must be properly connected to allow entry to the joystick digitizing screen.

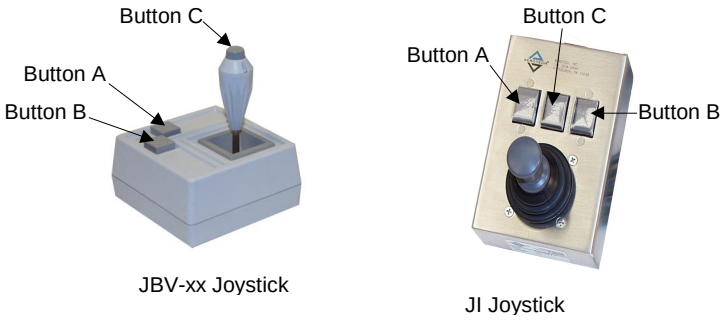


Figure 4-23. U500 Joysticks

The Joystick Digitizing window contains a radio buttons section and a program list box (refer to Figure 4-24). The radio buttons allow the user to select the type of motion between the current and next point. The “Linear” buttons will move the axis linearly to the next point. “CW” and “CCW” will create an arc of the move to the next point (to determine the trajectory of the arc, two points and 2 axes must be selected to have the same arc type to do either CW or CCW digitizing).

To select a particular point, move the axes to the desired point and hit joystick button C. If linear motion for an axis was selected, the motion will be inserted into the program list box.

This move will be either an incremental or absolute distance depending on the programming mode being used.

If “CW” or “CCW” was selected, move the axes to one location along the arc and hit joystick button C. Then move the axes to the end point of the arc and hit joystick button C again. The software will calculate the proper arc and enter a command into the program list box.

The program list box acts like a normal text editor. Commands can be added surrounding the digitized points to create the desired parts program.

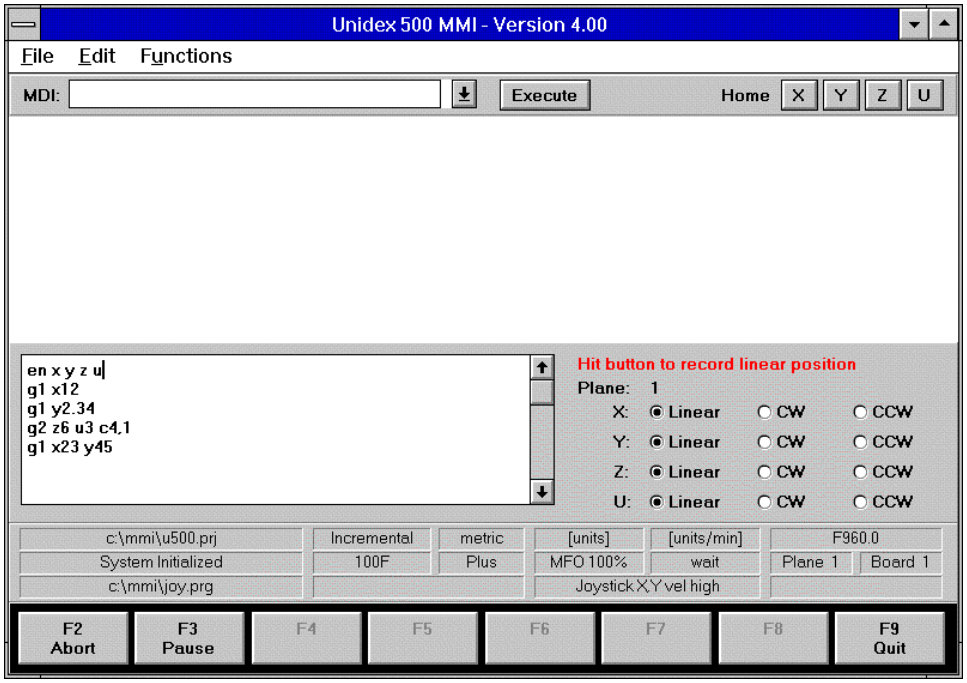


Figure 4-24. Joystick Digitizing Window

4.4.5. The "Diagnostics 2" Option of the Tools Menu (U500 PCI Only)

When using the U500 PCI, the File menu also contains the Diagnostics 2 option (Figure 4-25). This is used to provide diagnostic information for the additional 24 I/O lines, A/D inputs 5 – 8 and the status for axis 5 – 8.

Although the diagnostic windows are not identical, the definitions of the fields in both the Software Status portion and the Hardware Status portion of this window are the same as for the original Diagnostics window. Please refer to section 4.4.1. for field definitions.

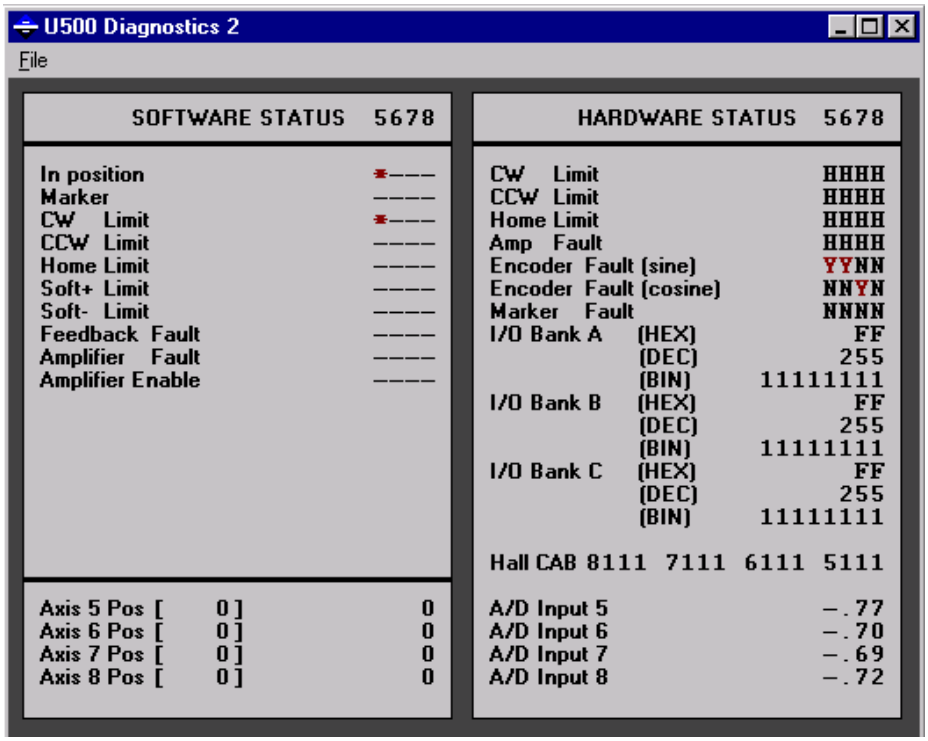


Figure 4-25. The Diagnostics 2 Window

The Software Status portion of the second Diagnostics screen contains 10 sets of status flags for each of the additional four axes. As in the main Diagnostic screen, normal conditions are displayed as a dash (-), and fault/trap/limit conditions are displayed as an asterisk (*). Each set of status flags contains four individual indicators, each corresponding to an axis from left to right (for example, **In position** - * - - indicates that axes 5, 7, and 8 have not reached their respective commanded positions and axis 6 has reached its commanded position). The Software Status is implemented in Version 5.12 and higher of the MMI.

The axis position portion of the Diagnostics screen contains the current positions (in machine steps) for axis 5 - 8. The corresponding value of the encoder counter or resolver to digital converter is shown in square brackets to the left.

The Hardware status portion of the Diagnostics screen displays the status of hardware related attributes (such as: the current CW and CCW limit status for each axis (L=low

limit, H=high limit), the encoder fault status (Y=yes, N=no) for SIN and COS signals for each of the four encoders, etc…). The values of analog-to-digital inputs 5-8 are also shown in this window.

The additional set of 24 I/O bits provided by the U500PCI are also displayed in this Diagnostic window. The I/O is arranged as 3 ports of 8 bits. Each port is configurable as outputs or inputs. The direction of each port is set with the "IOSET" command. IO data is read and written using the "IO" command. (Refer to Chapter 7 "Programming Commands). The active state of all outputs is low. Therefore a programmed logic "1" will result in a low impedance to GND for that output. A programmed logic of "0" will result in a high impedance state for that output. All IO bits become inputs (high impedance) during reset.

4.4.6. The Variable Watch Option of the Tools Menu

The Variable Watch Window (shown in Figure 4-26) provides a display for the 256 user variables available with the U500. The value of a variable can be changed directly by double-clicking the mouse on the selected variable. An input box will appear, prompting the user to enter the new value for the selected variable.

The order in which the variables appear can also be changed to provide a customized display. For example, the user may need to watch the state of Variable 0 and Variable 100 as they are changed by a U500 parts program. This can be done by selecting V1, and clicking the right mouse button. An input box will appear, prompting the user to enter the new variable to display in place of variable V1. Entering the value of 100 will cause the value of Variable 100 to be displayed directly below the value of Variable 0.

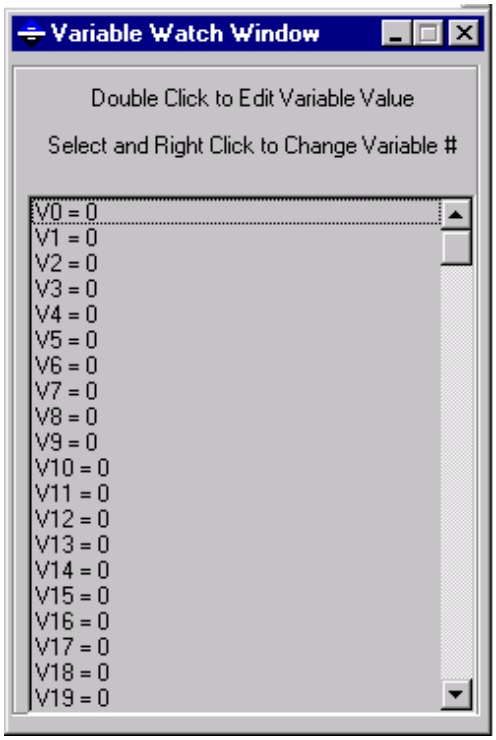


Figure 4-26. The Variable Watch Window

4.5. The Windows Menu

The Windows menu is a pull-down menu of the U500 software that contains the commands listed in Table 4-15. Figure 4-27 shows the main software screen with all of the available windows enabled.

Table 4-15. Windows Menu Commands

Command	Description
Functions	Displays software buttons F2 (Abort) through F9 (Reset)
Status	Displays a 16-field status window of system status
Jog	Displays jog window for axis control (program window [F4 Load] is closed if Jog window is open)
Gain Adjust	Displays the Gain Adjust popup for loop tuning
Mdi	Displays the immediate mode command entry window
MFO	Displays the manual feed override adjustment window
Axis Display	Displays a cascading menu of axis display options such as enable buttons, plane information, axis status fields, and position fields

4.5.1. The Functions Option of the Windows Menu

The Functions option of the Windows menu is used to toggle the display of the Functions menu. The Functions menu appears in the lower portion of the software screen and consists of eight software buttons. The software buttons of the Functions window correspond to function keys F2 through F9 on the standard keyboard. Refer to Figure 4-27. Each of these buttons is described in detail in following sections.

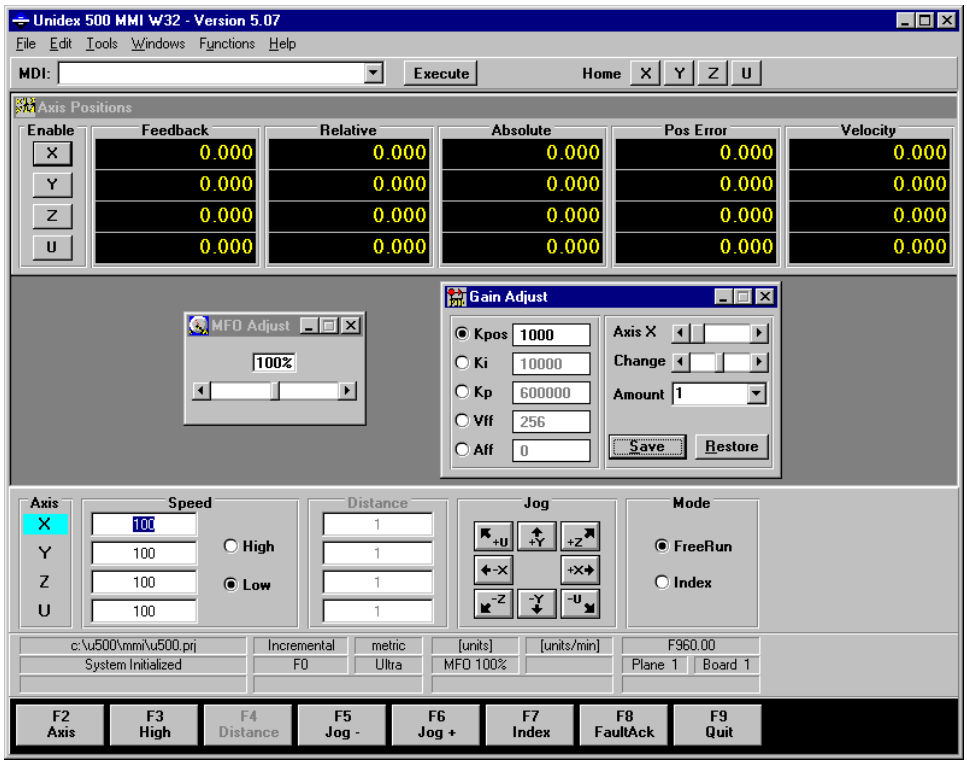


Figure 4-27. Software Screen with All Windows Visible

4.5.1.1. The F2 Abort Button

The F2 Abort button is used to terminate the execution of the program or any motion that is currently running. The U500 command buffer is cleared. After program execution has been stopped (aborted), the program can be restarted using the Restart button (on the program display window [after selecting F4 Load]) and then the Cycle button.

4.5.1.2. The F3 Pause/Resume Button

The F3 Pause button is used to temporarily suspend execution of the current motion. Each axis ramps down to a stop at the max AC/DC value as set in the parameters. Once a program is paused, the F3 Pause button changes to the F3 Resume button. When the F3 Pause button is pressed, any axis motion is stopped until the Resume button is pressed. When the Resume button is pressed, the axes will ramp back up to speed and continue the program trajectory.

4.5.1.3. The F4 Load Button and the Program Display Window

The F4 Load button is used to bring a program file from disk into memory so that it can be run. When this button is selected, an Open Program File window is displayed. From this window, the operator selects the desired program (.PRG). When the loading process is complete, the Program Display window is activated and the program code is displayed.

This window (shown in Figure 4-28) displays the program code as well as six program control buttons.

If the F5 Jog button is selected, the Program Display window is closed and the Jog window is displayed. The Program Display window is displayed again after the Jog window is closed.

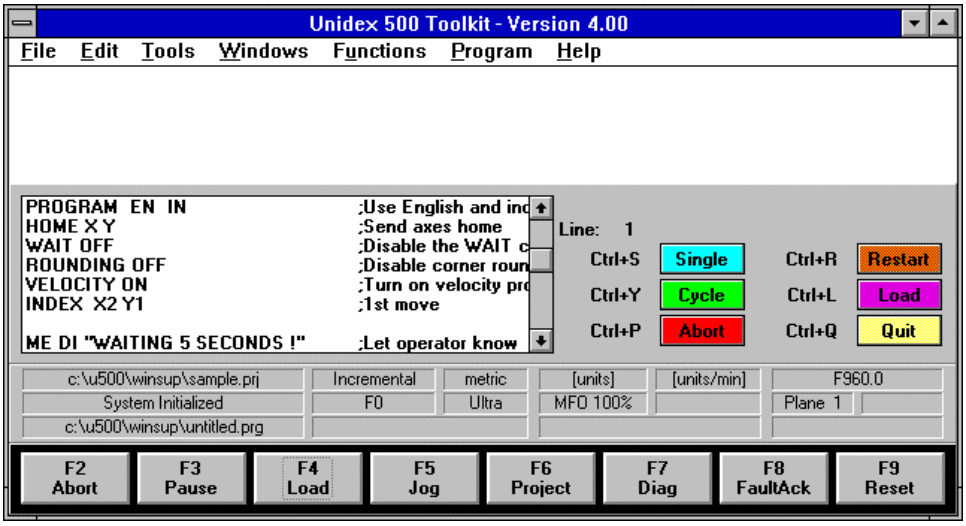


Figure 4-28. Program Control Window after F4 Load of UNTITLED.PRG

The Program Display field is a rectangular window on the left side of the Program Display window. This field contains the program code of the currently loaded program. The operator can scroll through the program using the vertical scroll bar located on the right of the Program Display field (when the program size exceeds the size of the window).

Clicking on a line in the Program Display window will move the highlight bar to this line. If cycle is hit, program execution will proceed from this line.

The Program Display field can alternately show the messages displayed by an executing program. The default display can be set up in the System Options screen, with the "Program Display Window" option. The display can also be toggled by double clicking on the status block that shows either "Program" or "Messages."

Program code is edited using the Edit / Program... option from the main menu bar.

Besides the program display, this window contains six program control buttons: Single/Auto, Cycle, Abort, Restart, Load, and Quit. The Single/Auto button selects how the program is run – Single (one line at a time; the operator must press Cycle after each

line is executed) or Auto (the program runs completely until it is stopped, aborted or an error occurs).

The Cycle button is used to start execution of the .PRG file that is currently loaded into memory. In single step mode, this button must be pressed after each line of code is interpreted. In Auto mode, this button starts the program.

The Abort button is used to terminate the execution of the currently running program. The Abort button of the Program Display window has the same effect as the F2 Abort button in the Functions menu.

The Restart button of the Program Display window is used to reset program execution to the beginning of the program and begin execution.

The Load button is used to bring a program file from disk into memory so that it can be run. When this button is selected, an Open Program File window is displayed. From this window, the operator selects the desired program (.PRG) file to be loaded. This button has the same effect as the F4 Load button of the Functions menu.

The Quit button is used to close the Program Display window.

4.5.1.4. The F5 Jog Button

The F5 Jog button is used to display the Jog window. The Jog window contains features that allow the operator to manually control one or more axes (without program intervention) by providing simple, discrete, override capabilities. In application, the F5 Jog button may be pressed during program execution, and an axis may be jogged (overridden) to a new position, and the program restarted. The operator can set two different speeds for jogging. These speeds are saved exiting the software in the .INI file. If the operator clicks and holds the jog buttons down, the chosen axis will move at the speed shown in the speed window. In the FreeRun mode, the axis will continue to move until the button is released. In the Index mode, the axis will move (as long as the button is down) the distance shown in the distance window. The Jog window is illustrated in Figure 4-29.



When F5 Jog is pressed during program execution, the program finishes any commands that are still in the program queue before activating the Jog screen. This action is different from that of the F3 Pause button that stops axis motion immediately.



If the F5 Jog button is selected, the Program Display window is closed and the Jog window is displayed. The Program Display window is displayed again after the Jog window is closed.

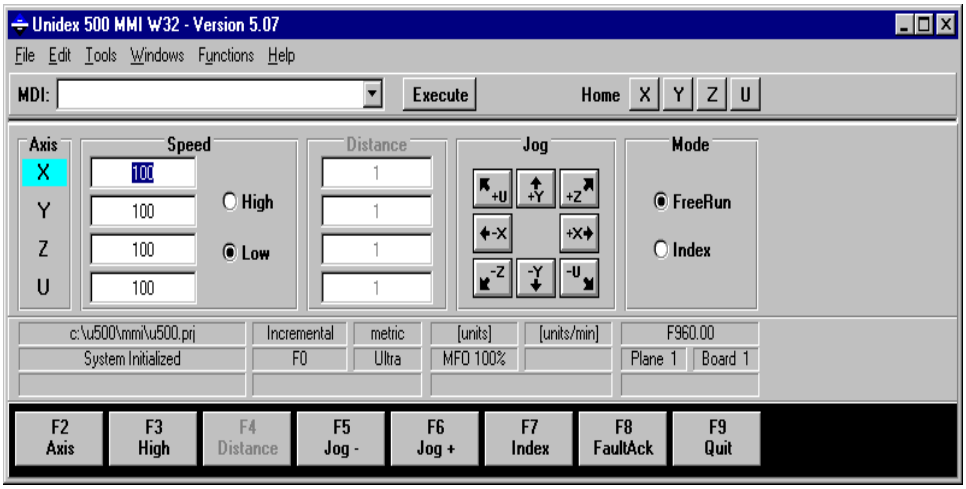


Figure 4-29. The F5 Jog Window

4.5.1.5. Password Protection

The MMI software provides the user the capability to password protect the levels of availability to other users. For example, one user may be allowed access to the Jog window upon system start up, while another user and his password is not. The Password window can be accessed in one of two ways. The user can hit the F6 Password function key located on the bottom of the Main screen, or select Password from the Edit pull-down menu.

Once F6 or Password has been selected the user will be prompted to enter a password in the System Password window (refer to Figure 4-30). The default password is "MASTERPW". Type in this password and hit "ENTER." After typing in the default password, the password editing options in Figure 4-31 will become visible.

The Access Level portion of the System Password window contains the list of check boxes that determine the access level of the password entered. Any item checked in this portion of the window allows the user access to those options when running the MMI software. An item that is deselected indicates this option is not available to the user with the corresponding password. Table 4-16 contains a list of the access level options and the functions affected.

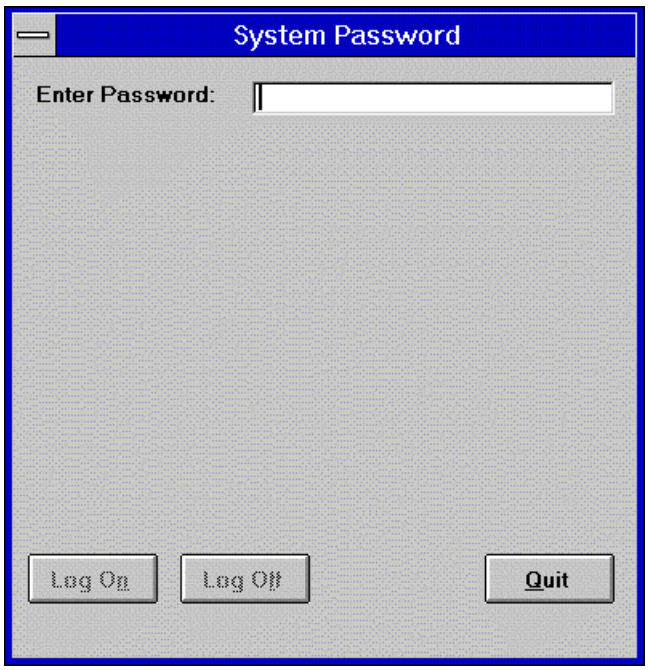


Figure 4-30. System Password Prompt Window



Figure 4-31. System Password Window

Table 4-16. Password Protected Access Level Options

Access Level	Options Protected
System Setup	The options affected are the menu items: Project, Parameters, and Configuration under both File and Edit menus. Also Firmware under the File menu and System Options under the Edit menu. These options are used to setup the UNIDEX 500 card.
Desktop Config	The options affected are the menu items: Functions, Status, Mdi, and MFO under the Windows menu. These options affect the window display.
Tuning Window	Determines whether the user has access to the Axis Tuning window and the Gain Adjust window.
Jog Window	Allows access to the Jog screen.
Program Execution	Permits execution of programs.
Program Edit	Permits editing of programs.
Diagnostics	Allowed access to the Diagnostics window.
Password Edit	Permits the user to add, delete, or edit options for all passwords.
Software Shutdown	Allows the termination of the software.

The "Require Password" check box, when checked, will prompt the user for a password when the software is started (see Figure 4-30).

The "Enter Password" dialog box is used to enter new or current passwords. To add a new password, type it in the "Enter Password" dialog box and click on the "Add" button. The new password will be entered in the list box on the right of the System Password window (see Figure 4-31). This list box will show all passwords available (limited to 8 passwords). After entering the password, select the appropriate check boxes to set the desired access level.

To enable the password entered or selected, hit the "Log On" Button. To change the current user, select "Log Off" to remove the current user. Then type in or select another password and hit "Log On." If the "Quit" button is selected no password will be enabled but the software will act as though all access levels were unchecked.

To delete a password, click on a password in the list box and then hit the "Delete" button.

4.5.1.6. The F7 Diag Button

The F7 Diag button is used to display a dynamic window of software and hardware status fields for each of the four axes (X, Y, Z, and U). The Diagnostics window displays axis positions, A/D input values, manual feed override, hardware/software faults, limits, traps, etc. The Diagnostics window is illustrated in Figure 4-8. This button has the same effect as the Diagnostics option of the Tools menu. For more information, refer to Section 4.4.1.: The Diagnostics Option of the Tools Menu on page 4-13.

4.5.1.7. The F8 Fault Ack Button

The F8 Fault Ack Button is used to acknowledge (that is, to clear) fault conditions in the Diagnostics screen. When a fault occurs, it is reflected in the Diagnostics screen typically as an asterisk (*). This fault display (*) will remain until the F8 Fault Ack button is selected.

4.5.1.8. The F9 Reset Button

The F9 Reset button is used to initialize the UNIDEX 500 system. Selecting this option has the same effect as selecting the Reset option of the File menu, selecting the Reset option from the Functions menu, or pressing the F9 function key on the keyboard.

Initialization is a process in which the current parameter (.PRM) file is sent to the memory of the U500 board from the PC. In addition, the actual software control program (the *firmware* file U500C.JWP) is also sent down to the U500 board. The last step of the initialization process is the restarting of the U500 firmware program.



Some menu commands and display features are not available if the U500 control board is not reset (i.e., initialized). An attempt to invoke such commands and/or features will cause a “System Not Initialized” popup box to be displayed.

4.5.2. The Status Option of the Windows Menu

The status bar is a window that is displayed at the bottom of the main software screen (above the function buttons if they are displayed). The status bar has 17 cells that contain status information about the system. The status bar is illustrated in Figure 4-32. Descriptions are given in Table 4-17. Double clicking on certain cells causes certain actions to occur. See Table 4-17 for a description of the actions taken on double clicking. For reference purposes, these cells are numbered 1 through 17 (starting with the upper left cell and continuing row by row to the lower right cell). Figure 4-33 indicates the numbering of the status cells in the status bar.

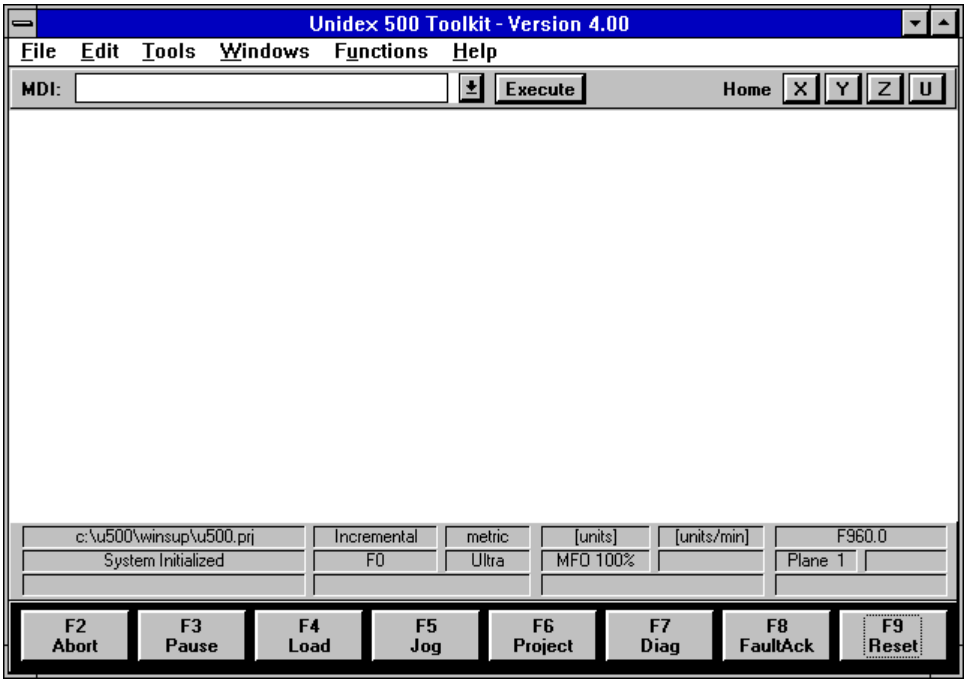


Figure 4-32. The Software Status Bar

Table 4-17. Cell Descriptions of the Status Bar

Status Cell #	Description/Display	Double Click on Cell Causes...
1	Current project file name (or blank)	Open new project popup window
2	Current programming mode	Toggles between incremental and absolute
3	English/Metric mode for current board	Toggles between English and Metric
4	Current display units (user units or machine steps)	Toggles between [units] and [steps]
5	Current feedrate units	Toggles between [units/min, units/sec] and [steps/min, steps/sec]
6	Current feedrate	*no action*
7	System status (initializing, faults, etc.)	*no action*
8	Hex code showing U500 status information. The data is the same as the result of the C function call WAPIAerReadStatus(5) (see chapter 10 for more details). Bit values of the data: bits 0-3 1=axis 1-4 enabled bits 4-7 1=axis 1-4 not in position bits 8-11 1=plane 1-4 comm. busy bits 12-15 1=plane 1-4 queue not empty bits 16-19 1=plane 1-4 on halt	*no action*
9	Board type (base, Plus or Ultra)	*no action*
10	MFO percentage (0% to 199%)	*no action*
11	Processing status (“Wait” or blank)	*no action*
12	Currently active plane (1-4)	*no action*
13	Board number for multiple U500s	*no action*
14	Current program file name (or blank)	*no action*
15	Defines contents of program window	Toggles between Program or Messages (available in MMI version only)
16	Joystick status (horz axis, vert axis, mode)	*no action*
17	Number of commands in queues 1 – 4. See Section 5.4.1. for more information about the U500 command queues.	*no action*

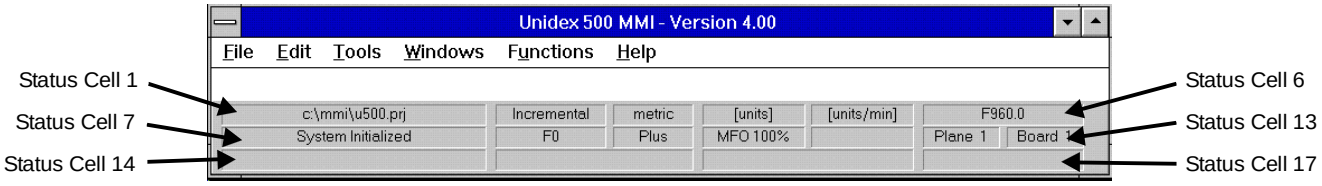


Figure 4-33. Status Cell Numbering

4.5.3. The Jog Option of the Windows Menu

See Section 4.5.1.4. for more information.

4.5.4. The Gain Adjust Option of the Windows Menu

The Gain Adjust option of the Windows menu is used to display the Gain Adjust popup. From this popup, the operator can view/change the values of the gain parameters (*Kpos*, *Ki*, *Kp*, *Vff*, and *Aff*) for any of the four available axes. These changes can be saved or previous values can be restored. The Gain Adjust window is illustrated in Figure 4-34.

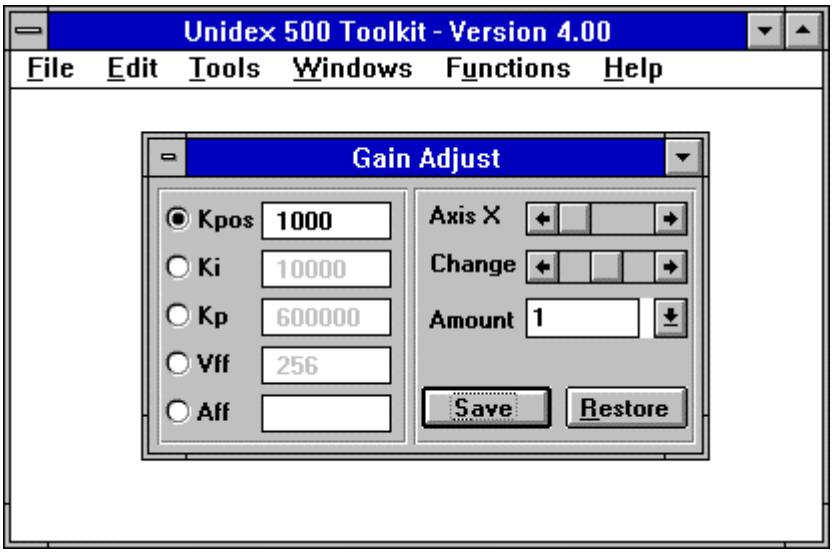


Figure 4-34. The Gain Adjust Popup

4.5.5. The MDI Option of the Windows Menu

MDI stands for Manual Data Interface. The MDI option of the Windows menu is used to display the MDI window. This window contains a small field into which immediate mode commands can be entered directly (rather than creating, loading, and running a program). The MDI window is illustrated in Figure 4-35.

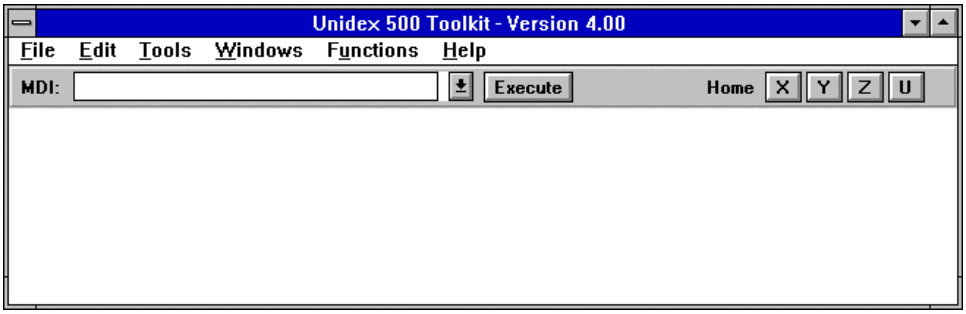


Figure 4-35. The MDI (Manual Data Interface) Window

4.5.6. The MFO Option of the Windows Menu

MFO stands for Manual Feedrate Override. The MFO option of the Windows menu is used to display the MFO Adjust popup. From this popup, the operator can manually change the current feedrate. This popup is illustrated in Figure 4-36.

The value displayed in the MFO Adjust popup is also displayed on the software Status menu (e.g., MFO 100%). The current feedrate is also displayed in the Status menu (e.g., F100.0).

In order for MFO adjustments to be made, parameter 002 (*MFO, 0 no MFO-pot, 1-255 pot offset*) must be set to a non-zero number.

If an external MFO potentiometer is being used, the MFO option should not be used. If the MFO Adjust popup is opened during the use of an MFO potentiometer, it will display the state of the MFO potentiometer when the popup was opened, not the current state (i.e., the MFO Adjust popup does not display dynamic MFO adjustments made from an MFO pot).

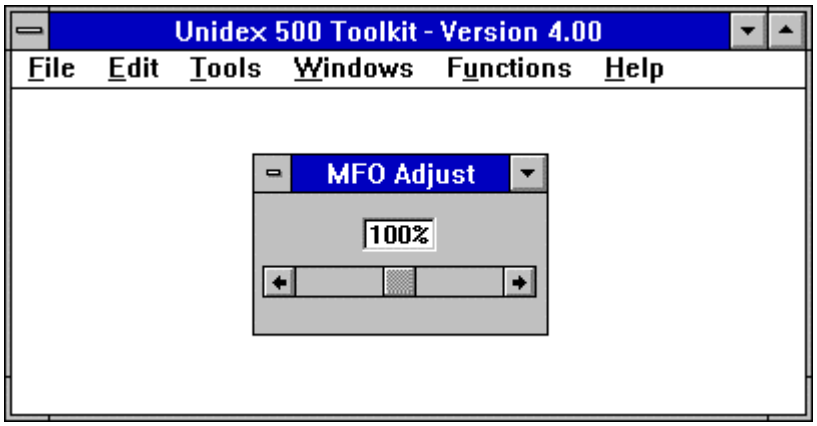


Figure 4-36. The MFO Adjust Popup

4.5.7. The Axis Display Option of the Windows Menu

The Axis Display option of the Windows menu is used to toggle the display of the axis menu as well as define the contents of the axis menu. When selected, this option presents a cascaded menu that lists seven display options as well as a toggle display option (Hide/Show). The Axis Display option is illustrated in Figure 4-37and the axis display is shown in Figure 4-38. The display options are listed and described in Table 4-18.

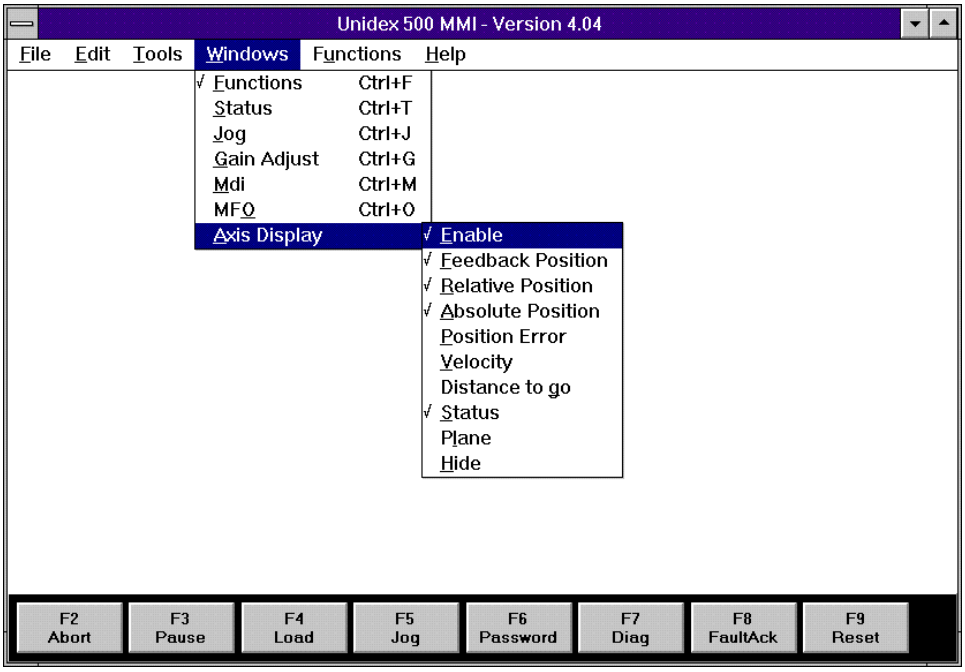


Figure 4-37. The Axis Display Menu with Options

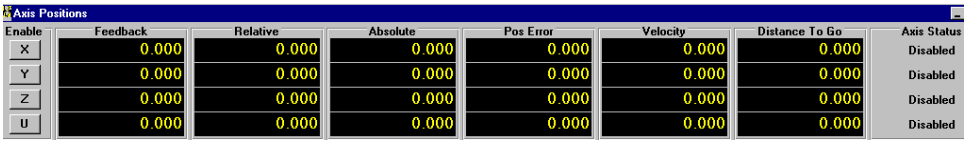


Figure 4-38. The Axis Display Window Showing Six Options

Table 4-18. Axis Display Options

Option	Description
Enable	Displays enable buttons (X, Y, Z and U) for each of the axes
Feedback Position	Displays the feedback position for each axis
Relative Position	Displays the relative position of each axis
Absolute Position	Displays the absolute position of each axis
Position Error	Displays the calculated position error for each axis (error = command position - feedback position)
Velocity	Displays the axis velocity
Distance to go	Displays the remaining distance to go in the move
Status	Displays the enable/disable status of each axis (amplifier)
Plane	Adds the appropriate plane number to the Enable axis buttons
Hide/Show	Toggles the display of the above options

4.6. The Functions Menu

The Functions menu of the main software screen contains the eight functions from the Windows / Functions option. These options correspond to the functions menu across the bottom of the software screen (when enabled) and the F2 through F9 function keys on the keyboard. The Functions menu is illustrated in Figure 4-39. Each of these options is discussed in Windows / Functions section (Section 4.5.1.: The Functions Option of the Windows Menu on page 4-35).

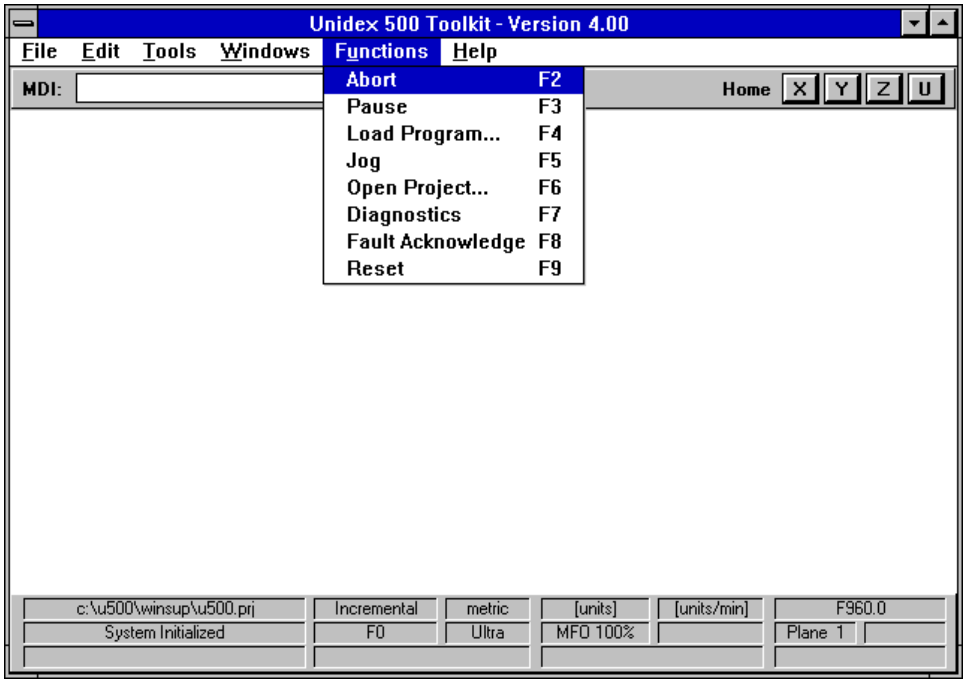


Figure 4-39. The Functions Menu

4.7. The Help Menu

The Help menu (Figure 4-40) has three options: Contents, Search for Help On…, and About…. The Contents and Search for Help On reference the U500 On-Line Help File: u500nw.hlp installed in the \u500\mmi\help subdirectory. The On-Line help file contains the help on all programming commands and information on the software interface. The About box (Figure 4-41) shows the current file information along with the software versions currently in use.

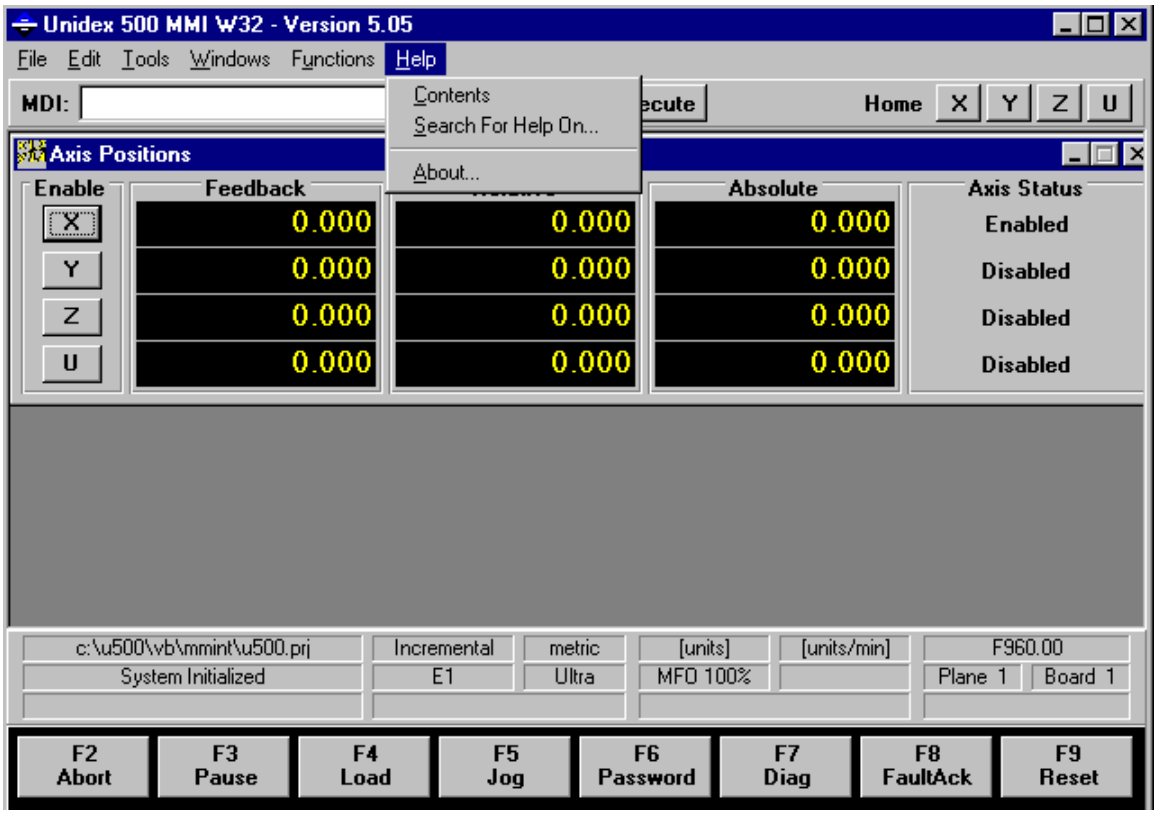


Figure 4-40. The Help Menu

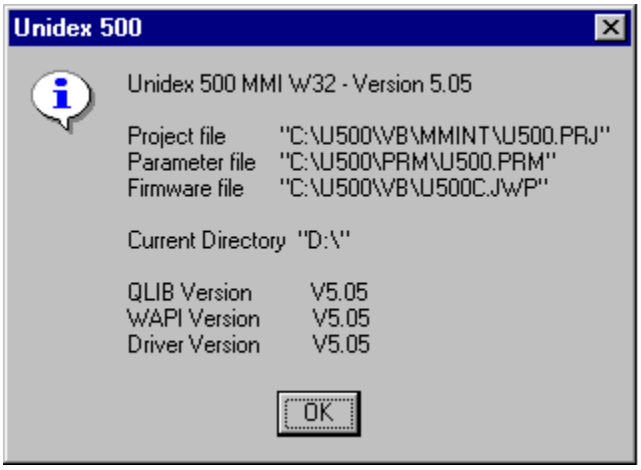


Figure 4-41. The About Popup Window

∇ ∇ ∇

CHAPTER 5: PARAMETERS

In This Section:

- Introduction 5-1
- The Edit Parameters Window 5-1
- The Position Tracking Tab (Tab 0) 5-5
- The Advanced Motion Tab (Tab 1) 5-13
- The Basic Motion Tab (Tab 2) 5-33
- The Homing and Limits Tab (Tab 3) 5-37
- The Traps Tab (Tab 4) 5-50
- The Motor and Feedback Configuration Tab (Tab 5) 5-62
- The Servo Loops Tab (Tab 6) 5-84
- The Other Tab (Tab 7) 5-94
- The Faults Tab (Tab 8) 5-106
- The General Tab 5-111
- The Axis Tab 5-112

5.1. Introduction

This chapter describes all of the parameters of the UNIDEX 500 system. Parameters are divided into 11 sections based on the tabs in the Edit Parameters window. Additional sections are included to explain topics that are closely related to the parameters.

For easy reference, a list of the 11 parameter sections (or tabs) is included in the outer margin of each parameter explanation. The appropriate section is highlighted and the parameter number is displayed below the list. An example is illustrated in the margin to the right.

Parameters are listed in tables in the respective sections. Default parameters are shown in bold for easy reference.

SAMPLE

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

001

5.2. The Edit Parameters Window

The software provides a user interface to the UNIDEX 500 parameters. This interface is called the Edit Parameters window and is activated through the Parameters... option of the Edit menu. This window is illustrated in Figure 5-1.

The Edit Parameters window contains the following components:

- Menu bar with File and Utilities menus
- Scrollable parameter display window
- Parameter number field
- Parameter value field
- Axis number radio buttons

These components are discussed briefly in the following sections and in detail in Chapter 4: The Software Interface.

5.2.1. The Menu Bar of the Edit Parameters Window

The menu bar of the Edit Parameters window is located across the top of the window. Two menu options (File and Utilities) are located on the menu bar. The File menu provides five options: Open, Save, Save As, Print... (by Tab Group or by Parameter Number), and Exit. The Utilities menu provides the ability to Transfer Parameters Between... (1) Axes or (2) Planes.

5.2.2. The Parameter Tabs

Below the menu bar of the Edit Parameters window are the parameter tabs. Each tab shows the name associated with a group of parameters. There are 11 parameter tabs that can be accessed from this window. The parameter tabs are:

- 0. Position Tracking
- 1. Advanced Motion
- 2. Basic Motions
- 3. Homing/Limits
- 4. Traps
- 5. Motor Feedback
- 6. Servo Loops
- 7. Other
- 8. Faults
- General
- Axis

Each of the numbered tabs contains a subset of the list of UNIDEX 500 parameters. All of the UNIDEX 500 parameters can be accessed through tabs 0 through 8 or through the General and Axis tabs. The General and Axis tabs provide a grouping that is more generalized than the nine specialized tabs. The General tab includes parameters numbered 001-099, 500, 501. The Axis tab contains parameters numbered 100-199, 200-299, 300-399, and 400-499 (abbreviated x00-x99, where x represents axis 1, 2, 3, or 4). The nine numbered tabs contain a variety of both general and axis parameters. The parameter tabs are illustrated in Figure 5-1.

5.2.3. The Scrollable Parameter Display Window

The scrollable parameter display window is located in the center of the Edit Parameters screen. This window contains the parameter names of the selected parameter tab. Initially, this window contains the first nine parameter names. Additional parameters can be viewed by using the scroll bar located on the right side of the window. Refer to Figure 5-1.

In order to select a particular parameter in the list, click the mouse on the parameter text (it may be necessary to use the scroll bar first). This causes the parameter text to be highlighted. The associated parameter number, parameter value, and axis number (as appropriate) are displayed in windows located below the display window.

5.2.4. The Parameter Number Field

Below the scrollable parameter display window is the parameter number field. This field contains the parameter number of the parameter selected in the display window.

Parameter numbers provide an additional way to differentiate parameters. Experienced operators may find parameter numbers easier to manipulate than their text counterparts. Refer to Figure 5-1.

The parameter number field may also be used to locate a particular parameter based on its number. To locate the name of a particular parameter, enter the parameter number in the parameter number field (and press <Enter>). If the parameter number is valid, the appropriate numbered tab is displayed and the requested parameter name is highlighted. If an invalid parameter number is requested, parameter 001 of tab 0 is displayed.

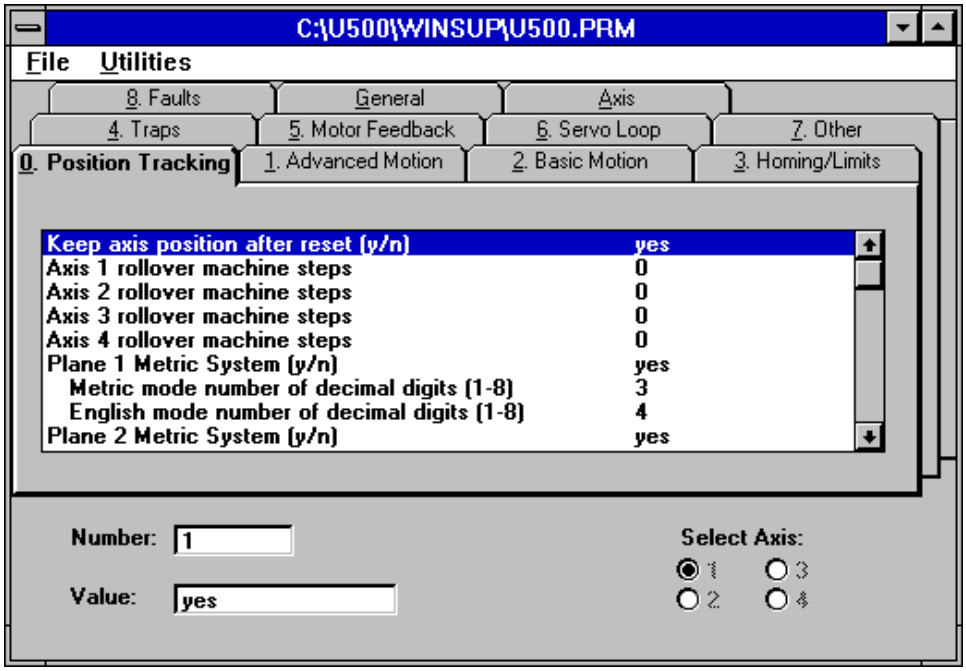


Figure 5-1. The Edit Parameters Window

5.2.5. The Parameter Value Field

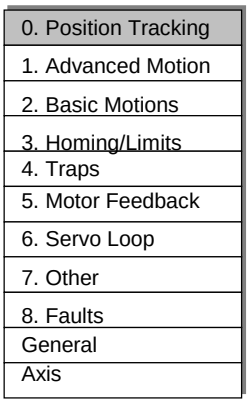
Below the parameter number field is the parameter value field. This field contains the current value of the parameter selected in the display window. The operator changes a parameter's value from this field. Refer to Figure 5-1.

To change the value of a parameter, the operator must first select the parameter. This can be done by using the parameter tabs (and scrolling through the appropriate list) or by entering the parameter number into the number field. When the desired parameter is selected, the operator must click the mouse in the value field and enter the new value on the keyboard, using the backspace key to erase unwanted values. When the desired value is displayed, the operator must press the <Enter> key to make the change. The default value of the parameter can be obtained by entering "d" into the value box and then hitting <Enter>.

5.2.6. Axis Number Radio Buttons

Axis number radio buttons are provided in the lower right corner of the Edit Parameters window. These buttons provide a method for quickly switching between axis parameters. Only one axis radio button can be selected at a time. For general parameters, the axis radio buttons are inactive (current radio button is selected).

5.3. The Position Tracking Tab

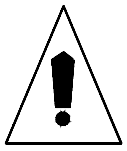


The position tracking and display parameters are used to define the scaling and resolution of the UNIDEX 500 system in either the Metric (the default) or English measurement systems. Configuration parameters for the optional PC-PSO board are also included. There are 19 parameters in the position tracking and display parameter group. These parameters are listed in Table 5-1 and explained in detail in this section.

Table 5-1. Position Tracking Parameters (Tab 0)

Param #	Description	Default Value
001	*Keep axis position after reset (y/n)*	No
011-014	*Axis n rollover machine steps* (*n*=1, 2, 3, or 4)	0
015	*PSO mailbox dual_port ram base address*	0x0D800
016	*PSO-HOST and PC interface address*	0x00
020, 038, 056, 074	*Plane n Metric System (y/n)* (*n*=1, 2, 3, or 4*)*	Yes
029, 047, 065, 083	*Metric mode number of decimal digits (1-8)* (planes 1-4)	3
030, 048, 066, 084	*English mode number of decimal digits (1-8)* (planes 1-4)	4

After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. This is done using the F9 RESET button located at the bottom of the main software window.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

001

5.3.1. Keep Axis Position after Reset (Y/N)

Parameter 001 configures the UNIDEX 500 to either clear (that is, set to 0) all absolute, relative and machine positions following a PC reset (no) or to retain the current axis values (yes).

A home cycle is often commanded as a normal startup function, in which case the setting of this parameter is immaterial. It is suggested, however, that this parameter be set to yes, as it may aid in recovering information and diagnosing problems.

This parameter can have one of two possible settings that are listed in Table 5-2.

Table 5-2. Settings for Parameter 001

Value	Function
Yes	Maintains position information following a PC reset
No	Clears position information following a PC reset (default)

This parameter defaults to “no.” This means that the position counter will clear the position information following a PC reset. This information is reflected in the Axis Positions window (refer to Figure 5-2).

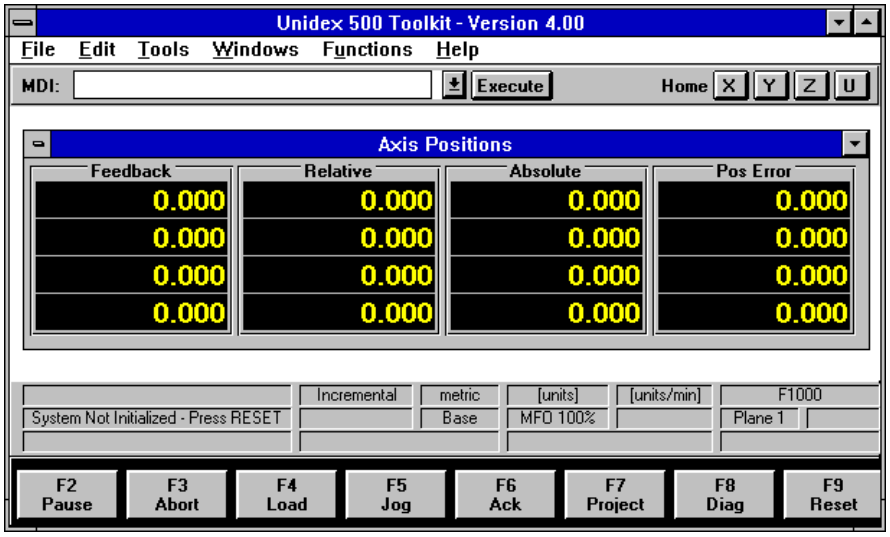


Figure 5-2. The Axis Positions Window With Position Information Cleared

IMPORTANT

U500 ISA: If the axis moves during the reset more than 16,384 counts for an encoder or 1/2 revolution for a resolver, the UNIDEX 500's position tracking registers (and the axis position display of the main software window) will not be accurate after the reset.

IMPORTANT

U500 PCI: Encoder movement during a hardware reset will result in a loss of position.

5.3.2. Axis *n* Rollover Machine Steps for Axes 1-4

The counter rollover parameters are used in rotary-type applications to define the number of machine steps associated with each *cycle*. Applications such as a rotary tables, for example, may require that you set a rollover point or a *modulo* distance. This allows the axis position to be read in units such as degrees. The sizes of the UNIDEX 500's absolute, relative and machine position registers are 47 bits plus one sign bit. This gives the position registers a range of 0 to 2^{47}-1. Parameters 011, 012, 013, and 014 correspond to axes 1 through 4, respectively.

For example, the position of a rotating machine part repeats every 360°. The actual servo position is of less importance than the machine's angular position. To provide the angular position, the position registers could be configured to rollover every 360° (see Figure 5-3). To do that, the following calculations must be made:

1. Determine the number of position counts that are in 360° of motion for the selected axis.
2. Set the appropriate rollover parameter to this value.

The range for these parameters is from a minimum of 0 (counter rollover feature is disabled) to a maximum value of 2^{47}-1. Counter rollover is applied to individual axes using parameters 011-014 (corresponding to axes 1-4, respectively). Counter rollover parameter settings are listed in Table 5-3.

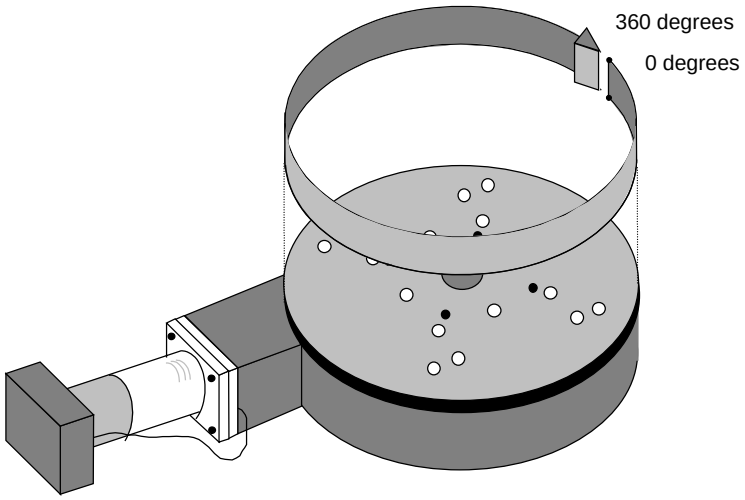


Figure 5-3. Axis *n* Rollover in Rotary Stage Application

Table 5-3. Settings for Parameters 011, 012, 013, and 014

Value	Function
0	Counter rollover is disabled for the associated axis (default)
1 to (2^{47} - 1)	Counter rollover specified in machine steps

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

015

5.3.3. PSO Mailbox Dual_port RAM Base Address

The UNIDEX 500 has several optional accessories that can be used to augment the operation of the system. One such option is the PC-PSO board (refer to Chapter 1 for more information on the **P**osition **S**ynchronized **O**utput board). If the PSO option is used with the UNIDEX 500 ISA system, then certain parameters of the software must be configured to permit proper communication between the two PC boards. A base address for each port must be selected by configuring two banks of jumpers on the PC-PSO board. These addresses must then be used in all software communications with the card. System parameters 015 specifies (in hexadecimal) the base address of the dual-ported RAM.

Some commonly used addresses for DPRAM are listed in Table 5-4. By default, the PC-PSO is configured to use DPRAM base address 0x0D800.

Table 5-4. Commonly Used Settings for the DPRAM Base Address

DPRAM Base Address	Description
0x08000	PC-PSO dual-ported RAM base address is 8000:0000
0x0C000	PC-PSO dual-ported RAM base address is C000:0000
0x0D000	PC-PSO dual-ported RAM base address is D000:0000
0x0D800	PC-PSO dual-ported RAM base address is D800:0000 (default)



The dual PC-PSO hardware decodes 4 K (4096) byte blocks of system memory. The dual-ported RAM size is 2 K (2048) bytes. Therefore, it appears twice within the 4 Kbyte memory block. The default jumper settings of the PC-PSO board map the dual-ported RAM to memory locations D800:0000 and D800:0800. The first 2 Kbyte block should be used for programming consistency.



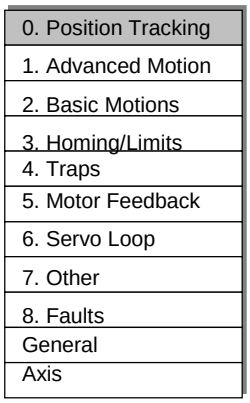
This parameter is to be used for the U500 ISA Only. The U500 ISA must communicate with the PC-PSO card to implement PSO functions. The PSO functions for the U500 PCI do not require additional hardware.

It may be necessary for the CMOS of the system PC to be reconfigured to allow communication with addresses within the first 1 MB of memory. Typically, this is done from the setup utility of the PC after rebooting the PC. Refer to the PC instruction manual for information on how to enter the setup program to change CMOS settings on the PC.



If UNIDEX 500 parameter 015 (*PSO mailbox dual_port RAM base address*) is set to 0 (the software default that indicates that a PC-PSO board is not used), the PC-PSO will not initialize. This parameter must be set to a unique address for proper operation and must agree with the hardware address setting specified by jumpers JP16-JP23 of the PC-PSO board. Refer to the *PC-PSO Manual (P/N EDO105)* for more information.

5.3.4. PSO-HOST and PC Interface Address (U500 ISA - 16 bit MMI software only)



016

The UNIDEX 500 has several optional accessories that can be used to augment the operation of the system. One such option is the PSO-PC board. If the PSO option is used with the UNIDEX 500 system, the PSO base address must be configured to permit proper communications between the two PC boards. A base address for each port must be selected by configuring two banks of jumpers on the PSO-PC board. These addresses must then be used in all software communications with the card. When using the 16 bit UNIDEX 500 MMI, parameter 016 specifies (in hexadecimal) the base host address. When using the 32 bit UNIDEX 500 MMI, this parameter is ignored. Instead, the base address information is stored in the Windows registry. The "U500 Registry Editor" program u500reg.exe) can be used to set up the PSO base address. Parameter number 016 specifies (in hexadecimal) the host base address for use with the UNIDEX 500 motion controller. The host interface is used to download system firmware and is the main communications channel to the PC. It is mapped in the PC's input/output (I/O) space. This mapping is accomplished using the host base-address jumper settings on the PC-PSO board. Some commonly used host addresses are listed in Table 5-5. Host base address 0x310 is used as the default for parameter 016.

Table 5-5. Commonly Used Settings for the Host Base Address Parameter 016

Host Base Address	Description
0x0	PC-PSO board not used (default)
0x200	PC-PSO host base address is 0x200
0x210	PC-PSO host base address is 0x210
0x300	PC-PSO host base address is 0x300
0x310	PC-PSO host base address is 0x310 (default)
0x350	PC-PSO host base address is 0x350
0x360	PC-PSO host base address is 0x360

If you are using the 16 bit UNIDEX 500 software and parameter 016 (PSO Host and PC Interface Base Address) is set to 0 (the software default which indicates that a PC-PSO board is not used), the PC-PSO will not initialize. This parameter must be set to a unique address for proper operation and must agree with the hardware address setting specified by jumpers JP2-JP7 on the PC-PSO board. For additional information, refer to the PC-PSO manual (EDO105)



This parameter is to be used for the U500 ISA Only. The U500 ISA must communicate with the PC-PSO card to implement PSO functions. The PSO functions for the U500 PCI do not require additional hardware.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

020
038
056
074

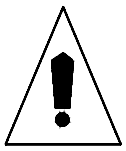
5.3.5. Plane *n* Metric System (Y/N)

These parameters specify the default use of the Metric or English measurement systems when programming the conversion factor for contour planes 1-4, respectively. The conversion factor is used to determine system scaling. The actual values for the conversion factors are programmed in parameters x00 (Metric) and x01 (English). Parameters 020, 038, 056, and 074 only specify the measurement system.

This parameter can have one of two possible values (yes or no) and is programmed on a per plane basis. Settings for parameters 020, 038, 056, and 074 are listed and described in Table 5-6.

Table 5-6. Settings for Parameters 020, 038, 056, and 074

Value	Function
Yes	Metric system *is* used for the associated contour plane (default)
No	Metric system *is not* used (i.e., English System is used) for the associated contour plane



Be sure to set parameters 029, 047, 065, and 083 (*Metric mode number of decimal digits*) or parameters 030, 048, 066, and 084 (*English mode number of decimal digits*) as appropriate, for each of the active contour planes. Refer to Table 5-7 for a reference of these associations.

Table 5-7. Parameter Associations between Planes, Measurement Units, and the Number of Decimal Digits

Plane #	English/Metric Units	Number of Decimal Digits
1	English (parameter 020=no)	Use parameter 030
	Metric (parameter 020=yes)	Use parameter 029
2	English (parameter 038=no)	Use parameter 048
	Metric (parameter 038=yes)	Use parameter 047
3	English (parameter 056=no)	Use parameter 066
	Metric (parameter 056=yes)	Use parameter 065
4	English (parameter 074=no)	Use parameter 084
	Metric (parameter 074=yes)	Use parameter 083



For information on determining an appropriate conversion factor, refer to parameter x00 (*Metric conversion factor*) or parameter x01 (*English conversion factor*).

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

029
047
065
083

5.3.6. Metric Mode Number of Decimal Digits (1-8)

These parameters set the number of zeros that are added after the decimal place in Metric mode displays. They are used in conjunction with parameter x00 (*Metric conversion factor*) to determine system scaling (i.e., the number of machine steps in relation to program steps). Parameters 029, 047, 065, and 083 correspond to contour planes 1 through 4, respectively. These parameters must be configured for each of the active contour planes.

These parameters can have values that range from 1-8 decimal places. Examples are shown in Table 5-8.

Table 5-8. Settings for Parameters 029, 047, 065, and 083

Value	Example	Function
1	123.1	Allots 1 decimal place after the decimal point
2	123.12	Allots 2 decimal places after the decimal point
3	123.123	Allots 3 decimal places after the decimal point (default)
4	123.1234	Allots 4 decimal places after the decimal point
5	123.12345	Allots 5 decimal places after the decimal point
6	123.123456	Allots 6 decimal places after the decimal point
7	123.1234567	Allots 7 decimal places after the decimal point
8	123.12345678	Allots 8 decimal places after the decimal point



For information on determining an appropriate metric conversion factor, refer to parameter x00 (Metric conversion factor).

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

030
048
066
084

5.3.7. English Mode Number of Decimal Digits (1-8)

These parameters set the number of zeros that are added after the decimal place in English mode displays. They are used in conjunction with parameter x01 (*English conversion factor*) to determine system scaling (i.e., the number of machine steps in relation to program steps). Parameters 030, 048, 066, and 084 correspond to contour planes 1 through 4, respectively. These parameters must be configured for each of the active contour planes.

These parameters can have values that range from 1-8 decimal places. Examples are shown in Table 5-9.

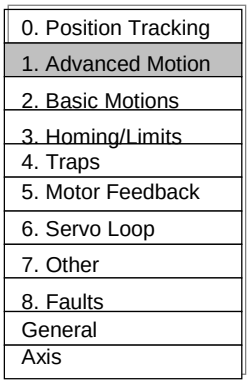
Table 5-9. Settings for Parameters 030, 048, 066, and 084

Value	Example	Function
1	123.1	Allots 1 decimal place after the decimal point
2	123.12	Allots 2 decimal places after the decimal point
3	123.123	Allots 3 decimal places after the decimal point
4	123.1234	Allots 4 decimal places after the decimal point (default)
5	123.12345	Allots 5 decimal places after the decimal point
6	123.123456	Allots 6 decimal places after the decimal point
7	123.1234567	Allots 7 decimal places after the decimal point
8	123.12345678	Allots 8 decimal places after the decimal point



For information on determining an appropriate English conversion factor, refer to parameter x01 (*English conversion factor*).

5.4. The Advanced Motion Tab



The advanced motion tab contains parameters that configure contour planes and gantry (master/slave) motion. These parameters are listed in Table 5-10 and explained in detail below. An overview of planes is presented, followed by descriptions of the advanced motion parameters.

Table 5-10. Advanced Motion Parameters (Tab 1)

Param #	Description	Default Value
000	*Number of contour planes (1,2,4)*	1
003	*Axis 1 map to plane (0, 1-4) as (XYZU)*	1,X
004	*Axis 2 map to plane (0, 1-4) as (XYZU)*	1,Y
005	*Axis 3 map to plane (0, 1-4) as (XYZU)*	1,Z
006	*Axis 4 map to plane (0, 1-4) as (XYZU)*	1,U
007	*Axis 1 gantry (y/none), slave (2,3,4)*	None
008	*Axis 2 gantry (y/none), slave (1,3,4)*	None
009	*Axis 3 gantry (y/none), slave (1,2,4)*	None
010	*Axis 4 gantry (y/none), slave (1,2,3)*	None
018, 036, 054, 072	*Plane n Indexing segment time (1-20ms)* (*n*=1, 2, 3, or 4)	10
021, 039, 057, 075	*Linear type ac/de (y/n) for contour plane n* (*n*=1, 2, 3, or 4)	No
027, 045, 063, 081	*Clamp feedrate (prog. steps/ms)* for plane *n* (*n*=1, 2, 3, or 4)	256.0
028, 046, 064, 082	*Corner rounding non-ramp time (1-32000 ms)* for plane *n* (*n*=1, 2, 3, or 4)	150
031,049, 067, 085	*Contouring mode* for plane *n* (*n*=1, 2, 3, or 4)	0
090, 092, 094, 096	*A/D channel n joystick deadband* (*n*=1, 2, 3, 4)	0
091, 093, 095, 097	*A/D channel n joystick center position* (*n* = 1, 2, 3, 4)	0
098	*Safe zone output bit (0, 1-8)*	0
099	*Option board setup code*	0
500	*User interrupt setup code*	0
501	*Abort on input high (0, 1-16)*	0

After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. This is done using the F9 RESET button located at the bottom of the main software window.



5.4.1. Overview of Planes

The UNIDEX 500 system can control up to four axes of motion as well as miscellaneous inputs and outputs. Typically, these inputs, outputs, and/or axes are controlled from a program that is written for the particular application. A UNIDEX 500 program consists of a series of instructions that are executed sequentially to perform the desired functions.

The programming/control process starts when a program is written using a set of UNIDEX 500 programming commands. When the operator starts the program, the first chunk is loaded from RAM on the PC into a program buffer (8 Kbytes on the U500) for execution (using the *aer_send* command). Execution of the program starts with the first command in the buffer. After the first command is finished executing, the command is removed from the buffer. This process continues until the entire program has been queued into the program buffer and has finished executing. This programming scheme is ideal for controlling a single multi-axis system through a series of discrete steps. Refer to Figure 5-4.

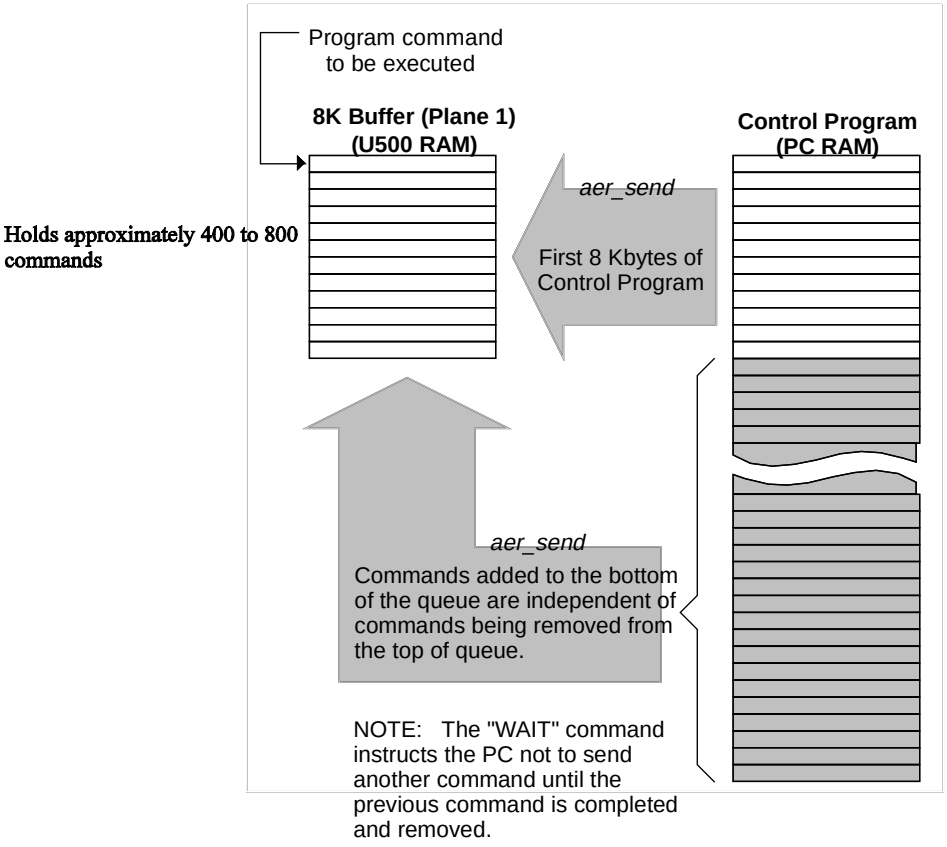


Figure 5-4. Programming Control Using a Single Plane

The versatility of the UNIDEX 500 system provides a second, more powerful programming scheme that allows multiple "programs" (two or four) to perform independent control functions asynchronously and "simultaneously." This programming scheme multitasks between a user-defined number of sections (1, 2, or 4) of the original 8 Kbyte program buffer. Each of these sections is called a *plane* or queue.

A plane is a program buffer (of fixed size) that contains programming statements. One, two, or four planes may be defined, each containing programming statements unique to that plane. The UNIDEX 500 executes the first (and then subsequent) lines in each plane in a round-robin fashion called multitasking. The multitasking is performed so quickly that the planes appear to be executing simultaneously. Refer to Figure 5-5.

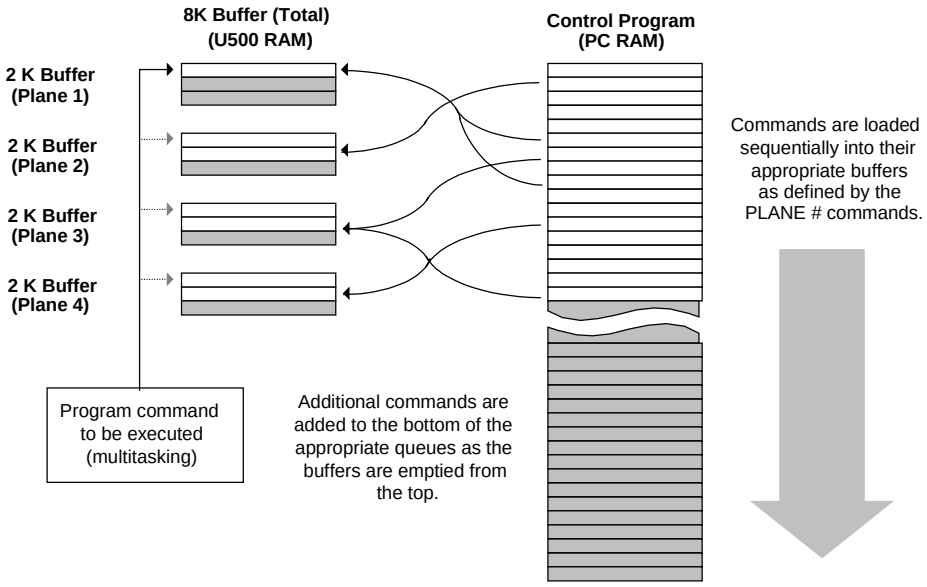


Figure 5-5. Programming Control Using Four Planes

Below are important issues to remember when using planes.

- The programming buffer is 8 Kbytes in size
- The U500 system can be configured to use 1, 2, or 4 planes
- The 8 Kbyte program buffer is divided equally among the planes being used (i.e., there will be one 8 Kbyte buffer, two 4 Kbyte buffers, or four 2 Kbyte buffers)
- A plane can be used for non-axis control such as setting outputs
- Special programming commands are used to route individual program statements to their appropriate planes (buffers)
- When a plane buffer is filled with commands, no additional commands to that plane can be queued until a slot in the queue is freed (i.e., the oldest command is completed and removed from the buffer). If this occurs, and the next sequential command to be loaded into the program buffer is directed to that plane, then the other planes must wait

A sample UNIDEX 500 program segment is listed in Figure 5-6. Although programming has not yet been explained, the program segment shown below is relatively simplistic and offers an example of the use of planes. Comments are preceded by semicolons and are listed to the right of programming commands for clarity.

```
; This program segment illustrates the use of planes in programming. This program
;segment assumes the following parameter settings have been established:
;       000 =   2               ;Number of contour planes is set to 2
;       003 =   1,X             ;Axis 1 [param 003] is mapped to plane 1 as "X"
;       004 =   2,Y             ;Axis 2 [param 004] is mapped to plane 2 as "Y"
;       005 =   2,Z             ;Axis 3 [param 005] is mapped to plane 2 as "Z"
;       006 =   1,U             ;Axis 4 [param 006] is mapped to plane 1 as "U"
;-------------------------------------------------------------------------------------------
            :                                       :
        PLANE 1                 ;Select plane 1
        EN  X   U               ;Enable axes X and U (of plane 1) for motion
        PLANE 2                 ;Select plane 2
        EN  Y   Z               ;Enable axes Y and Z (of plane 2) for motion
        G1  Y   50000           ;Linear move Y axis (in plane 2) 50000 machine steps
        PLANE 1                 ;Select plane 2 (Y axis is still moving...)
        G1  X   10000           ;Linear move X axis (in plane 1) 10000 machine steps
        G1  U   500             ;Linear move U axis (in plane 1) 500 machine steps
            :                                       :
```

Figure 5-6. Sample Programming Segment Showing the Use of Planes

Using the parameter settings shown above, the UNIDEX 500 assigns two internal buffers for plane commands. The 8 Kbyte program buffer (located on the UNIDEX 500 board) is therefore divided into two equal buffers of 4 Kbytes each. Next, the operator loads the program into PC memory (from disk, for example). When the operator starts the program, the software begins sending the program (one line at a time) to the appropriate program buffer in the UNIDEX 500 board using a special software command (the *aer_send command*). The software continues to send commands to the appropriate buffers until either a target buffer is filled, or until the program finishes.

On the UNIDEX 500 side, the multitasking program is checking each of the buffers (in a round-robin fashion) and executing the next appropriate instruction in each. When an instruction in a buffer is completed, the instruction is removed from the buffer, remaining instructions are shifted up, and a slot in the buffer is freed for additional programming statements from the software.

Notice that axis Y (in Figure 5-6) is given a linear move (G1) command of 50,000 units (in plane 2). This is followed by a linear X-axis move of 10,000 units (in plane 1). By using planes and multitasking, the UNIDEX 500 is able to carry out the request in plane 1 before the request in plane 2 is finished (i.e., before the Y-axis completes its 50,000-unit linear move).

For more information about programming using planes, refer to Chapter 7: Programming.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

000

5.4.2. Number of Contour Planes (1, 2, 4)

This parameter defines the number of contour planes through which the UNIDEX 500 will multitask. Each contour plane is assigned its own memory area, which holds program commands that are exclusively targeted for that plane.

The program buffer of the UNIDEX 500 is fixed at 8 Kbytes regardless of the number of contour planes selected. One, two, or four planes may be used for maximum flexibility and efficiency. If one plane is specified, the size of the program buffer is fixed at 8 Kbytes. If two planes are selected, the size of each program buffer is fixed at 4 Kbytes. Finally, if four planes are selected, the size of each program buffer is fixed at 2 Kbytes.

In a single plane configuration (parameter 000=1), the UNIDEX 500 will wait for one command to finish before beginning to execute the next command. For example, if an axis is commanded to move 300 mm, the UNIDEX 500 will wait until that position is reached before the next command is interpreted and executed. In this configuration, the entire 8 Kbyte program buffer is available to the control program. If the control program is larger than 8 Kbytes, portions are queued into the buffer as preceding commands are executed and removed from buffer.

In a multiple plane configuration (parameter 000=2 or 4), commands are queued to particular program buffers (i.e., planes). The first program statement in each of these buffers is executed in a multitasking environment so that the tasks appear to run concurrently. In these configurations, the 8 Kbyte program buffer is divided equally among the planes. As commands are read from the control program, they are sent to the appropriate buffers.

If a program buffer (i.e., a plane) becomes filled, and additional program commands for that plane are forthcoming, the plane is marked as “busy” until a command is completed and removed from the queue.



For additional information about planes, refer to the previous section (Overview of Planes). For additional information on mapping axes to planes, refer to parameters 003, 004, 005, and 006.

Analyzing the application and the best programming style for that application is vital to optimizing the number of programming planes required. For non-multitasking applications, one contour plane is suitable. This is the typical operating mode for most controllers and is the default setting for this parameter. In addition, all axes are assigned to contour plane one by default. However, many applications require independent, asynchronous motions with real time responses. In these cases, it is better to define multiple contour planes.

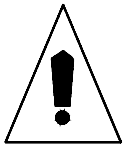
This parameter can have the value 1, 2, or 4. This corresponds to either 1 (the default), 2, or 4 contour planes. Settings for parameter 000 are shown in Table 5-11.

Table 5-11. Settings for Parameter 000

Value	Number of Program Buffers	Size of Each Program Buffer
1	1 (default)	8 Kbytes
2	2	4 Kbytes
4	4	2 Kbytes



It is suggested that the number of contour planes set by this parameter be as small as possible for the application. Doing this will provide the maximum buffer size and the fastest processing time.



IMPORTANT

Following configuration of this parameter, the system must be reinitialized so that the new number of planes is recognized.

5.4.3. Axis *n* Map to Plane {1-4} as {X, Y, Z, U}

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

003
004
005
006

The UNIDEX 500 is capable of multitasking at such high speeds, that it appears as though they are being performed simultaneously. The execution of these tasks, as well as how the tasks relate to each other is programmable by the operator. Four memory areas are available to receive motion commands. These memory areas are referred to as contour planes.

These parameters also are used to map one or more axes to a contour plane and assign the axes a programming name (i.e., X, Y, Z, or U).

a,*b* where:

a = the plane number {1, 2, 3, or 4} to which the respective axis is mapped

b = the axis designator {X, Y, Z, or, U} (e.g., 003=4, X maps axis 1 to plane 4 as "X")

The defaults settings for these parameters are (1,X), (1,Y), (1,Z), and (1,U) (i.e., axes 1,2,3, and 4 are all mapped to plane 1 as X, Y, Z, and U, respectively). This is summarized in Table 5-12.

Table 5-12. Settings for Parameters 003, 004, 005, and 006

Param #	Axis #	Value	Description	Examples
003	1	*a*,*b*	Map axis 1 to plane *a* as *b*, where *a* = {1, 2, 3, or 4}, and *b* = {X, Y, Z, or U}	**1,X** (default) 2,U 3,Z
004	2	*a*,*b*	Map axis 2 to plane *a* as *b*, where *a* = {1, 2, 3, or 4}, and *b* = {X, Y, Z, or U}	**1,Y** (default) 1,Z 2,X
005	3	*a*,*b*	Map axis 3 to plane *a* as *b*, where *a* = {1, 2, 3, or 4}, and *b* = {X, Y, Z, or U}	**1,Z** (default) 4,X 3,Y
006	4	*a*,*b*	Map axis 4 to plane *a* as *b*, where *a* = {1, 2, 3, or 4}, and *b* = {X, Y, Z, or U}	**1,U** (default) 4,Y 1,Z

Axis one is always assigned to amplifier/drive channel one, axis two to amplifier/ drive channel two, etc.



An axis must not be assigned to more than one contour plane. If the UNIDEX 500 system is inadvertently configured this way, a feedback error is generated.



For additional information on the use of planes, refer to Section 5.4.1: Overview of Planes, as well as, the PLane and PLC commands in Chapter 7: Programming.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

007
008
009
010

5.4.4. Axis *n* Gantry {Y/None}, Slave {1, 2, 3, 4}

When performing contour-type moves, it may be desirable to pair axes in a master/slave relationship. In such configurations, motions commanded to the master axis are automatically sent to the slave axis.

These parameters must be configured relative to each other. In addition, an axis may be designated as a master or a slave, but not both (i.e., parameter 007 cannot designate axis 2 as a slave and parameter 008 designate axis 2 as a master).

The syntax of parameters 007-010 can have one of two possible formats:

n which means the associated axis is not a gantry master, or

y,*b* which means the associated axis is a gantry master having axis *b* (one of the remaining 3 axes) as the slave axis (e.g., 007=y,2 sets axis 1 as the gantry master with slave axis 2). The *b* can also be preceded by a negative sign (-). This will invert the direction of motion of the slave with respect to the master.

The default settings for these parameters is "none" (i.e., axes 1,2,3, and 4 are not gantry masters). Other settings for these parameters are summarized in Table 5-13.

Table 5-13. Settings for Parameters 007, 008, 009, and 010

Param #	Axis #	Value	Description	Examples
007	1	None	Axis 1 is not a gantry master	**None** (default)
		y,*b*	Axis 1 is a gantry master having axis *b* as the slave axis, where: *b* = {2, 3, or 4}	Y,2 Y,-2 Y,3 Y,-3 Y,4 Y,-4
008	2	None	Axis 2 is not a gantry master	**None** (default)
		y,*b*	Axis 2 is a gantry master having axis *b* as the slave axis, where: *b* = {1, 3, or 4}	Y,1 Y,-1 Y,3 Y,-3 Y,4 Y,-4
009	3	None	Axis 3 is not a gantry master	**None** (default)
		y,*b*	Axis 3 is a gantry master having axis *b* as the slave axis, where: *b* = {1, 2, or 4}	Y,1 Y,-1 Y,2 Y,-2 Y,4 Y,-4
010	4	None	Axis 4 is not a gantry master	**None** (default)
		y,*b*	Axis 4 is a gantry master having axis *b* as the slave axis, where: *b* = {1, 2, or 3}	Y,1 Y,-1 Y,2 Y,-2 Y,3 Y,-3



Once the master is enabled, the slave is automatically enabled.

By default, gantry axes are linked together at all times except for home cycles. The home cycles are done independently except during the marker search. Each axis does an independent search for its marker (for encoders) or null (for resolvers). When one axis finds its marker, it will wait for the other to complete its marker search. The home cycle is not complete until both the master and the slave axes finish their move.

In this case, the home cycle parameters for the gantry axes should be set the same. Gantry alignment can be adjusted by changing the *Home switch to marker (machine steps)* (x07) and *Home offset (machine steps)* (x06) parameters. If individual adjustment is not desired, the master axis' limits and marker signals should be connected to the slave's channel.

However, the gantry feedback channel and home cycle type can be affected by parameter x38, the Primary Position Feedback Channel parameter. Please refer to Section 5.8.3., for more information.

It is desirable to have identical stages oriented in the same direction. The limit switches and markers should also be aligned as close as possible.



For resolver gantry systems, each axis determines the distance it must move to go to the resolver's absolute zero position. The setting of parameter x06 (*Home offset*) is added to this distance. The move is then executed at the feedrate specified by parameter x05 (*Normal home feedrate*). The home cycle is not complete until both the master and slave axes finish their move.

For example, consider a gantry system in which the slave axis marker is "d" machine steps from the master axis' marker. (In the case of a resolver, the marker position is replaced by the zero or "null" position.) The alignment of the axes can be adjusted by changing the master's *Home offset* parameter value to reflect the distance "d." The slave axis should have an offset setting of "0." Refer to Figure 5-7.

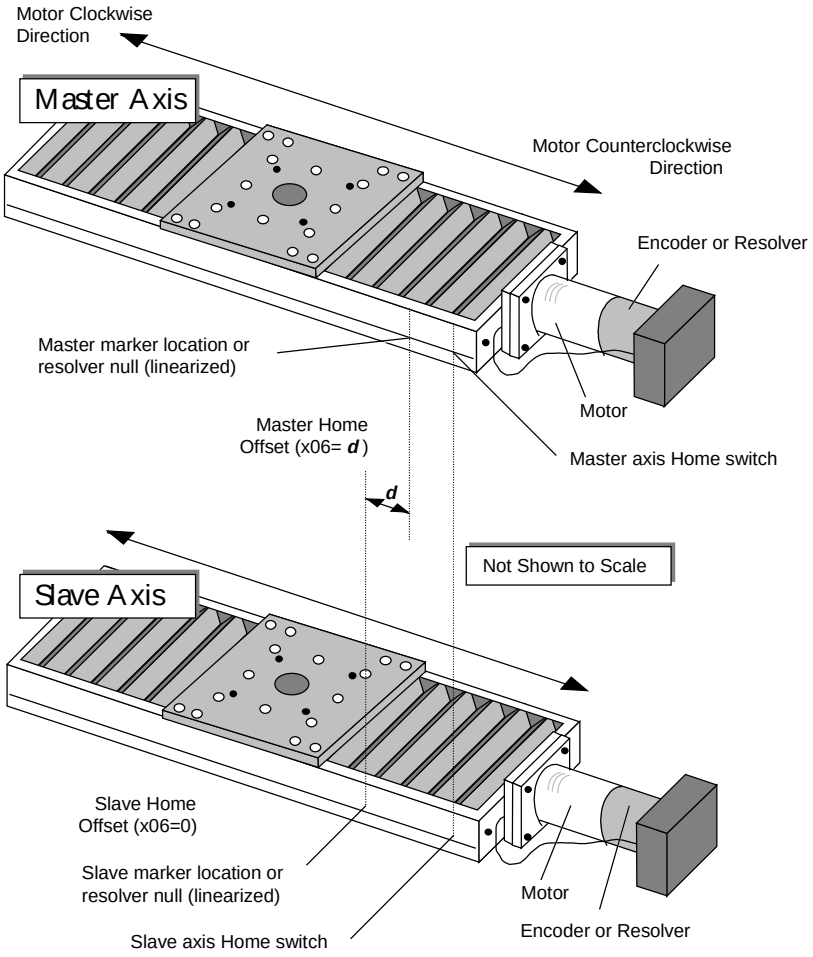


Figure 5-7. Using *Home Offset* Parameter to Keep Gantry Aligned After Homing

IMPORTANT

The following servo-related parameters should be set to the same values for both axes involved in the gantry:

x02 - *Home direction is CCW (y/n)*
x04 - *Power-on home feedrate (machine steps/ms)*
x05 - *Normal home feedrate (machine steps/ms)*
x16 - *Maximum ac/de (machine steps/ms^2)*

5.4.5. Plane *n* Indexing Segment Time (1-20 ms)

During trajectory generation, the UNIDEX 500 divides the motion into segments. This parameter represents the motion time for each segment (in milliseconds [ms]). The default setting of 10 ms is sufficient for most applications. If the application requires many short moves with short ramp times, the operator may wish to reduce the value of this setting. The minimum value is one millisecond.

These parameter values can range from 1 to 20 ms for each of the planes under command. The system default is 10 ms. Refer to Table 5-14.

Table 5-14. Settings for Parameters 018, 036, 054, and 072

Param #	Plane #	Range	Examples
018	1	1-20 ms	1 = Provides the slowest calculation time and yields the maximum number of indexing segments 10 = Provides a moderate calculation time and yields a moderate number of indexing segments (default) 20 = Provides the fastest calculation time and yields the minimum number of indexing segments
036	2	1-20 ms	*See examples shown above*
054	3	1-20 ms	*See examples shown above*
072	4	1-20 ms	*See examples shown above*

This parameter will not increase servo velocities in any way. It may be used to improve the processing efficiency of the calculation effort during trajectory generation.



021
039
057
075

5.4.6. Linear Type Ac/De (Y/N)

The UNIDEX 500 supports two types of acceleration/deceleration (ac/de) ramping trajectories: linear and sine. Each contour plane must be delineated as either linear or sine for this ramping. The strict form of linear ramping may be replaced by the smoother sine ramping option to reduce "jerky" motion during axis acceleration/deceleration. Refer to Figure 5-8.

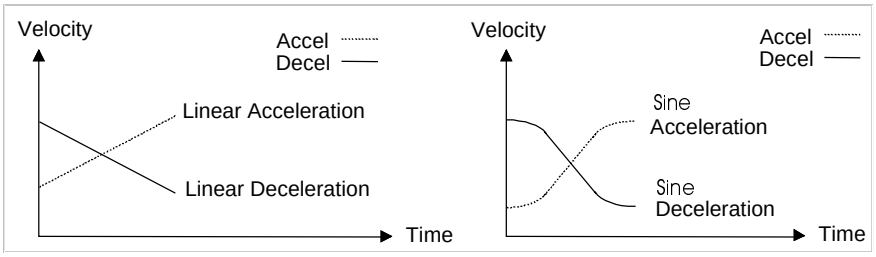


Figure 5-8. Graphs of Linear and Sine Ramping Trajectories

Each of these parameters can have either a "yes" or "no" value, where "yes" indicates that linear acceleration/deceleration is defined for the associated plane, and "no" (the default value) indicates that sine acceleration/deceleration is defined for the associated plane. These values are summarized in Table 5-15.

Table 5-15. Settings for Parameters 021, 039, 057, and 075

Param #	Plane #	Descriptions
021	1	Yes = Accel/decel is linear for plane 1 No = Accel/decel is sine type for axis 1 (default)
039	2	Yes = Accel/decel is linear for plane 2 No = Accel/decel is sine type for axis 2 (default)
057	3	Yes = Accel/decel is linear for plane 3 No = Accel/decel is sine type for axis 3 (default)
075	4	Yes = Accel/decel is linear for plane 4 No = Accel/decel is sine type for axis 4 (default)



As a recommendation, the operator should set this parameter to "no" (sine ramping) for systems having high inertia and/or mass. Sinusoidal ramping will require approximately 60% more peak current than linear ramping.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

027
045
063
081

5.4.7. Clamp Feedrate (Program Steps/ms)

This parameter specifies the maximum feedrate allowed on the corresponding plane for all *contour type* motion (linear or circular) in that plane. The value specified in this parameter is given in program steps/ms and can range from 0.004 to 32,767 program steps/ms. The system default is 256.0 program steps/ms for each of the contour planes. A maximum feedrate must be specified for each of the active contour planes. Settings for this parameter are listed in Table 5-16.

Table 5-16. Settings for Parameters 027, 045, 063, and 081

Param #	Plane #	Range
027	1	0.0004 to 32,767 program steps/ms (default = 256.0)
045	2	0.0004 to 32,767 program steps/ms (default = 256.0)
063	3	0.0004 to 32,767 program steps/ms (default = 256.0)
081	4	0.0004 to 32,767 program steps/ms (default = 256.0)

If a contour feedrate (programmed or derived after MFO adjustment) is larger than the setting of this parameter, then the UNIDEX 500 will automatically clamp it to the appropriate *Clamp feedrate* value.



5.4.8. Corner Rounding Non-Ramp Time (1-32,000 ms)

Corner rounding can be used to remove sharp edges on corners from a profile. It does this by blending the deceleration of the current move with the acceleration of the next move.

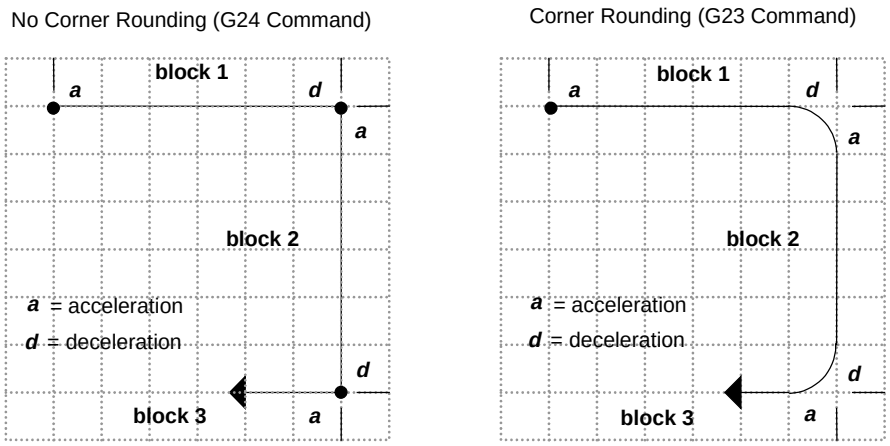


Figure 5-9. Sample Motion Path Shown with and without Corner Rounding

The non-ramp time specifies a time in milliseconds (from 1 to 32,000) during which the deceleration of one motion overlaps the acceleration of the next motion. This overlap causes one motion control block to begin its acceleration ramp before the preceding motion block finishes its deceleration. The result is a rounded corner, the size of which is determined by the acceleration/deceleration times and the setting of the corner rounding non-ramp time parameter. A sample non-ramp time overlap for corner rounding is illustrated in Figure 5-10.

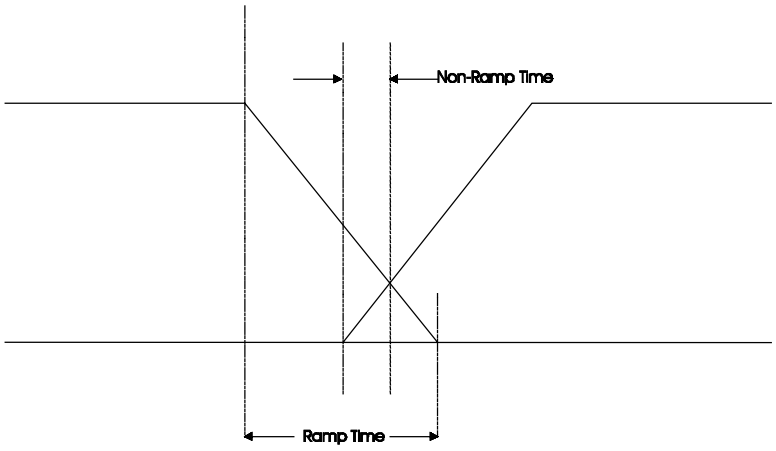


Figure 5-10. Sample of Non-Ramp Time Overlap for Corner Rounding

When corner rounding is not used (e.g., G24 or ROUNDING OFF programming command), each contour path decelerates to its target position before the next block of motion begins. Refer to Figure 5-11.

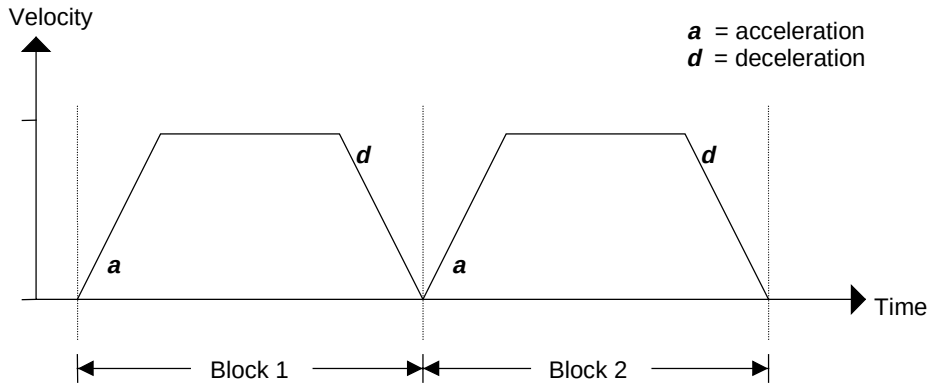


Figure 5-11. Velocity Diagram of Non-Corner Rounding (G24)

Non-ramp time parameter values can range from 1 to 32,000 ms and have default settings of 150 ms. Refer to Table 5-17.

Table 5-17. Settings for Parameters 028, 046, 064, and 082

Param #	Plane #	Range	Default Value
028	1	1-32000 ms	150 ms
046	2	1-32000 ms	150 ms
064	3	1-32000 ms	150 ms
082	4	1-32000 ms	150 ms

When performing a contour motion, this command effects the behavior of deceleration.



Programming a non-ramp time (using the ROUNDING *time* command) overrides (but does not change) the settings of parameters 028, 046, 064, or 082.



Make certain the non-ramp time is less than or equal to the ramp time.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

031

049

067

085

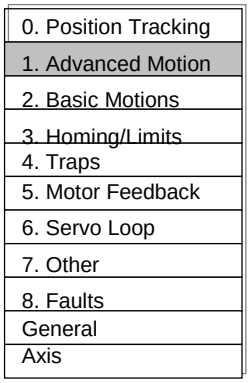
5.4.9. Contouring Mode

These parameters select the contouring mode. The default is 0, which uses the normal contouring mode, and 1 selects the alternate contour mode. This parameter is used to turn on enhanced G8 mode. Contour mode (0) blends moves (similar to corner rounding mode) together by combining deceleration of one move with the acceleration of the next move. The alternate mode does not. It requires that the last move be preceded by a G9 ("velocity profiling off") if in G8 mode. See the related parameter x83, *Filter time constant*, for more information. The settings for parameters 031, 049, 067, and 085 are given in Table 5-18.

Table 5-18. Settings for Parameters 31, 49, 67, and 85

Param #	Plane #	Range	Default Value
031	1	0 - 100	0
049	2	0 - 100	0
067	3	0 - 100	0
085	4	0 - 100	0

5.4.10. A/D Channel *n* Joystick Deadband

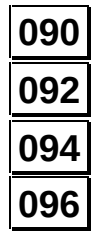


These parameters define the deadbands associated with the center position of the joystick. There is no resulting motion when the joystick is within this band. The parameter value is the number of A/D counts in the deadband. The default value is zero, which is internally interpreted as 16 A/D counts for backward compatibility.

The parameter definitions are listed in Table 5-19. Parameters 090 and 092 are currently not used because the joystick is connected to channel numbers 3 and 4. A/D channel no. 1 is normally the MFO input. A/D channel 2 is normally the user analog input.

Table 5-19. Joystick Deadband Parameters

Param #	A/D Channel	Default Value
090	A/D Channel No. 1	0
092	A/D Channel No. 2	0
094	A/D Channel No. 3 (joystick vertical axis)	0
096	A/D Channel No. 4 (joystick horizontal axis)	0



The A/D converter is 8 bits scaled so that 0 V gives an output of 0 and +5 V gives an output of 255. An A/D output of 128 corresponds to an input of +2.5 V.



5.4.11. A/D Channel *n* Joystick Center Position

091
093
095
097

These parameters specify the center position of the A/D inputs used for joystick mode. This allows the joystick pot to be digitally centered or calibrated. The parameter value is the number of A/D counts assigned to the center position.

The parameter definitions are listed in Table 5-20. Parameters 091 and 093 are currently not used because the joystick is connected to channel numbers 3 and 4. A/D channel number 1 is normally the MFO input. A/D channel number 2 is normally the user analog input.

Table 5-20. Joystick Center Position Parameters

Param #	A/D Channel	Default Value
091	A/D Channel No. 1	0
093	A/D Channel No. 2	0
095	A/D Channel No. 3 (joystick vertical axis)	0
097	A/D Channel No. 4 (joystick horizontal axis)	0



The A/D converter is 8 bits scaled so that 0 V gives an output of 0 and +5 V gives an output of 255. An A/D output of 128 corresponds to an input of +2.5 V.

5.4.12. Safe Zone Output Bit (0-8)

098

Parameter 098 specifies which UNIDEX 500 output to turn on (low) when all axes are in their specified safe zones. See parameters x75 and x76 under Homing and Limits for an explanation of safe zone.

This parameter can have a value of zero or 1 through 8. A parameter value of zero, which is the default value, defeats the safe zone function.

5.4.13. Option Board Setup Code

Parameter 099 indicates which option board is being used. The option board is determined by the status of the first two bits of parameter 099. Entering an appropriate decimal value for Parameter 099 can change the configuration. The settings for parameter 099 are given in Table 5-21.

bit #0: Setting bit #0 causes the U500 to enable access to the 4EN Option board. Inputs are read using the IN2 and IN3 commands. Outputs are written using the OU1, OU3, and OU4 commands.

bit #1 Setting bit #1 enables scanning of an optional iSBX encoder card. The encoder position can be read from the U500's memory at L:1BC3.

bit #2 If bit #2 is set, an optional 12-bit A/D will be scanned. This is only applicable for a special version of the U500 card.

bit #3 For U500 PCI only. If bit #3 is set to 1, hall effect signals are read through an OP500 cable connected to P1 of the U500 PCI card. If bit #3 is set to 0, hall signals are read through the standard P5 connector.

bits #2-#23 Reserved.

Table 5-21. Settings for Parameter 099

Param #	Range	Default Value
099	0 – 8,388,607	0

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

500

5.4.14. User Interrupt Setup Code

Parameter 500 sets the usage of the user interrupt input. The usage is determined by the status of the first two bits of parameter 500. Entering an appropriate decimal value for the parameter can change the configuration. This input is active low. Settings for parameter 500 are given in Table 5-22. Refer to file, 'enccapt.txt' located in the "\doc" installation subdirectory.

bit #0: Set to 1 to abort all axis motion on user interrupt.

bit #1 If set to 0, the UINT_N input will disable itself after the first occurrence. If this bit is set to 1, the UINT_N input will remain active.

bit #2 If bit #2 is set, UINT will generate a PC bus interrupt.

bit #3 For U500 PCI only. If bit #3 is set, U500 PCI will scan for a position capture at the servo loop update rate. See Appendix F, U500 Applications for more information.

bits #4-#23 Reserved.

Table 5-22. Settings for Parameter 500

Param #	Range	Default Value
500	0 – 8,388,607	0

501

5.4.15. Abort On Input High (0, 1-16)

Parameter 501 allows the user to define an input bit on the 16 IN/8 OUT I/O connector as a global abort input. All enabled axes will ramp to a stop and "Abort Active" will be displayed in the MDI window. This will occur when the input bit is in the logic "1" (+5 V or high impedance) state. A parameter setting of 0 (the default) defeats the Abort input. Parameter settings are given in Table 5-23.

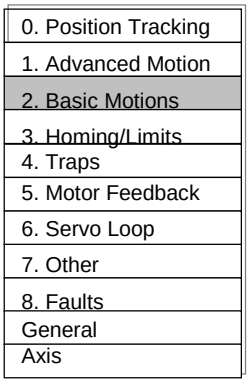
Table 5-23. Settings for Parameter 501

Parameter #	Range	Default Value
501	0, 1-16	0



A fault acknowledge must be issued to return to normal mode.

5.5. The Basic Motion Tab

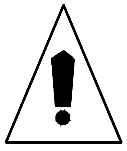


The basic motion parameters are used to configure the UNIDEX 500 for basic motion. Parameters on this tab define ramping times and feedrates for axes and contour planes. There are 25 parameters in the basic motion parameter group. These parameters are listed in Table 5-24 and explained in detail in this section.

Table 5-24. Basic Motion Parameters (Tab 2)

Param #	Description	Default Value
002	*MFO, 0 no MFO-pot, 1-255 pot offset*	0
019, 037, 055, 073	*Plane n Contour ramping time* (*n*=1, 2, 3, or 4)	150
022, 040, 058, 076	*Contour feedrate (program steps/ms)*	16.0
023, 041, 059, 077	*X point-to-point feedrate (program steps/ms)*	16.0
024, 042, 060, 078	*Y point-to-point feedrate (program steps/ms)*	16.0
025, 043, 061, 079	*Z point-to-point feedrate (program steps/ms)*	16.0
026, 044, 062, 080	*U point-to-point feedrate (program steps/ms)*	16.0

After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. This is done using the F9 RESET button located at the bottom of the main software window.



5.5.1. MFO, 0 No MFO-pot, 1-255 Pot Offset

This parameter is used to enable or disable an optional manual feed override (MFO) potentiometer (pot). The MFO pot, if used, is attached to the miscellaneous I/O connector of the DR500 chassis. If a BB500 or BB501 is being used, then the MFO pot is connected to the appropriate screw terminals.

The MFO pot offset parameter has a range from 0 to 255. A value of 0 should be used if the MFO option is not used. A 0 value can also be used to disable an existing MFO pot. If an MFO pot is enabled (that is, parameter 002>0), then the value set by this parameter represents an offset that becomes the 0% MFO position. Refer to Table 5-25.

Table 5-25. Settings for Parameter 002

Value	Description
0	Used when the MFO potentiometer option is not being used or to disable an existing MFO pot (default)
1-255	Specifies the MFO pot offset for 0% MFO

If the pot is enabled (i.e., parameter 002>0), the U500 reads the current pot position (as a value from 0 to 255 counts) through the A/D converter. The value of parameter 002 *shifts* the 0% MFO mark, thereby creating a user-definable low-end deadband over which the MFO is 0%. The new MFO percentage is then defined as a function of the MFO offset from parameter 002 (a value from 1-255) and the actual A/D converter value (0-255) read by the U500. Setting parameter 002>0, effectively creates a low-end pot deadband and automatically rescales the remainder of the pot range over the remaining number of converter counts. This is accomplished using the following equation.

MFO % = ((1 + Converter Value - Value of Parameter 002) / 255) * 100%

For example, setting parameter 002 = 55 will create a 55-count deadband at the low end of the pot range. The A/D converter data from 0 to 55 counts will be treated as a 0% MFO. Any data greater than 55 can be calculated by substituting the current A/D converter value (56-255) in the above equation and solving for MFO %. See Figure 5-12. The typical value for this parameter when connecting to a pot is 10.

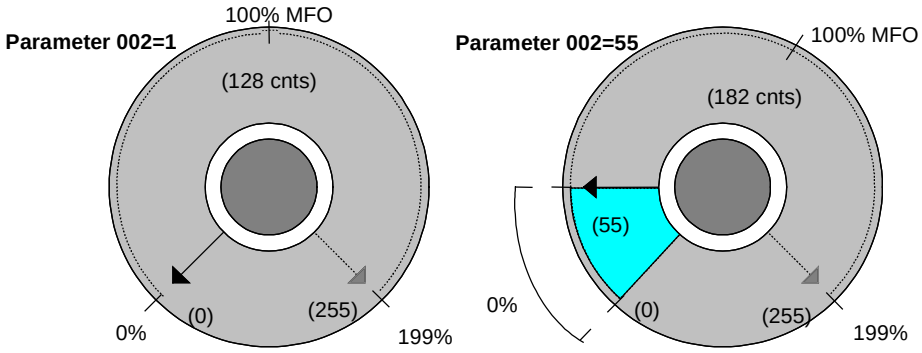


Figure 5-12. MFO Potentiometer with and without Offsets

5.5.2. Plane *n* Contour Ramping Time (ms)

The acceleration and deceleration time of linear and circular motion is set using parameters 019, 037, 055, and 073 (refer to Figure 5-13). These parameters also specify the time it takes to change velocities in velocity profiling mode. Ramp time applies to linear or sinusoidal acceleration profiles. A ramp time reflecting the system capabilities must be established for each of the active contour planes (parameter 019 for contour plane 1, parameter 037 for plane 2, etc.).

These parameters can range from 1 to 32,000 milliseconds. The system default for these parameters is 150 ms. Settings for these parameters are listed in Table 5-26.

Table 5-26. Settings for Parameters 019, 037, 055, and 073

Param #	Plane #	Range	Default
019	1	1-32,000 ms	150 ms
037	2	1-32,000 ms	150 ms
055	3	1-32,000 ms	150 ms
073	4	1-32,000 ms	150 ms

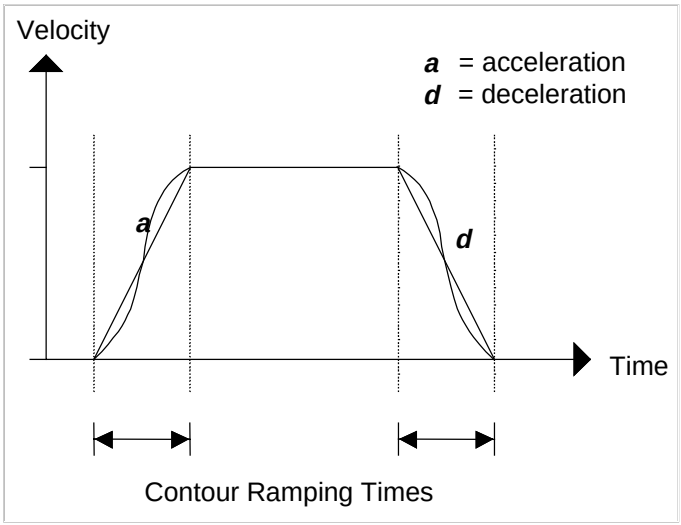


Figure 5-13. Contour Ramping (Acceleration/Deceleration) Time

Systems with high mass or inertia will require longer ramping times.

Plane n contour ramping time does not apply to G0/INdex, HOme, and FReerun moves.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

022
040
058
076

5.5.3. Contour Feedrate (Program Steps/ms) for Plane *n*

These parameters specify a default feedrate (in program steps/ms) to be used by axes in each contour plane if a feedrate is not explicitly stated (in the program) for that plane. Typically, however, most programs that request contour type motion specify a feedrate. A feedrate that is explicitly stated in a program (for a particular plane) will override the value of parameter 022, 040, 058, or 076, as appropriate. A default feedrate must be set for each active contour plane.

These parameters can have values from 1 to 32,767 program steps/ms. The system default is 16 program steps/ms. Refer to Table 5-27.

Table 5-27. Settings for Parameters 022, 040, 058, and 076

Param #	Plane #	Range	Default
022	1	1-32,767 program steps/ms	16 program steps/ms
040	2	1-32,767 program steps/ms	16 program steps/ms
058	3	1-32,767 program steps/ms	16 program steps/ms
076	4	1-32,767 program steps/ms	16 program steps/ms

023 - 026
041 - 044
059 - 062
077 - 080

5.5.4. X, Y, Z, and U Axes' Point-to-point Feedrates (Program Steps/ms) for Plane *n*

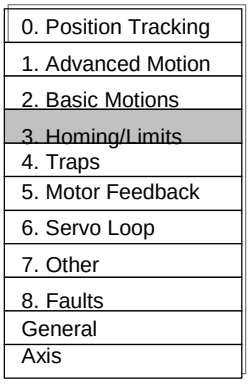
These parameters set the default axis feedrates (in program steps/ms) of axes 1, 2, 3, and 4 for each active contour plane (1-4) when performing point-to-point (indexed) moves. A command line feed rate (if specified) will override the settings of these parameters. See the INDEX command for more details.

These parameters can have values from 0.004 to 32,767 program steps/ms. The system default is 16.0 program steps/ms. Refer to Table 5-28 for plane assignments and settings.

Table 5-28. Point-to-point Feedrate Parameter Assignments and Settings

Plane	Axes				Ranges in	Defaults in
#	1	2	3	4	prog steps/ms	prog steps/ms
1	023	024	025	026	0.004 to 32,767	16.0
2	041	042	043	044	0.004 to 32,767	16.0
3	059	060	061	062	0.004 to 32,767	16.0
4	077	078	079	080	0.004 to 32,767	16.0

5.6. The Homing and Limits Tab



The homing and limits parameters are used to configure the UNIDEX 500's *home cycle* and the accompanying limit switches. The *home cycle* is the process in which an axis is commanded to a known reference position (e.g., a zero position).

There are 17 parameters in the homing and limits parameter group. Each of these parameters has a unique setting for each of the four axes (i.e., parameter x02 is actually four parameters [102, 202, 302, and 402], each of which corresponds to axes 1-4, respectively). The homing and limits parameters are listed in Table 5-29 and are explained in detail in the following sections. A brief description of the home cycle precedes the parameter descriptions.

Table 5-29. Homing and Limits Parameters (Tab 3)

Param #	Description	Default Values (for each axis)
x02	*Home direction is CCW (y/n)*	Yes
x03	*Home switch normal open (y/n)*	Yes
x04	*Power-on home feedrate (machine steps/ms)*	25
x05	*Normal home feed rate (machine steps/ms)*	25
x06	*Home offset (machine steps)*	0
x07	*Home switch to marker (machine steps)*	0
x08	*Home/limit switch debounce (ms)*	100
x09	*Limit switch normal open (y/n)*	Yes
x10	*Switch to mechanical stop (machine steps)*	2000
x22	*CCW software limit (machine steps)*	140737488355328
x23	*CW software limit (machine steps)*	140737488355328
x73	*Enable vel/accel feedforward during home*	Yes
x74	*Use home limit during home cycle*	Yes
x75	*Safe zone -limit (machine steps)*	0
x76	*Safe zone + limit (machine steps)*	0
x77	*Home/limit switch debounce (machine steps)*	0
x81	*Home marker search speed (machine steps/ms)*	0

After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. This is done using the F9 RESET button located at the bottom of the main software window.



5.6.1. The Home Cycle

The home cycle is used to move specified axes to a hardware referenced position. Two distinct types of home cycles are used by the UNIDEX 500: the power-on home cycle and the runtime home cycle. There are individual axis feedrates associated with each of these cycles. The *Power-on home feedrate (machine steps/ms)* parameter setting (parameter x04) is used the first time the axis is sent home following power-up. The *Normal home feedrate (machine steps/ms)* parameter setting (parameter x05) is used for subsequent home cycles. Both home cycles use the *Max ac/de (machine steps/ms/ms)* parameter (x16) for the acceleration/deceleration rates.



Power-up Home Cycle

The *Power-on home feedrate (machine steps/ms)* is usually set slower than the *Normal home feedrate (machine steps/ms).*

The home cycle is illustrated in Figure 5-14 and is comprised of the following moves.

1. The axis will move from its current position (❶) at the rate set by the *Power-on home feedrate (machine steps/ms)* parameter (x04) in the direction set by the *Home direction is CCW (y/n)* parameter (x02) until the home limit input is activated (❷). The polarity of the home limit switch is set by the *Home switch normal open (y/n)* parameter (x03). During a home cycle, the end of travel limit in the home direction will be ignored while the home limit input is active.
2. The axis reverses direction and moves out of the limit (❸). The servo begins searching for a marker pulse after moving the number of machine steps specified in the *Home switch to marker (machine steps)* parameter (x07). For the U500 ISA, the axis will move at 2,000 machine steps per second while searching for the marker. The axis will stop on the marker (❹) then move the distance specified by the *Home offset (machine steps)* parameter (x06). When the move is complete, the UNIDEX 500 will reset its position counters to zero and use this position as a reference for future home cycles (❺). The U500 PCI latches the encoder position of the marker while moving at the home feedrate.

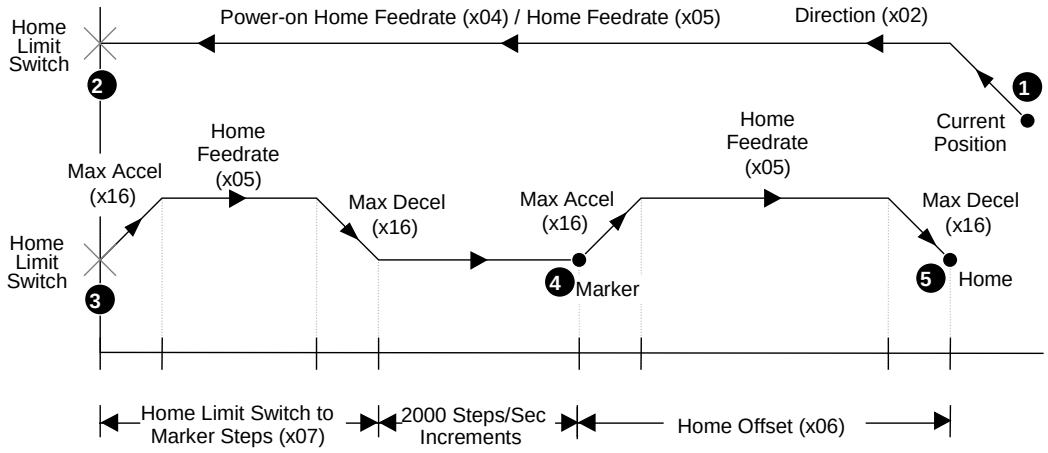


Figure 5-14. Home Cycle

The marker search process may take considerable time with high resolution encoder feedback systems. The *Home switch to marker (machine steps)* (x07) should be set to minimize this time. (**U500 ISA only**)

If the feedback device type is a resolver, the axis will move to a null instead of a marker.

Runtime Home Cycle

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x02

5.6.2. Home Direction is CCW (Y/N)

Each axis of the UNIDEX 500 needs to be configured for the direction that the axis motor will turn when going to the home position (refer to Figure 5-14 on page 5-39). This parameter must be configured to reflect the motor direction that causes the axis to move toward the home limit switch. Motor direction (clockwise [CW] or counterclockwise [CCW]) is specified "looking into" the shaft end of the motor. Refer to Figure 5-15.

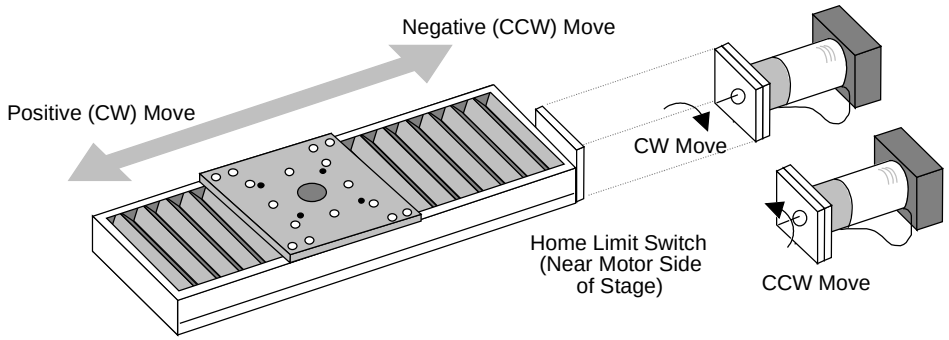


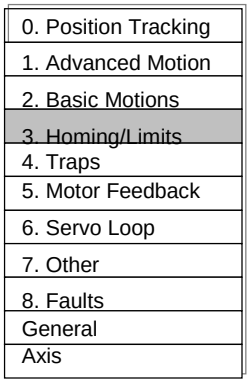
Figure 5-15. Typical Stage Showing CW and CCW Motor Rotation

This parameter can be set to one of two possible values, which are explained in Table 5-30.

Table 5-30. Settings for Parameter x02

Param #	Axis #	Values	Description (* indicates default setting)
102	1	Yes (**Y**)	CCW motor rotation moves axis to home position *
		No (N)	CW motor rotation moves axis to home position
202	2	Yes (**Y**)	CCW motor rotation moves axis to home position *
		No (N)	CW motor rotation moves axis to home position
302	3	Yes (**Y**)	CCW motor rotation moves axis to home position *
		No (N)	CW motor rotation moves axis to home position
402	4	Yes (**Y**)	CCW motor rotation moves axis to home position *
		No (N)	CW motor rotation moves axis to home position

Although most stages are typically configured as shown in Figure 5-15 (i.e., CW motor rotation is a positive move), stages using fold-back motors and/or gearheads may have positive and negative moves reversed.



5.6.3. Home Switch Normally Open (Y/N)

This parameter must be configured to correspond to the polarity of the home limit switch in its inactive state. Typically, Aerotech stages are configured with normally open home limit switches.

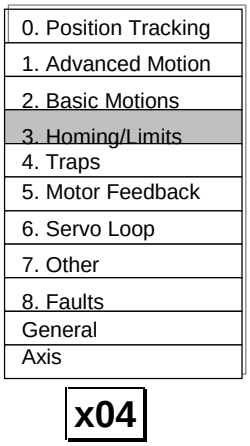
As the stage moves toward the home limit, a projection on the underside of the stage table comes in contact with the home limit switch assembly. In a normally open configuration (x03 = yes), the stage table projection moves into the limit switch thereby closing the contacts and causing a home limit fault. Conversely, in a normally closed configuration (x03 = no), the stage table projection moves into the limit switch thereby opening the contacts and causing a home limit fault.

The settings for parameters 103, 203, 303, and 403 are summarized in Table 5-31.

Table 5-31. Settings for Parameter x03

Param #	Axis #	Values	Description
103	1	Yes (**Y**)	Home limit switch is normally open (default)
		No (N)	Home limit switch is normally closed
203	2	Yes (**Y**)	Home limit switch is normally open (default)
		No (N)	Home limit switch is normally closed
303	3	Yes (**Y**)	Home limit switch is normally open (default)
		No (N)	Home limit switch is normally closed
403	4	Yes (**Y**)	Home limit switch is normally open (default)
		No (N)	Home limit switch is normally closed

The limit inputs are pulled to logic high on the UNIDEX 500 control board.



5.6.4. Power-on Home Feedrate (Machine Steps/ms)

Following power-up, the first commanded home cycle will be a power-on home cycle. The axis will move at the feedrate set by parameter x04 (the *Power-on home feedrate* parameter given in machine steps per ms) until the home limit input becomes active. Refer to the home cycle figures at the beginning of this section.

This parameter can range in value from 0.004 to 32,767 machine steps per ms. The default value for this parameter is 25 machine steps per ms. Refer to Table 5-32.

Table 5-32. Settings for Parameter x04

Param #	Axis #	Range	Default
104	1	0.004 to 32,767 machine steps/ms	25 machine steps/ms
204	2	0.004 to 32,767 machine steps/ms	25 machine steps/ms
304	3	0.004 to 32,767 machine steps/ms	25 machine steps/ms
404	4	0.004 to 32,767 machine steps/ms	25 machine steps/ms



During a home cycle, the end of travel limit in the home direction will be ignored while the home limit input is active.



To avoid possible equipment damage, this feedrate should be set to a value considerably slower than the *Normal home feedrate*, since axis disorientation is likely at power up.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x05

5.6.5. Normal Home Feedrate (Machine Steps/ms)

Following power-up, the first commanded home cycle will be a power-on home cycle. Subsequent home cycles are runtime home cycles. Parameter x05 specifies the normal home feedrate for each axis during runtime home cycles.

This parameter can range in value from 0.004 to 32,767 machine steps per ms. The default value for this parameter is 25 machine steps per ms. Refer to Table 5-33.

Table 5-33. Settings for Parameter x05

Param #	Axis #	Range	Default
105	1	0.004 to 32,767 machine steps/ms	25 machine steps/ms
205	2	0.004 to 32,767 machine steps/ms	25 machine steps/ms
305	3	0.004 to 32,767 machine steps/ms	25 machine steps/ms
405	4	0.004 to 32,767 machine steps/ms	25 machine steps/ms

x06

5.6.6. Home Offset (Machine Steps)

If the desired home position is not at the marker location, parameter x06 may be used to set the offset value. The distance from the encoder marker (or resolver null) to the home position must be measured, converted into machine steps, and entered into parameter x06. After the axis has moved the offset distance, the position counters will reset to zero.

Parameter x06 has values that range from -7×10^{13} to 7×10^{13}. A value of 0 (the default) indicates that the encoder marker (or resolver null) is located at the home position. Refer to Table 5-34.

Table 5-34. Settings for Parameter x06

Param #	Axis #	Range (in machine steps)	Default
106	1	-7×10^{13} to 7×10^{13}	0 (no home offset)
206	2	-7×10^{13} to 7×10^{13}	0 (no home offset)
306	3	-7×10^{13} to 7×10^{13}	0 (no home offset)
406	4	-7×10^{13} to 7×10^{13}	0 (no home offset)

The "Axis Positions" field in the Diagnostics window, can be helpful in setting this parameter.



The marker search process may take a considerable amount of time when using a high resolution encoder for feedback. Parameter x07 should be set to minimize this time.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x07

5.6.7. Home Switch to Marker (Machine Steps) – U500 ISA Only

Parameter x07 decreases the time involved in a homing cycle. This is done by specifying the distance (in machine steps) between the home switch and the encoder marker. After the home limit switch is detected during a home cycle, the axis moves at a rate given by x05 (with appropriate accel/decel times) for a distance given by parameter x07. The axis then slows to 2000 machine steps/sec until the encoder marker is located.

Parameter x07 can have values that range from $-(2)^{47}$ to $+(2)^{47}$. The value of this parameter specifies the distance (in machine steps) that the axis travels (at a speed specified by parameter x05) before slowing to search for the encoder marker. The sign of parameter x07 should reflect the orientation of the limit switch. When the encoder and programming directions are the same, the sign should be positive. If these directions are switched (i.e., the limit switch is at the opposite side of the stage), then the sign should be negative. A value of 0 (the default) forces the UNIDEX 500 to immediately begin searching for the encoder marker (at 2,000 machine steps/sec increments) after the limit switch is located. Refer to Table 5-35.

Table 5-35. Settings for Parameter x07

Param #	Axis #	Range (in machine steps)	Default "Switch to Marker" Distance
107	1	$-(2)^{47}$ to $+(2)^{47}$	0 machine steps
207	2	$-(2)^{47}$ to $+(2)^{47}$	0 machine steps
307	3	$-(2)^{47}$ to $+(2)^{47}$	0 machine steps
407	4	$-(2)^{47}$ to $+(2)^{47}$	0 machine steps



If this parameter is set too large, the UNIDEX 500 may overshoot the marker. (Refer to the home cycle figures at the beginning of this section.)



If the feedback device is a resolver, the axis will move to the absolute zero position of the resolver instead of the marker.



The value selected for this parameter is dependent upon the system configuration and hardware variances.



The value of the *Power-on home feedrate* (x04), *Normal home feedrate* (x05) or *Max ac/de* (x16) parameter determines the optimum effect of this parameter.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x08

5.6.8. Home/Limit Switch Debounce (ms)

Depending on the type of home limit switch being used, a certain amount of time is required for the switch to remain in the "off" position before the UNIDEX 500 considers it to be inactive. This parameter specifies a time (in milliseconds) that the home limit switch must remain off before the U500 considers it to be inactive. This parameter generates a hysteresis, which is useful when interfacing to electrically "noisy" switches.

This parameter value can range from 0 to 8,388,607 ms. The default value for this parameter is 100 ms. Refer to Table 5-36.

Table 5-36. Settings for Parameter x08

Param #	Axis #	Range (in milliseconds)	Default
108	1	0 to 8,388,607 ms	100 ms
208	2	0 to 8,388,607 ms	100 ms
308	3	0 to 8,388,607 ms	100 ms
408	4	0 to 8,388,607 ms	100 ms

It is recommended that parameter x77 be used instead. This will make the home cycle less vulnerable to other parameter changes.

x09

5.6.9. Limit Switch Normally Open (Y/N)

This parameter must be configured to correspond to the polarity of the axis limit switch in its inactive state. The parameter can have one of two possible values: "yes" for a normally open limit switch (default) and "no" for a normally closed limit switch. Refer to Table 5-37.

Table 5-37. Settings for Parameter x09

Param #	Axis #	Values
109	1	Yes - normally open limit switch (default)
		No - normally closed limit switch
209	2	Yes - normally open limit switch (default)
		No - normally closed limit switch
309	3	Yes - normally open limit switch (default)
		No - normally closed limit switch
409	4	Yes - normally open limit switch (default)
		No - normally closed limit switch

It is recommended that the limit switch be manually checked before connecting the motor to the system.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x10
x22

5.6.10. Switch to Mechanical Stop (Machine Steps)

This parameter specifies the stopping distance, in machine steps, when an axis hits a limit switch. When this value is specified, a deceleration rate is calculated so that deceleration occurs within the specified distance regardless of the current speed. This prevents the stage from hitting a mechanical stop. If axis parameter x16 (*Max ac/de*) specifies a greater deceleration rate, then the value of parameter x16 rather than the calculated rate is used.

The default value for parameter x10 is 2000 machine steps. Setting this parameter to zero forces the deceleration rate to take on the value specified in axis parameter x16.

5.6.11. CCW Software Limit (Machine Steps)

Parameter x22 is used to establish a software limit (in machine steps) when the motor is rotating in the CCW direction. This limit is referenced from the hardware home position and is not active until the axis is sent home.

This parameter value can range from $(-2)^{47}$ to $(+2)^{47}$ machine steps. The default and minimum value is $(-2)^{47}$ (which equals -140,737,488,355,328 steps). See Table 5-38.

Table 5-38. Settings for Parameter x22

Param #	Axis #	Range (in machine steps)	Default
122	1	$(-2)^{47}$ to $(+2)^{47}$	$(-2)^{47}$ machine steps
222	2	$(-2)^{47}$ to $(+2)^{47}$	$(-2)^{47}$ machine steps
322	3	$(-2)^{47}$ to $(+2)^{47}$	$(-2)^{47}$ machine steps
422	4	$(-2)^{47}$ to $(+2)^{47}$	$(-2)^{47}$ machine steps



The "Fault Mask" default setting enables the software limits. Refer to the next section (Traps) for more information.

x23

5.6.12. CW Software Limit (Machine Steps)

This parameter is used to establish a software limit (in machine steps) when the motor is rotating in the CW direction. This limit is referenced from the hardware home position and is not active until the axis has completed a home cycle.

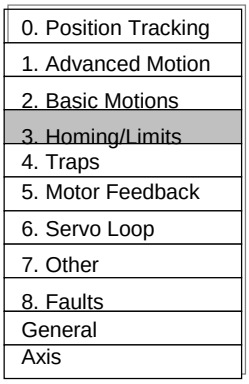
This parameter value ranges form $(-2)^{47}$ to $(+2)^{47}$ machine steps. The default and maximum value is $(+2)^{47}$ (which equals 140,737,488,355,328 steps). See Table 5-39.

Table 5-39. Settings for Parameter x23

Param #	Axis #	Range (in machine steps)	Default
123	1	$(-2)^{47}$ to $(+2)^{47}$	$(+2)^{47}$ machine steps
223	2	$(-2)^{47}$ to $(+2)^{47}$	$(+2)^{47}$ machine steps
323	3	$(-2)^{47}$ to $(+2)^{47}$	$(+2)^{47}$ machine steps
423	4	$(-2)^{47}$ to $(+2)^{47}$	$(+2)^{47}$ machine steps

The "Fault Mask" default setting enables the software limits. Refer to the next section (Traps) for more information.





5.6.13. Enable Vel/Accel Feedforward During Home (Y/N)

Velocity feedforward sends velocity commands directly to the velocity loop portion of the servo loop. Acceleration feedforward (Aff) sends acceleration commands directly to the amplifier. During a home cycle marker search, the commanded velocity is half of the servo sample rate (4 Khz/2 = 2 kHz) or 1 step every other servo cycle. The resulting velocity command is a sequence like this: (0,1,0,1,0,1,.....). This can result in a high pitched squealing sound. Acceleration feedforward usually produces increased squealing especially if Aff is a large value.

When parameter x73 is set to "no," it automatically disables both feedforward terms during the entire home cycle. When the home cycle is complete, the feedforward terms are re-enabled.

The default for this parameter is "yes." Refer to Table 5-40.

Table 5-40. Settings for Parameter x73

Param #	Axis #	Values
173	1	**Yes** - Acceleration and Velocity feedforward are enabled (default)
		No - Acceleration and Velocity feedforward are disabled
273	2	**Yes** - Acceleration and Velocity feedforward are enabled (default)
		No - Acceleration and Velocity feedforward are disabled
373	3	**Yes** - Acceleration and Velocity feedforward are enabled (default)
		No - Acceleration and Velocity feedforward are disabled
473	4	**Yes** - Acceleration and Velocity feedforward are enabled (default)
		No - Acceleration and Velocity feedforward are disabled

- 0. Position Tracking
- 1. Advanced Motion
- 2. Basic Motions
- 3. Homing/Limits
- 4. Traps
- 5. Motor Feedback
- 6. Servo Loop
- 7. Other
- 8. Faults
- General
- Axis

x74

5.6.14. Use Home Limit During Home Cycle (Y/N)

The UNIDEX 500 has three limit inputs, "CW," "CCW," and "HOME." If this parameter is set to "yes," the home cycle will move in the specified direction until the home limit is activated. If set to "no," the home cycle will ignore the home limit input. Instead, it will reference the limit switch that is in the home direction. In this case, the limit switch will not generate a fault condition.

The default for this parameter is "yes." Refer to Table 5-41.

Table 5-41. Settings for Parameter x74

Param #	Axis #	Values
174	1	**Yes** - Uses home limit input (default)
		No - Ignores home limit input
274	2	**Yes** - Uses home limit input (default)
		No - Ignores home limit input
374	3	**Yes** - Uses home limit input (default)
		No - Ignores home limit input
474	4	**Yes** - Uses home limit input (default)
		No - Ignores home limit input

x75

x76

5.6.15. Safe Zone Limits (Machine Steps)

A safe zone is a region defined by a range of specified positions on 1, 2, 3, or 4 axes. For each axis, one range within its travel may be defined for this function with the *Safe zone +limit* and *-limit* parameters. This range is in machine steps and is referenced to the hardware home position. Refer to Table 5-42.

Each axis with a specified range must be enabled, homed, and within the specified limits to be considered within the safe zone. The output bit specified by general parameter number 098 will be driven to its active state (low impedance to ground) when all axes are in their safe zone ranges. Setting parameters x75 and x76 to zero (the default value) defeats the safe zone function for that axis. Also see general parameter number 098 for more information.

Table 5-42. Safe Zone Limit Parameters

Param #	Definition	Default Value
x75	Safe zone -limit (machine steps)	0
x76	Safe zone +limit (machine steps)	0

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x77

5.6.16. Home/Limit Switch Debounce (Steps)

This parameter specifies the distance that the axis takes to decelerate when moving out of the home or limit switch during a home cycle. It overrides axis parameter number x08, *Home/limit switch debounce (ms)*. The value is a distance in machine steps. A zero value defeats the function and reverts to axis parameter x08 for the debounce time. The range and default values for this parameter are shown in Table 5-43.

Table 5-43. Settings for Parameter x77

Param #	Axis #	Range	Default Value
177	1	0 - 8388607 mach. steps	750 mach. steps
277	2	0 - 8388607 mach. steps	750 mach. steps
377	3	0 - 8388607 mach. steps	750 mach. steps
477	4	0 - 8388607 mach. steps	750 mach. steps

The use of a debounce ramp distance during the home cycle is recommended. It allows the user to change the home speed and/or acceleration/deceleration (ac/de) rate without having to readjust all home search distances.

The ramp distance can be calculated from:

$$x = {}^1/_2 vt$$

or,

$$x = {}^1/_2 at^2$$

where: x = ramp distance (steps)

v = home velocity (steps/ms)

a = acceleration (steps/ms/ms)

t = ramp time (ms)

x81

5.6.17. Home Marker Search Speed (Machine Steps/ms) – U500 ISA Only

This parameter allows the user to slow down the speed at which an axis searches for the marker. The units are in machine steps/msec. The parameter may have a value ranging from 0 to 2 machine steps/msec. The default value of 0 will cause the U500 to search for the home limit at the preset speed of 2 machine steps per millisecond. Refer to Table 5-44.

Table 5-44. Settings for Parameter x81

Param #	Axis #	Range (machine steps/ms)	Default Value (machine steps/ms)
181	1	0 to 2	0
281	2	0 to 2	0
381	3	0 to 2	0
481	4	0 to 2	0

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

5.7. The Traps Tab

Trap parameters are a part of the UNIDEX 500's error checking and safety system. They are used to define the limits for fault conditions. The trap parameters for the U500 are listed in Table 5-45. These parameters are explained in the sections that follow. For additional information, refer to The Faults Tab section.

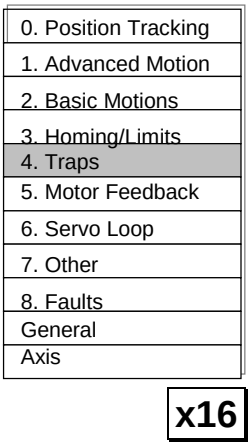
Table 5-45. Trap Parameters

Param #	Description	Default Value(s)
x16	*Max ac/de (machine steps/ms/ms)*	1.0
x17	*Top feedrate (machine steps/ms)*	440
x18	*Max velocity error (0-8,388,607)*	1000
x19	*Max position error (0-8,388,607)*	4000
x20	*Max integral error (0-8,388,607)*	655,360
x48	*RMS current trap (0-100%)*	29.6875
x49	*RMS current sample time (1-16,383 ms)*	10,000
x53	*Clamp current output (0-100%)*	99.2188
x54	*Which output bit for AUX OUTPUT*	1,2,3,4
x70	*Drive fault signal active low*	Yes
x84	*AUX OUTPUT active high*	Yes



IMPORTANT

After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. This is done using the F9 RESET button located at the bottom of the main software window.



5.7.1. Max Ac/De (Machine Steps/ms/ms)

Parameter x16 is the *Maximum ac/de* parameter. The value of this parameter specifies the acceleration/deceleration (ac/de) rate of axis motion for all freerun (FReerun), home (HOme), and point-to-point (G0/INdex) acceleration profiles. The same value is used for both acceleration and deceleration.

This parameter value can range from 0.004 to 255 machine steps/ms^2. The default value is 1.0 machine steps/ms^2. Refer to Table 5-46.

Table 5-46. Settings for Parameter x16

Param #	Axis #	Range (in machine steps/ms^2)	Default (in machine steps/ms^2)
116	1	0.004 to 255	1.0
216	2	0.004 to 255	1.0
316	3	0.004 to 255	1.0
416	4	0.004 to 255	1.0

The value of parameter x16 should be reduced if abrupt motion or servo traps occur during acceleration or deceleration during the home cycle (HOme), freerun (FReerun), or G0(INdex) programming commands.



This parameter does not apply to linear and circular contour motion. Contour motion uses the "*Plane n contour ramping time*"(019, 037, 055, 073) parameter.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x17

5.7.2. Top Feedrate (Machine Steps/ms)

This parameter sets the highest speed (in machine steps/ms) for which the axis is mechanically configured. If a feedrate is requested that is higher than the value specified in x17, then the message "Feedrate Error" is displayed in the status window of the software. In addition, the feedrate error status can be viewed from the Diagnostics screen in the software. See Chapter 4: The Software Interface for more information.

This parameter value can range from 0.004 to 131,071 machine steps/ms. The default value is 440 machine steps/ms. Refer to Table 5-47.

Table 5-47. Settings for Parameter x17

Param #	Axis #	Range (in machine steps/ms)	Default (in machine steps/ms)
117	1	0.004 to 131,071	440
217	2	0.004 to 131,071	440
317	3	0.004 to 131,071	440
417	4	0.004 to 131,071	440



This parameter is provided as a system safety feature. The system will "trap" if a feedrate that is higher than this setting is inadvertently requested.



A custom interface program using the *Aer_read_status* C library function can read the axis status. Refer to Chapter 10: Programming Tools for more information about C library functions. Refer to the Faults Tab section for more information about the fault mask.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x18

5.7.3. Max Velocity Error (0-8,388,607)

This parameter sets the maximum amount of velocity error (the difference between the actual velocity and the programmed velocity) that is acceptable in the application. For most applications it is advisable to set the maximum velocity error to the same value as the commanded velocity. The units of this parameter are machine steps per servo cycle (the duration of one servo cycle is 0.25 ms or 1 qms [quarter millisecond]).

If the velocity of an axis exceeds the value set in x18, then the message "Velocity Error" is displayed in the status window of the software. In addition, the velocity error state can be viewed from the Diagnostics screen in the software. See Chapter 4: The Software Interface for more information.

This parameter value can range from 0 to 8,388,607 machine steps/qms. The default value is 1,000 machine steps/qms. Refer to Table 5-48.

Table 5-48. Settings for Parameter x18

Param #	Axis #	Range (in machine steps/qms)	Default (in machine steps/qms)
118	1	0 to 8,388,607	1000
218	2	0 to 8,388,607	1000
318	3	0 to 8,388,607	1000
418	4	0 to 8,388,607	1000

A custom interface program using the *Aer_read_status* C library function can read the axis status. Refer to Chapter 10: Programming Tools for more information about C library functions. Refer to the Faults Tab section for more information about the fault mask.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x19

5.7.4. Max Position Error (0-8,388,607)

This parameter sets the maximum amount of position error (the difference between the actual position and the programmed position) that is allowed before a fault is generated.

If the position error of an axis exceeds the value set in x19, then the message "Position Error" is displayed in the status window of the software. In addition, the position error state can be viewed from the Diagnostics screen in the software. See Chapter 4: The Software Interface for more information.

This feature can be used to detect abnormal runtime conditions such as mechanical degradation of the system, motor failure, amplifier failure, etc. The U500 Axis Scope window should be used to evaluate the position error dynamically under typical operation.

This parameter value can range from 0 to 8,388,607 machine steps. The default value is 4000 machine steps. Refer to Table 5-49.

Table 5-49. Settings for Parameter x19

Param #	Axis #	Range (in machine steps)	Default (in machine steps)
119	1	0 to 8,388,607	4000
219	2	0 to 8,388,607	4000
319	3	0 to 8,388,607	4000
419	4	0 to 8,388,607	4000



This value may need to be significantly higher when tuning the servo loop for the first time.



A custom interface program using the *Aer_read_status* C library function can read the axis status. Refer to Chapter 10: Programming Tools for more information about C library functions. Refer to the Faults Tab section for more information about the fault mask.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x20

5.7.5. Max Integral Error (0-8,833,607)

This parameter sets the maximum amount of acceptable integral error before an error condition is generated. If this setting is exceeded, an "Integral Error" message is displayed. This type of error indicates an amplifier or motion failure.

If the integral error exceeds the value set in x20, then the message "Integral Error" is displayed in the status window of the software. In addition, the integral error state can be viewed from the Diagnostics screen in the software. See Chapter 4: The Software Interface for more information.

This parameter value can range from 0 to 8,388,607. The default value is 655,360. Refer to Table 5-50.

Table 5-50. Settings for Parameter x20

Param #	Axis #	Range	Default
120	1	0 to 8,388,607	655,360
220	2	0 to 8,388,607	655,360
320	3	0 to 8,388,607	655,360
420	4	0 to 8,388,607	655,360

A custom interface program using the *Aer_read_status* C library function can read the axis status. Refer to Chapter 10: Programming Tools for more information about C library functions. Refer to the Faults Tab section for more information about the fault mask.



5.7.6. RMS Current Trap (0-100%)

The RMS current trap parameter specifies a percentage (0% to 100%) of the maximum output voltage (± 10 volts nominal) commanded by the UNIDEX 500 that corresponds to the desired RMS current limit. An "RMS Current Level Exceeded" fault condition occurs if the RMS current exceeds the RMS limit for the specified *RMS current sample time* (x49). Likewise, a fault occurs if twice the RMS current limit is exceeded for half of the *RMS current sample time* (x49) (and so on, for any fractional portion of the RMS current sample time). An RMS fault may occur before the RMS current sample time expires if the accumulated RMS current level for the present sample period exceeds the product of the RMS level and the sample time. Refer to Figure 5-16.

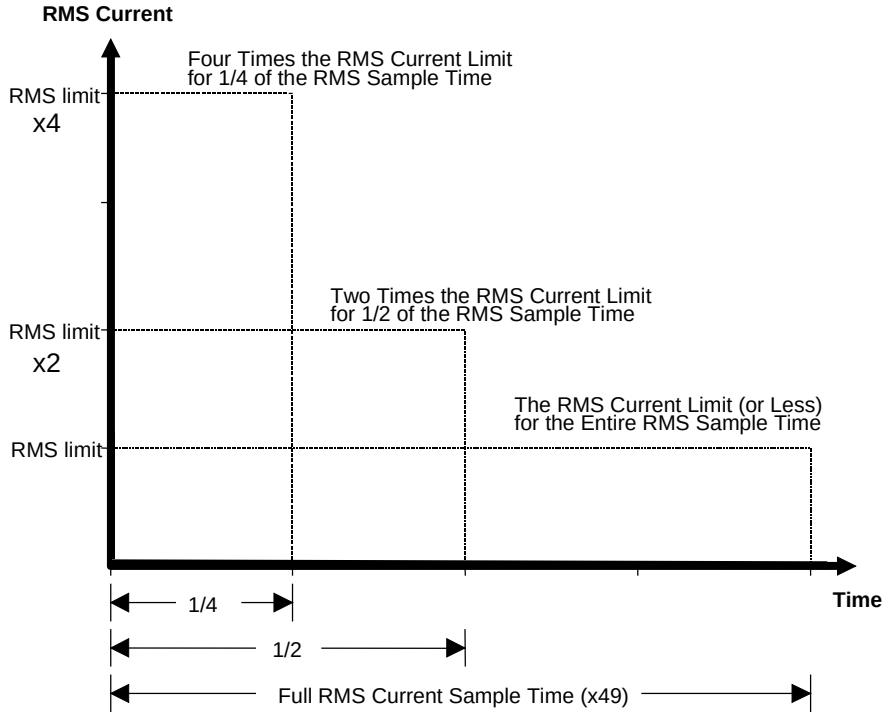


Figure 5-16. Sample RMS Current Maximums

When an RMS current trap occurs, the message "RMS Current Level Exceeded" is displayed in the status window of the software. In addition, the RMS current limit error can be viewed from the Diagnostics screen in the software. See Chapter 4: The Software Interface, for more information.

This parameter value can range from 0% to 100%. The default value is 30%. Refer to Table 5-51.

Table 5-51. Settings for Parameter x48

Param #	Axis #	Range	Default
148	1	0% to 100%	29.6875 %
248	2	0% to 100%	29.6875 %
348	3	0% to 100%	29.6875 %
448	4	0% to 100%	29.6875 %

If the value of parameter x48 is set to 100%, an RMS current trap will never be generated for the associated axis.



This parameter is used in conjunction with parameter x49 to determine RMS current level faults and can be thought of as an electronic fuse.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x49

5.7.7. RMS Current Sample Time (1-16,383 ms)

Parameter x49 sets the RMS current sample time. This time represents the span over which the RMS current must remain below the RMS current limit (calculated using parameter x48), otherwise an "RMS Current Level Exceeded" fault will occur. A fault may occur before the sample time expires if the accumulated RMS current level for the present sample period exceeds the corresponding RMS level for the fractional portion of the sample time (e.g., twice the level for half the time, three time the level for one third the time, etc.). Refer to Figure 5-16.

The RMS current sample time parameter may be referred to as the *thermal time constant*. This name reflects the function of the parameter, i.e., the RMS current drawn by a motor over a period of time will tend to heat up the motor. Therefore, the operator should choose a parameter value that will cause a fault *before* the motor overheats and fails.

This parameter value can range from 1 to 16,383 ms. The default value is 10,000 ms (or 10 seconds). Refer to Table 5-52.

Table 5-52. Settings for Parameter x49

Param #	Axis #	Range	Default
149	1	1 to 16,383 ms	10,000 ms
249	2	1 to 16,383 ms	10,000 ms
349	3	1 to 16,383 ms	10,000 ms
449	4	1 to 16,383 ms	10,000 ms



This parameter is used in conjunction with parameter x48 to determine RMS current level faults.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x53

5.7.8. Clamp Current Output (0-100%)

Parameter x53 is the *Clamp current output* parameter. The maximum output voltage of the control loop may be clamped in order to limit the amplifier current and motor torque. This parameter is expressed as a percentage of the maximum output voltage.

The actual motor current depends on amplifier scaling. This parameter should be set such that the maximum peak current of the motor is not exceeded. A fault condition is not generated if the UNIDEX 500 tries to exceed the maximum current output level, however, position errors or integral error faults may occur.

This parameter value can range from 0 % to 100 %. The default value for this parameter is 100 %. Refer to Table 5-53.

Table 5-53. Settings for Parameter x53

Param #	Axis #	Range	Default
153	1	0 % to 100 %	100 %
253	2	0 % to 100 %	100 %
353	3	0 % to 100 %	100 %
453	4	0 % to 100 %	100 %

This parameter provides a safety feature to prevent the peak currents from damaging the amplifiers and/or motors. Proper configuration of this parameter can help to avoid equipment damage, especially during system debug.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x54

5.7.9. Output Bit for AUX OUTPUT

This parameter is used to specify an output bit (1-8) that is activated (set low) if a programmable "AUX OUTPUT" fault condition (see x58) is met. This feature may be deactivated by entering 0. Refer to Section 12.1.6: Output Bus Specifications for technical details about the eight TTL outputs of the UNIDEX 500 including output lines, pin numbers, and electrical characteristics.

Parameter x54 can have a value that ranges from 0 to 8. These values are explained in Table 5-54.

Table 5-54. Settings for Parameter x54

Param #	Axis #	Values and Descriptions
154	1	0 disabled 1 - output bit 0 is activated on associated AUX OUTPUT faults (default for axis 1) 2 - output bit 1 is activated on associated AUX OUTPUT faults 3 - output bit 2 is activated on associated AUX OUTPUT faults 4 - output bit 3 is activated on associated AUX OUTPUT faults 5 - output bit 4 is activated on associated AUX OUTPUT faults 6 - output bit 5 is activated on associated AUX OUTPUT faults 7 - output bit 6 is activated on associated AUX OUTPUT faults 8 - output bit 7 is activated on associated AUX OUTPUT faults
254	2	*same as above* (default value is 2)
354	3	*same as above* (default value is 3)
454	4	*same as above* (default value is 4)



A reset of the UNIDEX 500 sets all output bits high (deactivated).

5.7.10. Drive Fault Signal Active Low

x70

Parameter x70 specifies the polarity of the amplifier fault input to the UNIDEX 500. This parameter must be configured to correspond to the signal in its active state.

In a normally open (active low) configuration (x70 = yes), a 5 volt signal represents a normal (non-fault) condition and a 0 volt signal indicates a drive fault condition. Conversely, in a normally closed (active high) configuration (x70 = no), a 0 volt signal represents a normal (non-fault) condition and a 5 volt signal indicates a drive fault condition. The settings for parameter x70 are shown in Table 5-55.

Table 5-55. Settings for Parameter x70

Param #	Axis #	Values	Description
170	1	Yes (**Y**)	Drive fault signal is active low (default)
		No (N)	Drive fault signal is active high
270	2	Yes (**Y**)	Drive fault signal is active low (default)
		No (N)	Drive fault signal is active high
370	3	Yes (**Y**)	Drive fault signal is active low (default)
		No (N)	Drive fault signal is active high
470	4	Yes (**Y**)	Drive fault signal is active low (default)
		No (N)	Drive fault signal is active high

5.7.11. AUX OUTPUT Active High

x84

Parameter x84 sets the active state of the auxiliary output bit. If this parameter is set to yes, the auxiliary output bit will be set high if a fault condition is set to activate the auxiliary output bit. The settings for this parameter are shown in Table 5-56.

Table 5-56. Settings for Parameter x84

Param #	Value	Description
x84	Yes (default)	AUX OUTPUT bit active high
	No	AUX OUTPUT bit active low

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

5.8. The Motor and Feedback Configuration Tab

The UNIDEX 500 utilizes several parameter settings for configuration based on the motor and drive type being used. This section provides an introduction to motor and feedback configuration. The motor and feedback configuration parameters are explained in detail in the sections that follow.

The parameters that are used to configure motor and feedback functions of the U500 depend on the type of motor that is being used. A list of motor types (Stepper, AC Brushless, and DC Servo), their related parameters, and default values are shown in Table 5-57.

Table 5-57. Motor Feedback Parameters

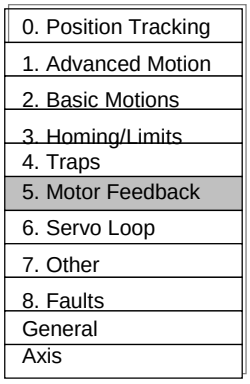
Param #	Description	Step	AC Brush	DC Servo	Default Value(s)
x38	*Primary/Position feedback channel*	✓	✓	✓	1, 2, 3, 4
x39	*Secondary/Velocity feedback channel*	✓	✓	✓	0
x40	*Primary feedback setup code*	✓	✓	✓	3
x41	*Secondary feedback setup code*	✓	✓	✓	3
x42	*Drive (motor) type (0-DC Brush, 1-AC Brushless, 2/3-Stepper)*	✓	✓	✓	0
x43	*AC brushless motor commutation factor (cycles/rev)*	N/A	✓	N/A	4
x44	*Encoder feedback (steps/rev/(* 4))*	✓	✓	N/A	4000
x45	*AC brushless motor phase offset (degrees)*	N/A	✓	N/A	0
x46	*Stepper high current (0-100%)*	✓	N/A	N/A	70
x47	*Stepper low current (0-100%)*	✓	N/A	N/A	35
x63	*Stepper microstepping resolution (steps/rev)*	✓	N/A	N/A	4000
x64	*Stepper encoder verification (y/n)*	✓	N/A	N/A	Yes
x65	*Stepper encoder speed (microstep/ms)*	✓	N/A	N/A	1.0
x66	*AC brushless motor phase advance, base speed*	N/A	✓	N/A	0
x67	*AC brushless motor base speed advance (degrees)*	N/A	✓	N/A	0
x68	*AC brushless motor phase speed (steps/ms)*	N/A	✓	N/A	0
x69	*AC brushless motor phase speed advance (degrees)*	N/A	✓	N/A	0
x79	*Primary current command offset (mV)*	✓	✓	✓	0
x80	*Secondary current command offset (mV)*	✓	✓	✓	0
x82	*Encoder multiplication*	✓	✓	✓	0



IMPORTANT

After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. This is done using the F9 RESET button located at the bottom of the main window.

5.8.1. Introduction to Motors and Feedback Configurations



Parameters in the Motor Feedback tab are used to configure the motors that control the system's axes. There are three types of motors that are typically used for control. These include stepper motors, AC brushless motors (including linear motors), and DC brush servo motors. Some of the parameters in this tab apply to all motor configurations, while some are valid only for certain types of motors.

Motors may or may not be used in conjunction with a feedback device. In open-loop configurations, a feedback device is not used. Such configurations make the assumption that the axis will attain its commanded position without any feedback or verification. This is typically seen in stepper motor applications. Conversely, a closed-loop system uses a feedback device to verify the position of the axis. Such configurations compare the desired axis position with the actual position from the feedback device. Two common feedback devices are encoders and resolvers. An encoder is a rotary mechanism that transmits a pulsed signal based on the number of revolutions of the device. A resolver is a two-phase, rotary, electromagnetic transducer in which inductive coupling (between the rotor and stator windings) and trigonometric principles are used to provide absolute position information over one electrical cycle.

In rotary motors and feedback devices, it is important to determine the rotation direction for proper orientation and configuration. A motor is said to be rotating in the positive (+) direction if the shaft is turning in the clockwise (CW) direction while looking "into" the motor from the shaft end. In this case, the feedback device should be counting in the positive direction. Refer to Figure 5-17.

Reversing the SIN+ and SIN- encoder signals can change the encoder direction. Similarly, the direction of a resolver can be changed by reversing the SIN+ and SIN- feedback signals. Use this to correct a feedback phasing problem.



Each axis can be connected to two feedback devices: one for position (the primary feedback device) and the other for velocity (the secondary feedback device). Each of the feedback devices also has setup parameters that specify the transducer type, location, resolution, and mode of operation.

The secondary feedback channel and setup code parameters need to be configured only for dual loop applications. For all other applications, these parameters should be set to zero.



Multiple axes must not be configured for feedback from the same channel, otherwise improper operation will result.



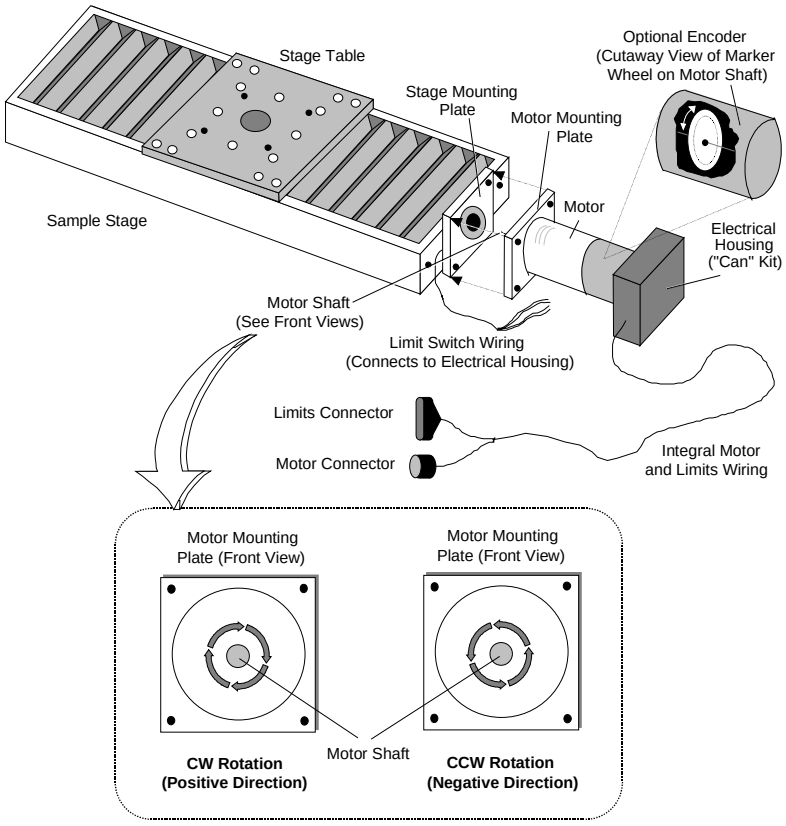


Figure 5-17. Motor and Encoder Rotation

Encoder feedback channels are RS-422 differential quadrature signals. Channels 1-4 are located on the UNIDEX 500 main board. The U500 automatically multiplies the fundamental encoder line count by four. Conversion to user units is done using axis parameters x00 (*Metric conversion factor*) and x01 (*English conversion factor*).

A U500 resolver-to-digital converter board (U500 RDP) must be installed and configured for applications requiring resolver feedback. The U500 can accept two RDP boards. The first contains feedback channels 9-12, while the second contains feedback channels 13-16.

Normally, one feedback device is used per axis, however in some applications, a dual-loop setup may provide greater control. In this case, one transducer provides the position feedback and a separate transducer provides velocity feedback. The user must specify the channel of both transducers in the setup parameters. The channel inherently specifies the feedback device type. The setup code specifies the resolution and mode of operation of the feedback device.

Commutation information in the case of AC brushless motors comes from the velocity feedback transducer. Encoder counts per revolution should be entered for the velocity feedback transducer.

English and Metric conversion factors are calculated with respect to the position feedback resolution.



As mentioned earlier, the three most common types of motors are stepper motors, AC brushless motors, and DC brush servo motors. For stepper motor applications, parameter x42 (*Drive type*) must be configured to either 2 or 3. Most stepper motor applications are open loop applications (that is, they have no feedback). As such, the commanded position is the assumed motor position.

Be sure that the DR500 is properly configured for the selected amplifier. Refer to the *DR500 Operation and Technical Manual (P/N EDA120)* for more information.



Stepper Motors

U500 Stepper Motor Control

The UNIDEX 500 ISAcan drive up to four stepper motors. To drive the motors, two current command phases are output, separated by 90 electrical degrees. For typical open-loop operation, the UNIDEX 500 generates 2048 micro-machine steps per pole of the motor. This equates to 102,400 machine steps per revolution (2048 x 50 = 102,400). The maximum commutation frequency for stepper motors is 2500 Hz yielding 3000 rpm maximum. In open-loop applications, the stepper motor uses the home limit switch as a reference point during home cycles. A marker wheel or encoder may be used to provide a more repeatable home cycle reference.

If parameter x38 (*Primary/Position feedback channel*) is configured for an encoder (i.e., x38=1, 2, 3, ..., or 8), then the UNIDEX 500 will stop on the marker pulse during home cycles.



The UNIDEX 500 checks for encoder feedback if parameter x38 (*Primary/Position feedback channel*) is set for an encoder (i.e., x38=1, 2, 3, ..., or 8), and parameter x44 (*Encoder feedback [steps/rev/(* 4)]*) is non-zero. If x38 is set for an encoder, but parameter x44=0, then the UNIDEX 500 assumes that a marker wheel is attached. In this case, the marker wheel is used during the home cycle.

For encoder verification applications, the UNIDEX 500 scales the microsteps to encoder counts. The U500 uses parameter x44 (*Encoder feedback [steps/rev/(* 4)]*) to determine the number of microsteps per revolution. This value must be evenly divisible by 50 (poles per revolution). The UNIDEX 500 automatically multiplies the SIN/COS signals by four.



When using stepper motors, the motor torque must be high enough to prevent motor stall or drop out. This can be detected by attaching an encoder to the stepper motor and entering a value into parameter x19 (*Max position error [0-8,388,607]*). This value specifies the maximum allowable encoder count error between the commanded motor position and the actual position. If the difference between these two positions exceeds the value set in parameter x19, the axis generates a fault condition.

The UNIDEX 500 provides a feature called dynamic current scaling in applications using stepper motors. The UNIDEX 500 changes the stepper motor current level based on the commanded velocity. If the commanded velocity is zero for a duration of 500 ms, the current level goes to a programmable low value (axis parameter x47 - *Stepper low current [0-100%]*). Once a move is commanded, the current level immediately goes to the high value (axis parameter x46 - *Stepper high current [0-100%]*).

U500 CLK/DIR Outputs (U500 PCI ONLY)

The U500 PCI BASE/PLUS controller can control up to 4 axes of servo or stepper in any combination. The stepper axes may be dual current command (amplifier type 2,3) or CLK/DIR output (amplifier type 4). The U500PCI Ultra can control up to 8 axes, 4 servos as axis 1-4 and 4 CLK/DIR outputs as axis 5-8. Axis 1-4 may be configured for CLK/DIR, but only 4 CLK/DIR channels total are available. For example, if Axis 1 is configured as CLK/DIR, axis 5 cannot be used. Axis 5-8 CLK/DIR signal are located on the P9 connector.

Axis 1-4 can be individually configured for CLK/DIR output by setting parameter x42 (*Drive type*) 4 and setting jumpers JP6/JP7 . See Chapter 12 for information on setting jumpers JP6/JP7on the U500 PCI. After setting these jumpers to configure an axis for CLK/DIR, the CLK signal will be generated in place of the ICMDB current command, and the DIR signal will be generated in place of the ICMDA current command output for that axis.

Any axis that is configured for CLK/DIR output will only accept the following U500 Commands: INDEX, HOME, ENABLE, DISABLE, ERROR.

AC Brushless Motor with Encoder

Another common type of motor is an AC brushless motor. For AC brushless motor applications, parameter x42 (*Drive type*) must be configured to 1 (AC brushless). The UNIDEX 500 can drive up to four AC brushless motors. Two DAC current command channels are output, separated by 120 electrical degrees and updated at a rate of 4 kHz.



Be sure that the DR500 is properly configured for the selected amplifier. Refer to the DR500 Operation and Technical Manual (P/N EDA120) for more information.

For closed-loop operation using an encoder for feedback, parameter x38 (*Primary/Position feedback channel*) must be set to x38=1, 2, 3, ..., or 8. Hall effect signals are assumed to be present and are read on the upper 12 bits of the user input bus (three per axis). The Hall sensors are used for six-step and sinusoidal commutation.

Encoder channels 5-8 are used for the 4EN encoder interface card. The 4EN encoder card can only be used for velocity loop feedback.



To configure the UNIDEX 500 with an AC brushless motor for commutation exclusively from the six step Hall sensors, set parameter x43 (*AC brushless motor commutation factor [cycles/rev]*) to zero.



Inputs IN0-IN3 are always reserved for general-purpose user inputs. Other inputs that are not being used for Hall sensors are available as general-purpose inputs. These are illustrated in Table 5-58.

Table 5-58. Summary of General Purpose Inputs

U500 Input Bus	Functions
IN4 - IN6	Axis 1 Hall inputs or general purpose inputs
IN7 - IN9	Axis 2 Hall inputs or general purpose inputs
IN10 - IN12	Axis 3 Hall inputs or general purpose inputs
IN13 - IN15	Axis 4 Hall inputs or general purpose inputs

The U500 PCI board can optionally run the hall inputs through the 100-pin connector instead of using the input lines. See Option Board Setup Code section 5.4.13.

An invalid Hall state (“000” or “111”) generates a feedback fault.



If the commutation factor is non-zero, the UNIDEX 500 will switch to sinusoidal commutation on the first Hall state transition encountered after the axis is enabled. The number of encoder counts per revolution (parameter x44) is used to generate the proper sinusoidal commutation signals.

AC Brushless Motor with Resolver

The UNIDEX 500 uses the resolver to determine the initial rotor position and all subsequent commutation. Parameter x43 (*AC brushless motor commutation factor [cycles/rev]*) and parameter x44 (*Encoder feedback [steps/rev/(* 4)]*) must be configured appropriately.

The commutation factor is the number of electrical cycles per motor revolution. Commutation factors for 4, 6, and 8 poles are shown in Table 5-59.

Table 5-59. Commutation Factors for 4, 6, and 8 Poles

Number of Poles	Commutation Factor
4 pole	2 cycles
6 pole	3 cycles
8 pole	4 cycles

Parameter x38 (*Primary/Position feedback channel*) must be set for a resolver channel (9-16). Parameter x40 (*Primary feedback setup code*) must specify the proper resolution of the RDP. The UNIDEX 500 RDP board must be factory configured for the proper system resolution. Use Table 5-60 to ensure proper configuration.

Table 5-60. Factory Configuration for UNIDEX 500 RDP

RDP Resolution (bits)	Counts per Revolution	Setup Code
10	1,024	1
12	4,096	2
14	16,384	3
16	65,536	4
16/14 Dynamic	65,536	5

The UNIDEX 500 uses the resolver to determine the initial rotor position and all subsequent commutation. Parameters x43 (*AC brushless motor commutation factor [cycles/rev]*) and x44 (*Encoder feedback [steps/rev/(* 4)]*) must be configured appropriately.

The phase currents may be offset from the reference position by setting parameter x45 (*AC brushless motor phase offset [degrees]*).



The UNIDEX 500 will reference to the resolver null during a home cycle.

Improper phasing of AC brushless motors may cause the system to fault when the axis is enabled or when motion is attempted. A fixed relationship exists between the feedback device and the generated phase currents.

DC Brush Motor

Another common type of motor is a DC brush motor. Prior to using a DC brush motor, parameter x42 (*Drive type*) must be configured to "0-DC Brush." The UNIDEX 500 supplies the DC brush motor one current (torque) command voltage.

Be sure that the DR500 is properly configured for the selected amplifier. Refer to the DR500 Operation and Technical Manual (P/N EDA120) for more information.



The DC brush motor can use any feedback channel. Parameter x39 (*Secondary/Velocity feedback channel*) and parameter x41 (*Secondary feedback setup code*) need only be set for "dual-loop" type applications.

For encoder operation, parameter x40 (*Primary feedback setup code*) should be set to 0. The UNIDEX 500 will reference to the marker during the home cycle.



For resolver operation, parameter x38 (*Primary/Position feedback channel*) must be set for a resolver channel (9-16). Parameter x40 (*Primary feedback setup code*) must specify the proper resolution of the RDP and the UNIDEX 500 RDP must be installed.

The UNIDEX 500 RDP board must be factory configured for the proper system resolution. Configuration information is shown in the following three tables.

Table 5-61. Feedback Channels, Types, and Additional Hardware

Feedback Channel	Feedback Type	Additional Hardware
0	Open Loop	None - for stepper only
1-4	Encoder	None - UNIDEX 500 main board
5-8	Encoder	4EN option board / U500 PCI Ultra
9-12	Resolver	U500 RDP board #1
13-16	Resolver	U500 RDP board #2
17 - 18	Laser	Requires RMX-PC board #1 (x 512 [λ/1024] resolution)
19 - 20	Laser	Requires RMX-PC board #2 (x 512 [λ/1024] resolution)
21 - 22	Laser	Requires RMX-PC board #1 (x 4 [λ/32] resolution)
23 - 24	Laser	Requires RMX-PC board #2 (x 4 [λ/32] resolution)

Table 5-62. Feedback Setup Codes, and the Associated Actions

Feedback Setup Code	Action
0	No velocity loop transducer
1	Resolver 10 bit mode
2	Resolver 12 bit mode
3	Resolver 14 bit mode
4	Resolver 16 bit mode
5	Resolver dynamic resolution mode (14-16 bit)

Table 5-63. RDP Resolution and Setup Codes

RDP Resolution (bits)	Counts per Revolution	Setup Code
10	1,024	1
12	4,096	2
14	16,384	3
16	65,536	4
16/14 dynamic	65,536	5



The UNIDEX 500 will reference the resolver null during the home cycle.

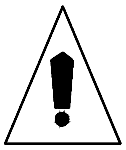
5.8.2. Motor Feedback and Configuration Parameters

The motor feedback and configuration parameters define the types of motors and the types of feedback devices (if any) that are being used. There are 20 parameters in the motor feedback and configuration parameter group. These parameters are listed in Table 5-64 and explained in detail in this section.

Table 5-64. Motor Feedback and Configuration Parameters

Param #	Description	Default Value(s)
x38	*Primary/Position feedback channel*	1, 2, 3, 4
x39	*Secondary/Velocity feedback channel*	0
x40	*Primary feedback setup code*	3
x41	*Secondary feedback setup code*	3
x42	*Drive type (0-DC Brush, 1-AC Brushless, 2/3-Stepper)*	0
x43	*AC brushless motor commutation factor (cycles/rev)*	4
x44	*Encoder feedback (steps/rev/(* 4))*	4000
x45	*AC brushless motor phase offset (degrees)*	0
x46	*Stepper high current (0-100%)*	70
x47	*Stepper low current (0-100%)*	35
x63	*Stepper microstepping resolution (steps/rev)*	4000
x64	*Stepper encoder verification (y/n)*	Yes
x65	*Stepper encoder speed (microstep/ms)*	1.0
x66	*AC brushless motor phase advance, base speed (steps/ms)*	0
x67	*AC brushless motor base speed advance (degrees)*	0
x68	*AC brushless motor phase speed (steps/ms)*	0
x69	*AC brushless motor phase speed advance (degrees)*	0
x79	*Primary current command offset (mV)*	0
x80	*Secondary current command offset (mV)*	0
x82	*Encoder multiplication*	0

After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. Use the F9 RESET button located at the bottom of the main window.



If the DR500 drive rack and Aerotech motors were purchased with the UNIDEX 500 controller, all applicable parameters are set to provide optimum performance.



5.8.3. Primary/Position Feedback Channel

This parameter is used to configure the channel of the primary feedback device being used. The parameter value is a code that corresponds to a particular feedback device for each axis (1-4). Feedback channels, their respective feedback types, and additional hardware requirements are summarized in Table 5-65.

Table 5-65. Settings for Parameter x38

Feedback Channel	Feedback Type	Additional Hardware Required
0	Open Loop	None
1 - 4	Encoder	None (UNIDEX 500 main board only) (defaults for axes 1-4)
5 - 8	Encoder	4EN / U500 PCI Ultra*
9 - 12	Resolver	Requires RDP-PC board #1
13 - 16	Resolver	Requires RDP-PC board #2
17 - 18	Laser	Requires RMX-PC board #1 (x 512 [λ/1024] resolution)
19 - 20	Laser	Requires RMX-PC board #2 (x 512 [λ/1024] resolution)
21 - 22	Laser	Requires RMX-PC board #1 (x 4 [λ/32] resolution)
23 - 24	Laser	Requires RMX-PC board #2 (x 4 [λ/32] resolution)
25 - 28	Hall signals	Hall effect position feedback (spindle with “000” and “111” hall states)
29 - 32	Encoder (Gantry)	These channels correspond to axis 1-4 actual feedback. For example, to make axis 3's servo loop identical to axis 1's, set axis 3's feedback channel to 29. Axis 3's servo loop will use the EXACT same velocity command and feedback that axis 1 used. All servo loop parameters and traps should be set the same. All motion and home parameters are based on the master axis.
33 - 36	Hall signals	Hall effect position feedback (standard – spindle without “000” and “111” hall states)
37 - 40	Encoder (Gantry)	These channels specify a slave axis that will read command position from a master axis on channel 1-4 respectively. In this case, feedback is from its own encoder channel. The axis will follow the home cycle marker search and home offsets of the master axis.
41 - 44	Encoder (Gantry)	These channels specify a slave axis that will read command position from a master axis on channel 1-4. Feedback is from its own encoder channel. The home cycle marker search and home offsets are separate from the master axis.
45 – 48	12 bit A/D	Optional 12 bit A/D converter on the U500.

* To Use Encoder channels 5-8 on the U500 PCI Ultra, FPGA #2 must be configured for the option “Encoder 5-8”. Refer to Chapter 4, Section 4.3.4 for details.

Feedback should always be verified before enabling the axis.



5.8.3.1. Encoder Gantry Mode

When an axis has its Position Feedback Channel (parameter 0x38) set to 41-44, it is linked as a slave to a master axis 1-4 (41-44). When the master axis is enabled, the slave axis is also enabled, and all motion commanded to the master will be commanded to the slave. The master and slave get feedback individually from their own encoder channels, therefore the servo loops are independent. The master and slave axes use their own parameters for servo loop setup. The master and slave axes perform separate marker searches during the home cycle. This is done to allow orthogonality adjustment of the gantry by setting the Home Offset parameter (x06).

Gantry home cycle sequence:

1)	Master and slave move in home direction until limit switch is found.	linked
2)	Limit switch is debounced	linked
3)	Execute “Limit switch to marker” offset (U500ISA only)	linked
4)	Begin marker search	un-linked
5)	Master finds marker, clears internal positon, keeps moving until slave finds its marker	un-linked
6)	Slave finds marker, clears internal position, keeps moving unitl master finds its marker.	un-linked
7)	When master and slave have each found their markers, both axes decelerate to a stop and wait for 1 second. The diagnostic window now contains the distance from each marker.	un-linked
8)	Each axis executes a move equal to the home offset (x06) minus the distance to the marker from step 7) above.	un-linked
9)	Master and slave internal positions are set to 0 and home cycle is complete.	linked

If the Home Offset parameter is set to 0 for both axes, each will attempt to move to the marker position. This move is the value read in Step 7, above, but in the opposite direction. The initial setting for the home offset parameter for each axis should be the value read in Step 7 (without direction inversion). The home cycle can be aborted during the 1 second dwell in Step 7 (by pressing the abort key F2) to make it easier to observe the offset numbers.

Home cycle parameters for the master and slave axes should be set the same. This includes home feedrates, accel/decel rate, home direction, etc. The slave axis should not be given any direct motion commands.

Table 5-66. Encoder Gantry Mode Configuration Parameters

Parameter	Setting	Description	Slave Feedback
138	41	*not allowed*	encoder channel 1
138	42	axis 1 is slave, axis 2 is master	encoder channel 1
138	43	axis 1 is slave, axis 3 is master	encoder channel 1
138	44	axis 1 is slave, axis 4 is master	encoder channel 1
238	41	axis 2 is slave, axis 1 is master	encoder channel 2
238	42	*not allowed*	encoder channel 2
238	43	axis 2 is slave, axis 3 is master	encoder channel 2
238	44	axis 2 is slave, axis 4 is master	encoder channel 2
338	41	axis 3 is slave, axis 1 is master	encoder channel 3
338	42	axis 3 is slave, axis 2 is master	encoder channel 3
338	43	*not allowed*	encoder channel 3
338	44	axis 3 is slave, axis 4 is master	encoder channel 3
438	41	axis 4 is slave, axis 1 is master	encoder channel 4
438	42	axis 4 is slave, axis 2 is master	encoder channel 4
438	43	axis 4 is slave, axis 3 is master	encoder channel 4
438	44	*not allowed*	encoder channel 4

5.8.4. Secondary/Velocity Feedback Channel

This parameter is used to configure the secondary feedback channel of UNIDEX 500. The parameter value is a code that corresponds to a particular feedback device for each axis (1-4). Feedback channels, their respective feedback types, and additional hardware requirements are summarized in Table 5-67.

Table 5-67. Settings for Parameter x39

Feedback Channel	Feedback Type	Additional Hardware Required
0	Open Loop	None (for stepper motors only)
1 - 4	Encoder	None (UNIDEX 500 main board only) (defaults for axes 1-4)
5 - 8	Encoder*	Requires 4EN option board / U500 PCI Ultra
9 - 12	Resolver	Requires RDP-PC board #1
13 - 16	Resolver	Requires RDP-PC board #2
17 - 18	Laser	Requires RMX-PC board #1 (x 512 [λ/1024] resolution)
19 - 20	Laser	Requires RMX-PC board #2 (x 512 [λ/1024] resolution)
21 - 22	Laser	Requires RMX-PC board #1 (x 4 [λ/32] resolution)
23 - 24	Laser	Requires RMX-PC board #2 (x 4 [λ/32] resolution)

* To Use Encoder channels 5-8 on the U500 PCI Ultra, FPGA #2 must be configured for the option “Encoder 5-8”. Refer to Chapter 4, Section 4.3.3 for details.

If this parameter is configured incorrectly, sporadic operation may occur.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x40

5.8.5. Primary Feedback Setup Code

This parameter specifies a code that corresponds to the bit resolution mode whenever a resolver is being used as the primary feedback device. This parameter has a range from 0-5. The meanings for these setup codes are listed in Table 5-68.

Table 5-68. Settings for Parameter x40

Feedback Setup Code	Action	RDP Resolution (bits)	Counts per Revolution
0	N/A	N/A	N/A
1	Resolver 10-bit mode	10 bits	1,024
2	Resolver 12-bit mode	12 bits	4,096
3	Resolver 14-bit mode	14 bits (default)	16,384
4	Resolver 16-bit mode	16 bits	65,536
5	Resolver dynamic resolution mode	16/14 bits dynamic	65,536/ 16,384



The UNIDEX 500's RDP hardware must be configured for the same resolution.

x41

5.8.6. Secondary Feedback Setup Code

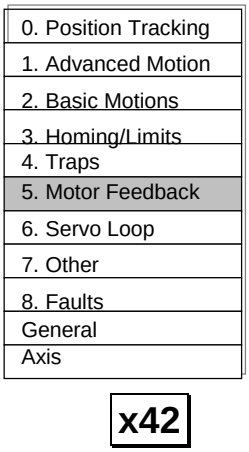
This parameter specifies a code that corresponds to the bit resolution mode whenever a resolver is being used as the secondary feedback device. This parameter has a range from 0-5. The meanings for these setup codes are listed in Table 5-69.

Table 5-69. Settings for Parameter x41

Feedback Setup Code	Action	RDP Resolution (bits)	Counts per Revolution
0	N/A	N/A	N/A
1	Resolver 10-bit mode	10 bits	1,024
2	Resolver 12-bit mode	12 bits	4,096
3	Resolver 14-bit mode	14 bits (default)	16,384
4	Resolver 16-bit mode	16 bits	65,536
5	Resolver dynamic resolution mode	16/14 bits dynamic	65,536/ 16,384



The UNIDEX 500's RDP hardware must be configured for the same resolution.



5.8.7. Drive (Motor) Type (0-DC Brush, 1-AC Brushless, 2/3-Stepper)

This parameter is used to configure the UNIDEX 500 for the type of motor being used. Amplifier Type 0 (DC brush) causes the U500 to output a single current command. This should be used with brush motors or with self commutating amplifiers. Amplifier Type 1 (AC brushless) cause the U500 to output two current commands separated by 120°. This setting should be used when the U500 is commutating the motor. Amplifier Type 2 (Stepper amplifier) causes the U500 to output 2 commutation signals separated by 90°. Amplifier Type 3 (Stepper amplifier - recirculation mode) is the same as Type 2 except that a recirculation mode is entered to reduce motor heating (requires AM8007 or AM16007). Amplifier Type 4 (CLK/DIR) causes the U500 PCI to output TTL pulses representing desired motion.

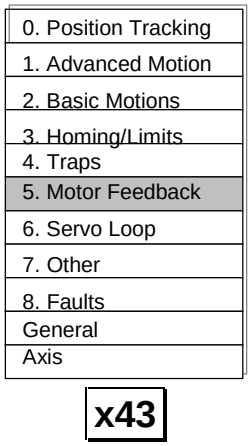
This parameter has a range from 0-4 with the drive types listed in Table 5-70.

Table 5-70. Settings for Parameter x42

Drive Type Code	Drive Type
0	DC brush (default)
1	AC brushless (including linear drives)
2	Stepper amplifier - no recirculation
3	Stepper amplifier - recirculation mode
4	CLK/DIR – U500 PCI

Improper configuration of this parameter will cause the motor to "trap" when it is enabled or when motion is commanded.





5.8.8. AC Brushless Motor Commutation Factor (Cycles/Rev)

This parameter is used to configure the UNIDEX 500 for the number of electrical cycles per motor revolution of the feedback device. The value of this parameter, in conjunction with the value of axis parameter x44 (*Encoder feedback [steps/rev/(* 4)]*), are used to generate the proper sinusoidal phase currents. This parameter should be set to one for linear motors. Refer to Table 5-71.

Table 5-71. Sample Commutation Factors for AC Brushless Motors

Commutation Factor	# of Poles	Cycles per Revolution
1	2 pole	1 cycle/rev
2	4 pole	2 cycles/rev
3	6 pole	3 cycles/rev
4 (default)	8 pole	4 cycles/rev

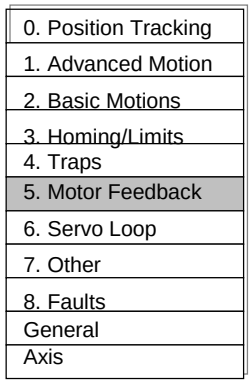


Setting this value to 0 for an AC brushless motor with encoder feedback will result in six step commutation.

This parameter has a range from 0 to 65,536 with a default value of 4. Sample commutation factors are shown above in Table 5-71.



Improper configuration will cause a "trap" when a move is commanded.

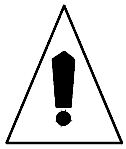


5.8.9. Encoder Feedback (Steps/Rev/(*4))

x44

The value of this parameter, in conjunction with the value of axis parameter x43 (*AC brushless motor commutation factor [cycles/rev]*), is used to generate the proper sinusoidal phase currents. This parameter applies to all servo motors and stepper axes with encoder verification. It does not apply to DC brush motors.

This parameter has a range from 0 to 2^{48}. The system default value is 4000 for a 1000 line encoder.



Improper configuration will cause a "trap" when a move is commanded. Also, the value of this parameter should be evenly divisible by 50 (poles per revolution) for stepper motors.

5.8.10. AC Brushless Motor Phase Offset (Degrees)

x45

Some AC brushless motors may require a phasing relationship that is different than the one provided by the UNIDEX 500. The default phasing may be adjusted by the configuration of parameter x45.

This parameter has a range from 0-359°. All Aerotech amplifiers require a 0° offset except for the AS3005, which requires a 300° offset.

Refer to the previous section for additional phasing information.

The AS3005 amplifier requires an offset of 300°.



- 0. Position Tracking
- 1. Advanced Motion
- 2. Basic Motions
- 3. Homing/Limits
- 4. Traps
- 5. Motor Feedback
- 6. Servo Loop
- 7. Other
- 8. Faults
- General
- Axis

5.8.11. Stepper High Current (0-100%)

x46

The UNIDEX 500 utilizes dynamic current scaling based on the motor's commanded velocity. If an axis is setup as a stepper and a motion is commanded, the UNIDEX 500 will output the percentage (set by this parameter) of the maximum output voltage (+/- 10 volts). For example, a value of 100% corresponds to 10 volts. This value is the peak of the sinusoidal current command during motion.

This parameter has a range from 0% to 100%. The system default is 70%.



Actual motor current depends on the amplifier's scaling. For the AM8007 and AM16007 amplifiers, the scaling is 7 amps/10 volts.

5.8.12. Stepper Low Current (0-100%)

x47

The UNIDEX 500 utilizes dynamic current scaling based on the motor's commanded velocity. If the commanded velocity is zero for 500 ms, the current level will go to the value set by this parameter, reducing motor heating.

This parameter sets the percentage of the maximum output voltage that the UNIDEX 500 can generate (+/-10 volts). This value is taken from the peak of the sinusoidal current command while in position.

This parameter has a range from 0% to 100 %. The system default is 35%.



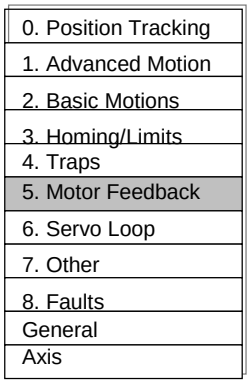
Actual motor current depends on the amplifier's scaling.

5.8.13. Stepper Microstepping Resolution (Steps/Rev)

x63

Parameter x63 sets the microstepping resolution of an open loop stepper. This value is specified in microsteps per revolution.

This parameter has a range from 200 to 102,400 microsteps per revolution. The system default is 4000 microsteps per revolution.



x64

5.8.14. Stepper Encoder Verification (Y/N)

Parameter x64 specifies whether or not encoder verification is enabled for each axis that is configured as an open loop stepper.

This parameter can have the values listed in Table 5-72.

Table 5-72. Settings for Parameter x64

Param #	Axis #	Settings
164	1	Yes - Encoder verification enabled for axis 1 (default) No - No encoder verification for axis 1
264	2	Yes - Encoder verification enabled for axis 2 (default) No - No encoder verification for axis 2
364	3	Yes - Encoder verification enabled for axis 3 (default) No - No encoder verification for axis 3
464	4	Yes - Encoder verification enabled for axis 4 (default) No - No encoder verification for axis 4

5.8.15. Stepper Encoder Speed (Microsteps/ms)

x65

Parameter x65 specifies a correction speed in microsteps per millisecond for each axis that is configured as an open loop stepper.

This range for this parameter value is from 1 to 8,388,607. The system default is 1 microsteps/ms.

5.8.16. AC Brushless Motor Phase Advance Base Speed (Steps/ms)

At top speeds, the motor's back EMF (Kb) limits the amount of current that can be driven into the motor. This occurs when the generated back EMF is near the bus voltage of the amplifier.

Phase advance is used to increase the usable speed of an AC brushless motor. It does this by a technique called "field weaking." The effective torque angle of the motor is advanced from 90 degrees at high speeds, thus reducing the motors back EMF. This allows more current to be driven into the motor for a given bus voltage. The phase advance characteristics curve is specified by four parameters. The first two, x66 and x67, specify the slope of the first section (the base speed). Parameters x68 and x69 specify the slope of the second section (the phase speed). The example in Figure 5-18 illustrates the phase advance slope for an application where a phase advance of 10° at 1200 rpm and 30° at 3000 rpm is needed.

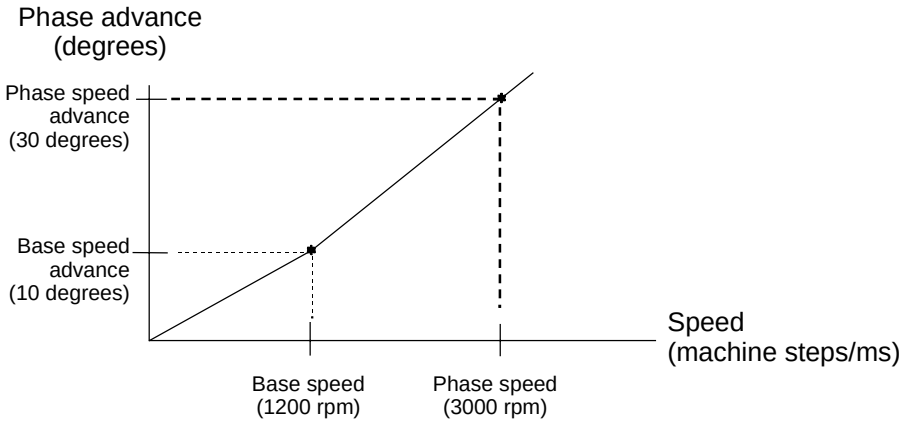


Figure 5-18. Phase Advance Slope

Parameter x66 is the base speed specified by the user in machine steps/ms. The range is a value from 0 to 8,388,608. The system default is zero (0).

The phase advance does not work with DC or stepper motors. Also, the function will not increase torque at low speeds.

The UNIDEX 500 clamps the maximum allowed phase advance at 40°.

5.8.17. AC Brushless Motor Base Speed Advance (Degrees)

x67

Parameter x67 is the phase advance in degrees at the specified base speed. Working in conjunction with parameter x66 this parameter is the number of degrees that the torque angle is increased beyond 90° at the speed set in parameter x66.

The range for this parameter value is from 0 to 359 degrees. The system default is zero (0).

5.8.18. AC Brushless Motor Phase Speed (steps/ms)

x68

Parameter x68 is the phase speed specified by the user in machine steps / ms. The range for parameter x68 is a value from 0 to 8,388,608. The system default is zero (0).

5.8.19. AC Brushless Motor Phase Speed Advance (Degrees)

x69

Parameter x69 is the phase advance in degrees at the specified phase speed. Working in conjunction with parameter x68 this parameter is the number of degrees that the torque angle is increased beyond 90° at the speed set in parameter x68.

The range for this parameter value is from 0 to 359 degrees. The system default is zero (0).

5.8.20. Current Command Offset Parameters (mV)

x79

x80

Parameters x79 and x80 provide a DC offset to the generated primary and secondary current command. They may be used in tachometer or AC servo applications to null-out the digital/analog (D/A) converters. The values for these parameters are specified in mV and can range from -10,000 to +10,000 mV. The default value is zero. Refer to Table 5-73 for a definition of these parameters.

Table 5-73. Settings for Parameters x79 and x80

Parameter	Definition	Range (mV)	Default Value
x79	Primary current command offset	-10,000 to 10,000	0 mV
x80	Secondary current command offset	-10,000 to 10,000	0 mV

5.8.21. Encoder Multiplication

x82

Parameter x82 can be used to change the position feedback resolution. This number multiplies encoder counts from the position loop encoder. This should be used only in dual loop applications in which the position loop resolution is less than the velocity loop resolution. Setting the parameter to -1 reverses the polarity of the encoder. The default value of 0 will not change the resolution. Settings for parameter x82 are given in Table 5-74.

Table 5-74. Settings for Parameter x82

Parameter #	Range	Default Value
x82	-8388607 - 8388607	0

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

5.9. The Servo Loops Tab

The servo loops parameters are used to configure and tune the servo control loops of the UNIDEX 500 system. There are 15 parameters in the servo loops parameter group. These parameters are listed in Table 5-75 and explained in detail in this section.

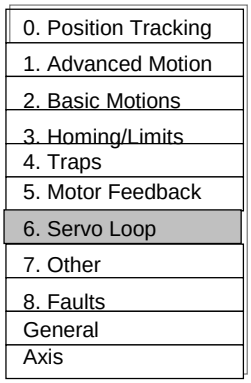
Table 5-75. Servo Loop Parameters (Tab 6)

Param #	Description	Default Value
x24	*Notch filter (y/n)*	No
x25	*Kpos (0-8,388,607)*	50
x26	*Ki (0-8,388,607)*	5000
x27	*Kp (0-8,388,607)*	100,000
x28	*Vff (0-8,388,607)*	256
x29	*Aff (0-8,388,607)*	0
x30	*Notch filter coefficient N0*	0.0
x31	*Notch filter coefficient N1*	0.0
x32	*Notch filter coefficient N2*	0.0
x33	*Notch filter coefficient D1*	0.0
x34	*Notch filter coefficient D2*	0.0
x36	*Integral error clamp (0-100%)*	49.2188 %
x62	*Servo loop update rate (1-100) x 0.25 ms*	1
x78	*Servo loop type*	0
x83	*Filter time constant*	0



After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. This is done using the F9 RESET button located at the bottom of the main software window.

Additional information is available in Chapter 6: Tuning Servo Loops.



5.9.1. Notch Filter (Y/N)

The notch filter enable parameter (x24) specifies whether or not notch or low pass filtering, is enabled for each axis of the system. This parameter can have one of two possible settings: yes or no. Setting parameter x24 to "yes" enables notch or low pass filtering. Setting parameter x24 to "no" (the default setting) disables notch/low pass filtering. Refer to Table 5-76.

Table 5-76. Settings for Parameter x24

Parameter Number	Axis Number	Value	Meaning
124	1	Yes	Notch/low pass filtering is active
		No	Filtering is disabled (default)
224	2	Yes	Notch/low pass filtering is active
		No	Filtering is disabled (default)
324	3	Yes	Notch/low pass filtering is active
		No	Filtering is disabled (default)
424	4	Yes	Notch/low pass filtering is active
		No	Filtering is disabled (default)

5.9.2. Kpos (0-8,388,607)(Position Loop Gain)



The Kpos value represents the position loop gain segment of the servo loop. This gain setting produces an output directly proportional to the position error, thus producing a constant counteracting force to the error. Parameter x25 can have a value ranging from 0 to 8,388,607 and defaults to 50.

A Kpos value that is too large may cause low frequency oscillation.



5.9.3. Ki (0-8,388,607)(Velocity Loop Integrator)



The Ki value represents the integral gain portion of the servo loop. The integral gain value produces an output, which is a summation of the velocity errors, producing an increasing counteracting force for an increasing position error.

This parameter can range from 0 to 8,388,607. The system default setting is 5000.

A Ki value that is too large may cause high frequency oscillation.



0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x27

x28

x29

5.9.4. Kp (0-8,388,607)(Velocity Loop Proportional Gain)

Parameter x27, Kp value, represents the proportional gain of the velocity loop, which is the inner loop portion of the dual control loop. This setting serves to dampen system response by producing a dampening force as long as the system is progressing toward error reduction.

This parameter can have a range from 0 to 8,388,607. The system default setting for this parameter is 100,000.

A Kp value that is too large may cause high frequency oscillation.

5.9.5. Vff (0-8,388,607)(Velocity Feed Forward)

The Vff value (parameter x28) represents velocity feed forward. The velocity feed forward bypasses the position portion of the control loop. Velocity commands are sent directly to the velocity loop, resulting in a reduction of position errors.

Resolution differences between position and velocity feedback transducers may be compensated for by the proper configuration of this parameter. The actual scaling of the feed forward command is Vff/256, where Vff is the value of axis parameter x28.

This parameter's value can range from 0 to 8,388,607. The system default setting for x28 is 256. This gives an actual scale factor of 1 (i.e., 256/256=1).



If no secondary feedback channel is specified, this parameter should be set to 256.

5.9.6. Aff (0-8,388,607)(Acceleration Feed Forward)

The Aff value represents acceleration feed forward. The acceleration feed forward value attempts to eliminate servo loop errors during acceleration and deceleration. It accomplishes this by sending a portion of the commanded acceleration/deceleration to the motor directly.

This parameter has a range from 0 to 8,388,607. The default setting is 0.



This parameter helps to eliminate errors during acceleration/deceleration only.

5.9.7. Notch Filter Coefficient N0, N1, N2, D1, and D2

Parameters x30 through x34 represent filter coefficients N0, N1, N2, D1 and D2 of a second order difference equation. Filter coefficients N0, N1, N2, D1, and D2 can be used to implement a notch filter or a second order low pass filter. These are described in the following sections.

These parameters have a range from -2.0 to 2.0. The system default setting is 0 for no notch filtering.

When one of these parameters is selected, a button labeled "calculate" will appear in the parameter editor window. This button will launch a utility that will automatically calculate the coefficients for a notch filter or a second order low pass filter. Refer to the Figure 5-19.

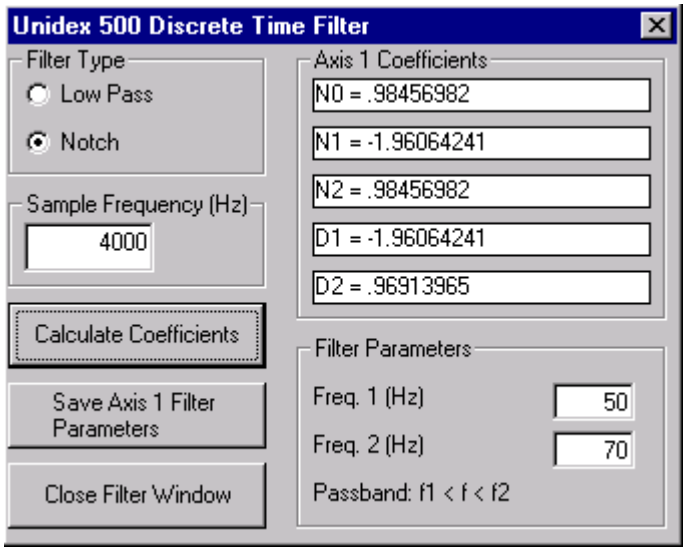


Figure 5-19. U500 Filter Utility

Once the filter parameters are specified, the coefficients can be calculated and updated within the parameter file.

5.9.7.1. The Notch Filter

The UNIDEX 500 implements a *second order discrete time filter*. The filter has a sample time specified by axis parameter no. x62. The general format of this equation is shown below.

$$H(z) = \frac{N_0 + N_1 z^{-1} + N_2 z^{-2}}{1 + D_1 z^{-1} + D_2 z^{-2}}$$

The filter coefficients N0, N1, N2, D1, and D2 (parameters x30, x31, x32, x33, and x34) are derived through calculations that are based on the *continuous time transfer function* equation shown below.

$$H(s) = \frac{s^2 + K(\frac{\omega_0}{Q})s + \omega_0^2}{s^2 + (\frac{\omega_0}{Q})s + \omega_0^2}$$

Two functions define a *notch filter*: the center frequency (ω_0), and the "quality factor" (Q). The resonant (center) frequency must be measured from the system. The quality factor is a characterization of the width of the notch. For example, a large "Q" value (e.g., 5) results in a narrow stop band, while a small "Q" value (e.g., 0.1) results in a wide stop band.

The backwards difference transformation:

$$s = \frac{z-1}{zTs}$$

is used to convert the continuous time equations into discrete time.

To determine appropriate notch filter coefficient values, it is convenient to make the following definitions:

$$A \equiv KT_s \frac{\omega_0}{Q} \qquad B \equiv T_s \frac{\omega_0}{Q} \qquad C \equiv (\omega_0 T_s)^2$$

where:

ω_0 = center frequency in radians/sec ($\omega=2\pi f$)

K = desired gain at center frequency

T_s = servo loop sample time (in sec) (normally 0.25×10^{-3} sec or 0.25 ms) (see parameter x62 for more information)

Q = quality factor (characterizes the width of the notch)

With these definitions in place, the simplified calculations of notch filter coefficients are listed below.

$$N_0 = \frac{1+A+C}{1+B+C} \qquad D_1 = \frac{-(2+B)}{1+B+C}$$

$$N_1 = \frac{-(2+A)}{1+B+C} \qquad D_2 = \frac{1}{1+B+C}$$

$$N_2 = \frac{1}{1+B+C}$$

5.9.7.2. Notch Filter Example

A system resonance has been identified at 70 Hz. Calculate the notch filter coefficient to provide 6 dB of attenuation at this frequency with a "Q" value of 5.

First, convert the center frequency in hertz (*fo*) to radians/sec (ω_0).

$$\omega_0 = 2\pi * 70 = 440 \text{ radians / sec}$$

Next, calculate the gain constant (K) at the center notch frequency (ω_0).

$$K = 10^{\frac{dB}{20}} = 10^{\frac{-6}{20}} = 0.5$$

The servo sample time (from axis parameter x62) is 0.25×10^{-3} sec (4 kHz update rate). We assume that this parameter is set to 0 or 1, which gives a 0.25 ms update time.

Although not required, the width of the notch can be calculated using the following equation:

$$BW = \frac{f_0}{Q} = \frac{70}{5} = 14 \ Hz$$

Next, the intermediate values A, B, and C must be calculated:

$$A = KT_s \frac{\omega_0}{Q} = (0.5)(0.00025)\left(\frac{440}{5}\right) = 0.011$$

$$B = T_s \frac{\omega_0}{Q} = (0.00025)\left(\frac{440}{5}\right) = 0.022$$

$$C = (\omega_0 T_s)^2 = (440 * 0.00025)^2 = 0.0121$$

Using the values for A, B, and C, and the formulas discussed earlier, the notch filter coefficients N0, N1, N2, D1, and D2 can be calculated and entered in parameters x30, x31, x32, x33, and x34, respectively.

$$N_0 = \frac{1 + A + C}{1 + B + C} = \frac{1.0231}{1.0341} = 0.989363$$

$$N_1 = \frac{-(2 + A)}{1 + B + C} = \frac{-2.011}{1.0341} = -1.944686$$

$$N_2 = \frac{1}{1 + B + C} = \frac{1}{1.0341} = 0.96704$$

$$D_1 = \frac{-(2 + B)}{1 + B + C} = \frac{-2.022}{1.0341} = -1.955323$$

$$D_2 = \frac{1}{1 + B + C} = \frac{1}{1.034} = 0.967024$$



Notch filter calculations are done in radians (not degrees).

5.9.7.3. The Second Order Low Pass Filter

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

The UNIDEX 500 may also implement a *second order low pass filter*. The coefficients for a low pass filter are defined below.

$$H(s) = \frac{\omega_0^2}{s^2 + (\frac{\omega_0}{Q})^2 s + \omega_0^2} \qquad Q = .707$$

$$E = 2\tan^{-1}(\pi f_0 Ts)$$

$$F = \frac{1 - \frac{d}{2}\sin(E)}{1 + \frac{d}{2}\sin(E)}$$

where:

f_0 = roll off frequency
d = damping factor (a value of 1.414 is recommended)
Ts = servo loop sample time (in sec) (normally 0.25×10^{-3} sec or 0.25 ms) (see parameter x62 for more information)

$$N_0 = (1 + D_1 + D_2)/4$$
$$N_1 = (1 + D_1 + D_2)/2$$
$$N_2 = (1 + D_1 + D_2)/4$$
$$D_1 = -(1 + F)\cos(E)$$
$$D_2 = F$$

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x62

5.9.8. Servo Loop Update Rate (1-100) x 0.25 ms

Parameter x62 is the *Servo loop update rate* parameter. This parameter specifies how often the servo control loop is to be updated by the UNIDEX 500. This parameter specifies a multiplier that corresponds to update rates of 0.25 ms (1 * 0.25 ms) to 25 ms (100 * 0.25 ms = 25 ms). Refer to Table 5-77.

Table 5-77. Settings for Parameter x62

Param #	Axis #	Range of Values (Update Rates Shown in Parentheses)	Defaults
162	1	1 to 100 (corresponding to 0.25 ms to 25 ms)	1 (=0.25 ms)
262	2	1 to 100 (corresponding to 0.25 ms to 25 ms)	1 (=0.25 ms)
362	3	1 to 100 (corresponding to 0.25 ms to 25 ms)	1 (=0.25 ms)
462	4	1 to 100 (corresponding to 0.25 ms to 25 ms)	1 (=0.25 ms)

Typical values are 1, 2 and 4 for 4 kHz, 2 kHz, and 1 kHz update rates, respectively.

When the system contains a combination of fast update and slow update rates, the slower axes should be assigned to a higher axis number to avoid beating the servo loop. For example, Axis 1+2 at 4kHz and Axis 3+4 at 1 kHz.

x78

5.9.9. Servo Loop Type

The UNIDEX 500 PID loop configuration can be changed. The configuration is determined by the status of the first 3 bits of parameter x78. See Table 5-78. Entering an appropriate decimal value for Parameter x78 can change the configuration.

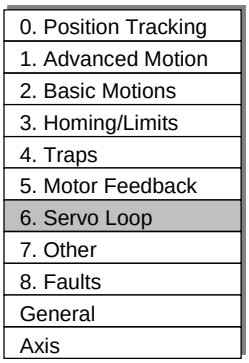
Table 5-78. Bits of Parameter x78 and PID Loop Configuration

Parameter Value	PID Loop Configuration
8	1 Cycle Velocity Averaging
16	4 Cycle Velocity Averaging



Parallel loop does not support dual loop mode.

5.9.10. Filter Time Constant

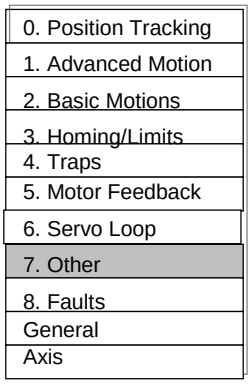


x83

Parameter x83 is used in conjunction with contour mode. A non-zero value activates an exponential filter on the specified axis. The time constant of the filter is given in milliseconds. The primary use of the filter is to smooth a trajectory that consists of non-tangential moves in G8 (velocity profiling) mode. A *Filter time constant* of 0 turns the filter completely off. A parameter setting of 1 dissipates the filter contents with no filter affect. If you do not plan to use the filter, the parameter should be set to 0. See related parameters 031, 049, 067, and 085, *Contouring mode*, for more information. Settings for parameter x83 are given in Table 5-79

Table 5-79. Settings for parameter x83

Parameter #	Range	Default Value
x83	0 – 8,388,607	0

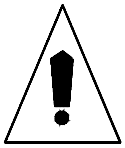


5.10. The Other Tab

The "Other" tab (tab 7) contains miscellaneous parameters used to configure the UNIDEX 500 system. There are 14 parameters in this parameter group. These parameters are listed in Table 5-80 and explained in detail in this section.

Table 5-80. Other (Miscellaneous) Parameters (Tab 7)

Param Number	Description	Default Value
x00	*Metric conversion factor*	1.0
x01	*English conversion factor*	1.0
x11	*Positive (+) move is clockwise (y/n)*	Yes
x12	*Positive (+) jog is same direction as + move (y/n)*	Yes
x13	*Pause enable in freerun (y/n)*	Yes
x14	*MFO enable in freerun (y/n)*	Yes
x15	*Calibration enable (y/n)*	Yes
x35	*In position deadband (machine steps)*	10
x37	*Backlash (machine steps)*	0
x50	*Joystick: High speed (machine steps/sec)*	40,960
x51	*Joystick: Low speed (machine steps/sec)*	2560
x52	*Absolute mode scale (machine steps)*	10
x71	*Orthogonality correction table enabled (y/n)*	Yes
x72	*2-D error mapping enabled (y/n)*	Yes



After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. This is done using the F9 RESET button located at the bottom of the main software window.

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x00

x01

5.10.1. Metric (x00) and English (x01) Conversion Factors

A *conversion factor* is a number that is used to determine system scaling (i.e., the number of machine steps in relation to program steps). These parameters give the operator the flexibility to define arbitrary program units (inches, tenths of inches, millimeters, centimeters, etc.) on a per axis basis for either Metric or English measuring systems. The following terms are used in the explanation of the system conversion (scale) factor.

Program Unit — User units such as inches, millimeters, degrees, etc. These are the units that are used within the application program.

Program Steps — The smallest programmable increment of motion.

Program steps = programming units * 10^{ndec}

where "ndec" is the number of decimal digits set by *Metric/English mode number of decimal digits* (029, 047, 065, and 083 for Metric mode, and 030, 048, 066, and 084 for English mode).

Machine Step — Smallest feedback device step. This is the smallest possible increment of movement as measured by the feedback device.

Machine Steps/Unit — The number of machine steps per programming unit.

If the number of decimal digits is specified as 3, then the programming step size is .001 and there are 1000 programming steps per programming unit.

The UNIDEX 500 uses one of two system conversion factors to convert programming units into machine steps. The default conversion factor used for each axis is specified as either English or Metric by the *Plane n Metric System (y/n)* parameter (020, 038, 056, and 074). The scaling mode set by these parameters may be overridden by use of the G70 (English) or G71 (Metric) commands.

Conversion Factor Formula

The UNIDEX 500 uses an internal formula to derive conversion factors. This formula is shown below.

$$\text{Conversion Factor} = \frac{\text{Machine Steps} / \text{Programming Unit}}{\text{Program Steps} / \text{Programming Unit}} \quad \text{or}$$

$$\text{Conversion Factor} = \frac{\left(\text{Machine Steps} / \text{Programming Unit}\right)}{10^{\text{ndec}}}$$

Conversion factors (i.e., parameters x00 and x01) are represented using a standard floating point number (e.g., 1.0) in the software package. The UNIDEX 500 contains a utility to aid in the calculation of this number.

The UNIDEX 500 prompts the operator with two questions in order to calculate the conversion factor. These questions are listed and explained below.

1. Enter number of machine steps per programming unit.

   This is the number of encoder counts in 1 inch, 1mm, or 1 degree, etc.

2. Enter number of program steps per program unit.

   This value is related to the number of decimal digits (ndec) parameter. Refer to Table 5-81.



The *Metric/English mode number of decimal digits* (029, 030, 047, 048 and 065, 066, 083, 084) parameter needs to be entered separately in the axis plane. This should be done for both English and Metric scale factors.

Table 5-81. Relationship Between Number of Decimal Digits Parameters and the Number of Programming Steps per Programming Unit

Smallest Programming Unit	Number of Decimal Digits (ndec) Parameter	Program Steps per Programming Unit
0.1	1	10
0.01	2	100
0.001	3	1,000
0.0001	4	10,000
0.00001	5	100,000
0.000001	6	1,000,000
0.0000001	7	10,000,000
0.00000001	8	100,000,000

A rotary axis has a 5000-line encoder with a “times 10” external multiplier box. Calculate the conversion factor so that the programmed unit is degrees with a resolution of 0.001 degrees.

Example 1

The total machine step count is 5000 x 10 x 4 (= 200,000 counts per revolution) every 360 degrees. The number of machine steps per programming unit is 200,000/360 = 555.55555555. The value $10^{ndec} = 10^3 = 1000$. The calculated conversion factor is 555.55555555/1000 = 0.55555555. Enter 0.55555555 in the value box.

Example 2

Consider the example of a system that has a 4 mm pitch ball screw (i.e., 4 mm/rev) and a 1,000 (x 4) line encoder. From this information, the English and Metric conversions are accomplished as follows:

In Metric mode, we know that the programming unit is 1 millimeter (1 mm). If 1 revolution of the ball screw motor produces 4,000 machine steps and 4 mm of motion, then the number of machine steps per programming unit is

$$\frac{\text{Machine Steps}}{\text{Prog Unit}} = \frac{4000 \text{ Machine Steps}}{4 \text{ mm}} = 1000 \text{ Machine Steps / mm.}$$

The number of decimal places is usually selected so that programming steps and machine steps are of similar size: ndec = 3.

$$\text{Conversion factor} = \frac{1000 \text{ Machine Steps / mm}}{10^3} = 1.0$$

machine steps/programming unit = 1000

Metric Scale Factor

programming steps/programming units = 1000

1 Programming Step = 1 mm.

English Scale Factor

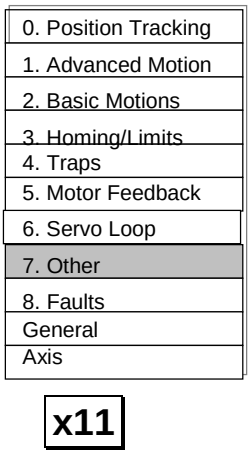
In English mode, one programming unit is 1 inch. One motor revolution is 1000 * 4 = 4000 machine steps. Since 1 inch = 25.4 mm and one motor revolution = 4 mm, there are 6.35 motor revolutions per inch or 6.35 * 4,000 = 25,400 machine steps/programming unit. Four decimal places should be selected so that the programming resolution is not sacrificed.

$$\text{Conversion factor} = \frac{25{,}400 \text{ Machine Steps / inch}}{10^4} = 2.54$$

This conversion factor is entered as 2.54.

Machine Steps / Programming Unit = 25,400

programming steps/programming unit = 1,000



5.10.2. Positive (+) Move is Clockwise (Y/N)

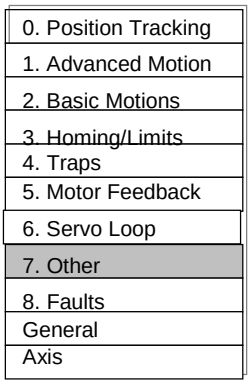
Each axis of the UNIDEX 500 may be configured so that either a positive or negative command results in clockwise motor rotation. The direction of motor rotation is specified relative to "looking into" the shaft end of the motor.

This parameter can be set to either yes or no. Descriptions are given in Table 5-82.

Table 5-82. Settings for Parameter x11

Param #	Axis #	Description
111	1	Yes - a positive command results in CW motor rotation (default) No - a positive command results in CCW motor rotation
211	2	*same as above*
311	3	*same as above*
411	4	*same as above*

5.10.3. Positive (+) Jog is Same Direction as + Move (Y/N)



Each axis of the UNIDEX 500 may be configured such that either a positive or negative jog command results in motion in the same or opposite direction as the *Positive (+) move is clockwise (y/n)* (x11) parameter.

This parameter can be set to either yes or no. Descriptions are given in Table 5-83.

Table 5-83. Settings for Parameter x12

Param #	Axis #	Description
112	1	Yes - a positive jog results in motion that is in the same direction as specified by the Positive (+) Move is CW parameter No - a positive jog results in motion that is in the opposite direction as specified by the Positive (+) Move is CW parameter
212	2	*same as above*
312	3	*same as above*
412	4	*same as above*

5.10.4. Pause Enable in Freerun (Y/N)

x13

This parameter is used to enable or disable the PAUSE function (F3 button) when the associated axis is in freerun.

This parameter can be set to either yes or no. Descriptions are given in Table 5-84.

Table 5-84. Settings for Parameter x13

Param #	Axis #	Description
113	1	Yes - F3 Pause function is enabled during freerun of the associated axis (default) No - F3 Pause function is disabled during freerun of the associated axis
213	2	*same as above*
313	3	*same as above*
413	4	*same as above*

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x14

5.10.5. MFO Enable in Freerun (Y/N)

The UNIDEX 500 may be equipped with an external Manual Feedrate Override potentiometer. This parameter is used to enable or disable this potentiometer's effect when the associated axis is in freerun.

This parameter value is set to either yes or no. Descriptions are given in Table 5-85.

Table 5-85. Settings for Parameter x14

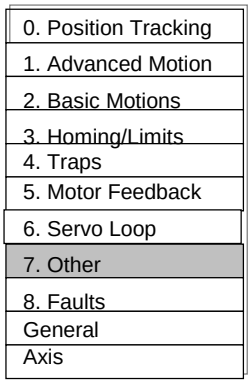
Param #	Axis #	Description
114	1	Yes - MFO potentiometer is enabled when the associated axis is in freerun (default) No - MFO potentiometer is disabled when the associated axis is in freerun
214	2	*same as above*
314	3	*same as above*
414	4	*same as above*

x15

5.10.6. Calibration Enable (Y/N)

Axis calibration is an option available to the UNIDEX 500 user. Data points representing a correction are loaded to the UNIDEX 500 during initialization from a calibration (.CAL) file. Subsequent axis positioning is then interpolated based on the .CAL file data.

See Appendix G for specific information.



5.10.7. In Position Deadband (Machine Steps)

The in-position deadband parameter (x35) specifies a window (given in machine steps) into which the axis position error must fall in order for the "in position" status bit to be set. The UNIDEX 500 continually compares the axis position error (the difference between the commanded position of an axis and its feedback position) with the in-position deadband value. If the position error is less than or equal to the value specified in parameter x35, and there is no commended motion for the axes, then the "in position" status bit of the diagnostics screen is set (a "*" appears). This bit is accessible from the library functions and is otherwise unused.

The *In position deadband* value is given in machine steps and can range from 0 to 65,536. The default value is 10 machine steps. Refer to Table 5-86.



The UNIDEX 500 continues to drive the position error to zero even after it is within the established deadband.

Table 5-86. Settings for Parameter x35

Param #	Axis #	Range	Default Values
135	1	0-65,536 machine steps	10 machine steps
235	2	0-65,536 machine steps	10 machine steps
335	3	0-65,536 machine steps	10 machine steps
435	4	0-65,536 machine steps	10 machine steps

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x37

5.10.8. Backlash (Machine Steps)

The Backlash (x37) parameter specifies the number of machine steps required to compensate for any backlash present in the mechanical system after a direction change. Positioning accuracy is increased when this value is added to the new direction.

This parameter value can range from 0 to 65,536 machine steps. The default setting for this parameter is 0 (no backlash compensation for the specified axis). See Table 5-87.

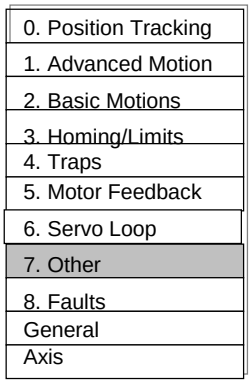
Table 5-87. Settings for Parameter x37

Param #	Axis #	Range	Default Values
137	1	0-65,536 machine steps	0 (no backlash compensation)
237	2	0-65,536 machine steps	0 (no backlash compensation)
337	3	0-65,536 machine steps	0 (no backlash compensation)
437	4	0-65,536 machine steps	0 (no backlash compensation)



IMPORTANT

Large amounts of mechanical backlash will limit the usable band width of the servo system. This function will not satisfactorily compensate for a poor mechanical system.



x50

5.10.9. Joystick: High Speed (Machine Steps/sec)

The *Joystick: High speed (machine steps/sec)* (x50) parameter defines the speed of the associated axis when a joystick (SLEW) command is issued. Since resolution ratios vary between axes, the operator must ensure that the speed/distance ratio for each affected axis is compatible prior to requesting a joystick move. To derive the joystick "high speed" setting for an axis, proceed as follows:

1. Determine the distance per second (mm/sec or in/sec) that the axis will move at this high speed.
2. Multiply this distance by the axis resolution ratio (the number of machine steps that equals 1 mm or 1 in) for the selected axis.
3. The resulting number (given in machine steps/sec) is the value that should be entered into parameter x50.

The range of and default values for this parameter are given in Table 5-88.

Table 5-88. Settings for Parameter x50

Param #	Axis #	Range	Default Values
150	1	0-8,388,607 machine steps/sec	40,960 machine steps/sec
250	2	0-8,388,607 machine steps/sec	40,960 machine steps/sec
350	3	0-8,388,607 machine steps/sec	40,960 machine steps/sec
450	4	0-8,388,607 machine steps/sec	40,960 machine steps/sec

5.10.10. Joystick: Low Speed (Machine Steps/sec)

x51

The *Joystick: Low speed (machine steps/sec)* (x51) parameter defines the speed of the associated axis when a joystick (SLEW) command is issued. Since resolution ratios vary between axes, the operator must ensure that the speed/distance ratio for each affected axis is compatible prior to requesting a joystick move. To derive the joystick "low speed" setting for an axis, proceed as follows:

1. Determine the distance per second (mm/sec or in/sec) that the axis will move at this low speed.
2. Multiply this distance by the axis resolution ratio (the number of machine steps that equals 1 mm or 1 in) for the selected axis.
3. The resulting number (given in machine steps/sec) is the value that should be entered into parameter x51.

The range of and default values for this parameter are given in Table 5-89.

Table 5-89. Settings for Parameter x51

Param #	Axis #	Range	Default Values
151	1	0-8,388,607 machine steps/sec	2560 machine steps/sec
251	2	0-8,388,607 machine steps/sec	2560 machine steps/sec
351	3	0-8,388,607 machine steps/sec	2560 machine steps/sec
451	4	0-8,388,607 machine steps/sec	2560 machine steps/sec

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

x52

5.10.11. Absolute Mode Scale (Machine Steps)

Absolute mode scale (x52) is a scaling value (from 0 to 255) that is multiplied by the analog-to-digital converter output value and creates a window of axis movement that is used for fine positioning when the system is in joystick absolute mode.

This parameter value can range from 0 to 255 which corresponds to axis movement windows from ±100 machine steps up to ± 25,500 machine steps. The default setting for this parameter is 10, which corresponds to a movement window of approximately -1,000 machine steps to 1,000 machine steps. Refer to Table 5-90.

Table 5-90. Settings for Parameter x52

Param #	Axis #	Param Range	Axis Movement Window (Range)	Default
152	1	0-255	0-25,500 machine steps	10 (1,000 machine steps)
252	2	0-255	0-25,500 machine steps	10 (1,000 machine steps)
352	3	0-255	0-25,500 machine steps	10 (1,000 machine steps)
452	4	0-255	0-25,500 machine steps	10 (1,000 machine steps)



The output from the A/D converter is from 0-255. Some of these "counts" are used internally by the UNIDEX 500. As a result, the *usable* portion of the A/D output is approximately 200 counts. This provides a ±100 count (approximate) axis movement window prior to multiplication by parameter x52.

The A/D converter is scaled so that 0 V gives an output of 0 and +5 V gives an output of 255. An A/D output of 128 corresponds to an input of +2.5 V.

x71

5.10.12. Orthogonality Correction Table Enabled (Y/N)

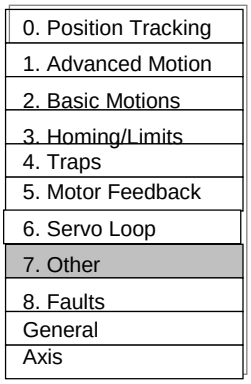
Parameter x71 is used to enable and disable the use of orthogonality data in a calibration (.CAL) file. When this parameter is set to "yes", the corresponding axis is enabled for orthogonality correction. The orthogonality entries in the specified .CAL file are used for this correction. Refer to Table 5-91. See Appendix G for specific information.

Table 5-91. Settings for Parameter x71

Param #	Axis #	Description
171	1	Yes - Axis orthogonality correction is enabled for axis 1 No - Axis orthogonality correction is not enabled for axis 1 (default)
271	2	*same as axis 1*
371	3	*same as axis 1*
471	4	*same as axis 1*

For additional information, refer to Section 5.10.6: Calibration Enable (on page 5-100).

5.10.13. 2-D Error Mapping Enabled (Y/N)



x72

Parameter x72 is used to enable and disable 2-dimensional error mapping of a pair of axes. When this parameter is set to "yes," the UNIDEX 500 uses the contents of a .MAP file (usually provided by Aerotech) to perform 2-dimensional calibration of selected axes.

The U500 can perform calibration based on a 2 Dimensional grid of correction points. Each point contains an absolute correction value in machine steps for 3 axes. The polarity of this correction is always the same as the encoder polarity as observed in the diagnostic window. See Appendix G for specific information.

5.10.14. Reverse Joystick Direction

x85

Parameter x85 is used to affect the joystick direction with respect to motor direction.. Setting this parameter to "yes" will cause a positive joystick direction to command the motor in the negative machine direction. The default setting of this parameter is "no".

0. Position Tracking
1. Advanced Motion
2. Basic Motions
3. Homing/Limits
4. Traps
5. Motor Feedback
6. Servo Loop
7. Other
8. Faults
General
Axis

5.11. The Faults Tab

The fault parameters are part of the UNIDEX 500's error checking and safety system. They are used to define the level at which error conditions are recognized and the resultant actions that will occur. There are seven parameters in the faults tab. These parameters are listed in Table 5-92 and explained in detail in this section.

Table 5-92. Fault Parameters (Tab 8)

Param Number	Description	Default Value
x55	*Error mask fault*	FFFFFFFF319F
x56	*Disable axis*	FFFFFFF0EF87
x57	*Interrupt PC*	FFFFFFF00000
x58	*AUX output*	FFFFFFF00000
x59	*Halt queue*	FFFFFFF08E00
x60	*Abort motion*	FFFFFFFF9E78
x61	*Enable brake*	FFFFFFF00000



After changing one or more parameter values, always reinitialize the UNIDEX 500 board to make the changes take effect. This is done using the F9 RESET button located at the bottom of the main software window.

5.11.1. Introduction to Fault Masks

Fault masks are used to define how an axis responds to a given error condition. Each axis can respond to an error condition by performing one or more of the following tasks:

- *Disable axis* Disables axis amplifier and servo loop
- *Interrupt PC* Generates an internal hardware interrupt
- *AUX output* Sets/clears an output bit
- *Halt queue* Immediately stops trajectory generation
- *Abort motion* Decelerates axis to a stop
- *Enable brake* Activates the U500's brake circuitry
- *Error mask fault* Enables/disables detection of error conditions for all tasks

Each task has an associated mask (number) displayed in hexadecimal format on the Edit Parameters Window. Each bit of this mask corresponds to a possible error condition. Refer to Table 5-93. If the mask bit is set to 1 and the error condition occurs, the task will execute. If the mask bit is set to 0, the task will not execute.

The "Error mask fault" determines which error conditions will be detected. Setting a bit to 0 disables detection of the error condition for all tasks.

It is not necessary to enter the parameter value in hexadecimal format. The UNIDEX 500 allows bitwise manipulation of these masks. In the Edit Parameters window, select Tab 8: Faults. In the "Axis *n*" window to the right, select the fault mask of interest. In the check boxes to the left, select or deselect the desired error condition to enable or disable it.

The UNIDEX 500 also contains a global emergency stop (E-Stop) task, which is linked to an external opto-isolated input. Setting the emergency stop bit in any fault mask on any axis to 1, enables this input. When E-Stop input occurs, all axes will disable and the "emergency stop" message will be displayed. Driving the opto isolation on and pressing the FLTACK button clears the condition.

Table 5-93. Fault Mask Bit Descriptions

Bit #	Description of Condition	Description
0	Position error	*Max position error (0-8,388,607)* (x19) exceeded
1	RMS current level exceeded	*RMS current trap (0-100%)* (x48) and *RMS current sample time (1-16,383 ms)* (x49) exceeded
2	Integral error	*Max integral error (0-8,388,607)* (x20) exceeded
3	Hardware limit (+)	CW limit input in active state
4	Hardware limit (-)	CCW limit input in active state
5	Software limit (+)	*CW software limit (machine steps)* (x23) exceeded
6	Software limit (-)	*CCW software limit (machine steps)* (x22) exceeded
7	Amplifier fault	Amplifier fault signal in active state
8	Feedback fault	Encoder line broken or resolver tracking error
9-11	*Reserved*	
12	Feedrate error	*Top feedrate (machine steps/ms)* (x17) exceeded
13	Velocity error	*Max velocity error (0-8,388,607)* (x18) exceeded
14	Emergency stop	Emergency stop input active
15	*Reserved*	
16	Axis 1 any fault	Linkage to other axis
17	Axis 2 any fault	Linkage to other axis
18	Axis 3 any fault	Linkage to other axis
19	Axis 4 any fault	Linkage to other axis
20-23	*Reserved*	

Table 5-93. Fault Mask Bit Descriptions (Continued)

Bit #	Description of Condition	Description
24-27	*Reserved*	
28-30	*Reserved*	
31	Cable interlock open	Verifies OP500 cable connection
32-35 36-39 40-43 44-47	*Reserved* *Reserved* *Reserved* *Reserved*	

x55

5.11.2. Error Mask Fault

Parameter x55 is the *Error mask fault* parameter. This parameter defines a "global" 48-bit pattern (mask) that either enables (1) or disables (0) detection of fault conditions associated with the corresponding bits for all tasks. The appropriate fault bit (see Table 5-93 on page 5-107) must be set to a "1" for the associated faults to be detected and reported. The default bit pattern for this parameter is FFFF FFFF 319F.

x56

5.11.3. Disable Axis

Parameter x56 defines a 48-bit pattern mask (corresponding to the faults listed in Table 5-93 on page 5-107) that specifies which fault conditions (if any) are used to disabled the associated axis. A “1” in a bit position indicates that the corresponding fault condition will disable the associated axis. A “0” in a bit position indicates that the corresponding fault condition is ignored. For example, if bit # 0 (position error bit) of parameter **3**56 is set to 1, then a position error fault on axis **3** will cause that axis to be disabled. The default bit pattern for the “Disable” axis fault mask is FFFF FFF0 EF87.

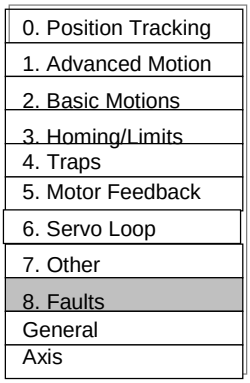
x57

5.11.4. Interrupt PC

Parameter x57 defines a 48-bit pattern mask (corresponding to the faults listed in Table 5-93 on page 5-107) that specifies which fault conditions (if any) are used to generate a hardware interrupt (when any of the selected fault conditions are true). A “1” in a bit position indicates that the corresponding fault condition will generate a hardware interrupt if the fault occurs. A “0” in a bit position indicates that the corresponding fault condition is not used to generate a hardware interrupt. For example, if bit # 14 (emergency stop bit) of parameter **2**57 is set to 1, then an emergency stop error fault on axis **2** will cause the UNIDEX 511 to generate a hardware bus interrupt. If multiple bits are set to 1 in parameter x57, then a hardware bus interrupt is generated if *any* of the faults associated with those bits occur.

The default bit pattern for the “Interrupt” fault mask is FFFF FFF0 0000 (all assigned bits are set to 0).

5.11.5. AUX OUTPUT



A fault (see Table 5-93 on page 5-107) is considered to be an AUX OUTPUT fault if the corresponding bit in this "*AUX OUTPUT*" fault mask is set to 1. If any of the selected faults occurs, then the UNIDEX 511 will set an output low. This output number is selected in parameter x54 (*Which output bit for AUX OUTPUT*). The default setting for this parameter is FFFF FFF0 0000 (no faults are selected, therefore "AUX OUTPUT" faults are effectively disabled).

5.11.6. Halt Queue

x58
x59

Parameter x59 specifies a fault mask pattern (corresponding to the faults in Table 5-93 on page 5-107) that causes the UNIDEX 511 to stop reading information from the internal queue (that is, stop program execution) if any of the selected conditions are true (i.e., if any of the selected faults occur). The default setting for this parameter is FFFF FFF0 8E00.

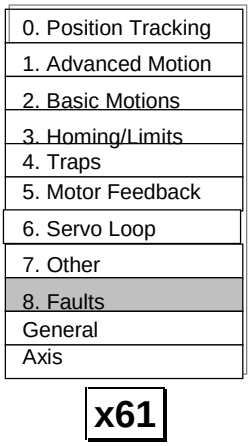
When this condition goes into effect, the commanded velocity will be immediately forced to zero. No ramping will occur and the contouring of the motion will be stopped.

5.11.7. Abort Motion

x60

Parameter x60 specifies a fault mask pattern (corresponding to the faults in Table 5-93 on page 5-107) that causes the corresponding axis of the UNIDEX 511 to ramp to a stop and wait for an acknowledgment if any of the selected conditions are true (i.e., if any of the selected faults occur). All active axes will decelerate linearly using their individual acceleration/deceleration rates. The default setting for parameter x60 is FFFF FFFF 9E78.

The ABORT cycle does not preserve the contour of the motion.



5.11.8. Enable Brake

Parameter x61 specifies a fault mask pattern (corresponding to the faults in Table 5-93 on page 5-107) that causes the brake output to be activated immediately if any of the selected conditions are true (i.e., if any of the selected faults occur). The brake will be automatically disengaged when the axis is enabled and engaged when the axis is disabled. The brake fault mask is usually configured to turn the brake on when a "disable" error occurs. For more information about the brake output, refer to Chapter 12: Technical Details. The default setting for this parameter is FFFF FFF0 0000.



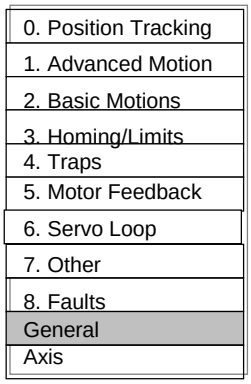
Only one axis should specify a non-zero mask for the brake. The brake will be automatically disengaged when the axis is enabled and engaged when the axis is disabled.

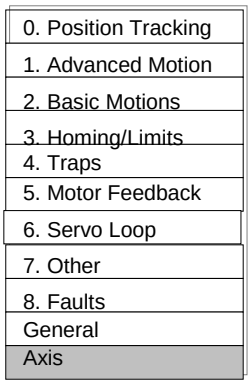


The brake fault mask is usually configured to turn the brake on when a "disable" error occurs.

5.12. The General Tab

The General tab is a numerical listing of the parameters found in the Position tracking, Advanced motion, and Basic Motions tabs. All of these parameters are duplicates of the parameters found in these other tabs and perform the same functions.





5.13. The Axis Tab

The Axis tab contains a numerical listing of the parameters that are found in the Homing/Limits, Traps, Motor Feedback, Servo Loop, Other, and Faults tabs. All of these parameters are duplicates of the parameters found in these other tabs and perform the same functions.

∇ ∇ ∇

CHAPTER 6: TUNING SERVO LOOPS

In This Section:

- Introduction.....6-1
- The Axis Scope Toolbars.....6-2
- Autotuning.....6-4
- Manual Tuning Procedure for Servo Loops.....6-9
- Tuning Tips.....6-19
- Tachometer Feedback Basics.....6-20
- Tuning Tachometer Loops.....6-22

6.1. Introduction

This chapter explains the procedures for tuning U500 servo loops with and without tachometer feedback using the software interface. The UNIDEX 500 software support package contains a graphics tool that can be used to display the effects of the servo loop gain settings. This tool is the Axis Tuning window. Refer to the Axis Tuning... option of the Tools menu in Chapter 4.

Included in this chapter are step-by-step procedures for tuning motors connected to the U500 to yield optimal performance. Most systems can be properly tuned using the autotuning procedure found in section 6.3. Stepper motors and tachometer systems can not use the autotuning procedure. These systems will need to use the manual tuning procedure found in sections 6.4, 6.6, and 6.7.

The U500 uses a dual control loop having an inner velocity loop and an outer position loop. The loop is updated according to the "*Servo loop update rate (1-100) x 0.25 ms*" (x62) parameter. Refer to Figure 6-1 for an illustration of the Servo Loop.

Before tuning can be performed, the motor and encoder must be properly connected and setup. For additional information, see the following sections:

Chapter 3 – Software Installation and Fundamentals
Chapter 5 – Parameters
Chapter 12 – Technical Details
Appendix E – Setting Up an AC Brushless Motor With the U500



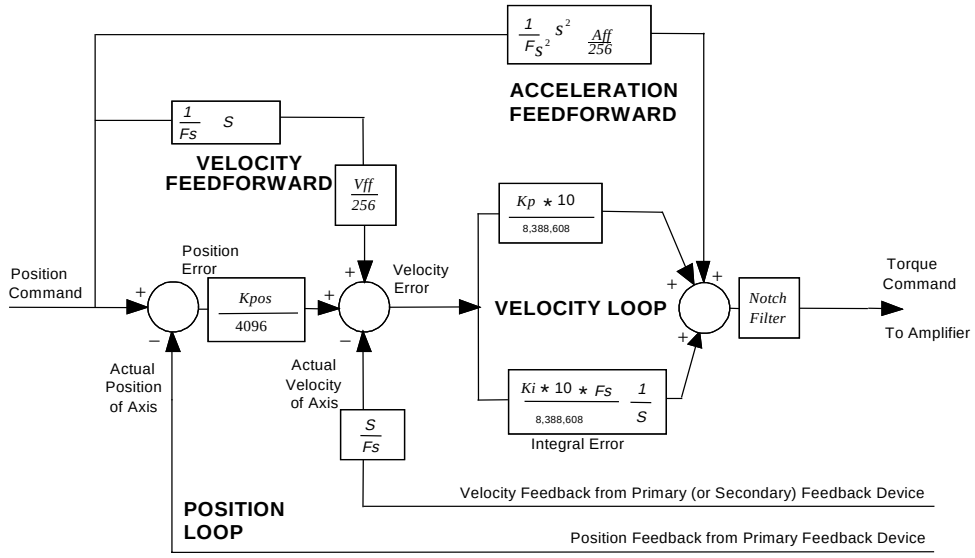


Figure 6-1. UNIDEX 500 Servo Loop

6.2. The Axis Scope Toolbars

The Axis Scope toolbars that are enabled and disabled with the Tools menu are illustrated in Figure 6-2.

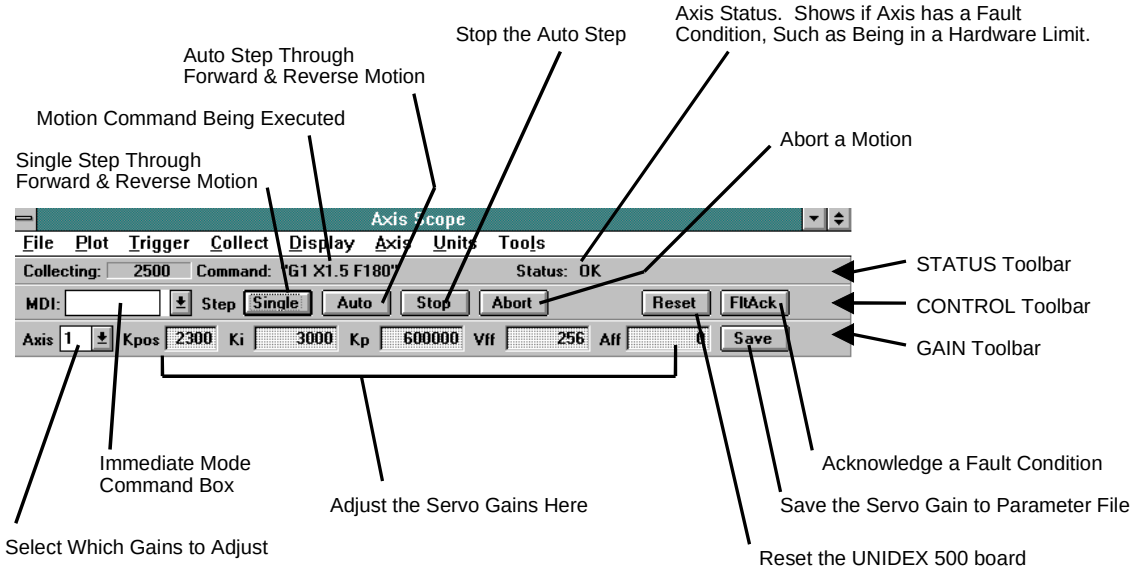


Figure 6-2. Axis Scope Toolbars

6.2.1. Kp Proportional Gain

This is the velocity loop proportional gain. This gain has a dampening effect in the servo loop. This is the first gain to adjust.

6.2.2. Ki Integral Gain

This is the velocity loop integral gain. This gain provides stiffness to the servo loop. This is the second gain to adjust.

6.2.3. Kpos Position Gain

This is the position gain. It is the only gain in the Position Loop in the UNIDEX 500's Servo Loop. This gain helps to remove the steady state position errors. This is the third gain to adjust.

6.2.4. Vff Velocity Feedforward Gain

This is the Velocity Feedforward Gain. It is the only gain in the velocity feedforward loop in the UNIDEX 500's servo loop. For motors without a secondary feedback device it is always 256. Otherwise, the user must calculate a value for Vff.

6.2.5. Aff Acceleration Feedforward Gain

This is the Acceleration Feedforward Gain. It is the only gain in the acceleration feedforward loop in the UNIDEX 500's servo loop. This gain is used to minimize position error during the acceleration and deceleration of a move. Normally this gain has a magnitude in the 100's. The user may or may not wish to adjust this gain.

6.3. Autotuning

Autotuning is a procedure that automatically identifies motor parameters and determines the motor gains. The process involves giving a sinusoidal Velocity Command to an axis. From the voltage and current used by the motor, motor parameters are identified and the servo loop gains are determined from the motor parameters. Autotuning is available only on Plus and Ultra versions of the U500.

Autotuning allows you to specify the system performance based on Bandwidth and Damping instead of arbitrary numbers Kpos, Ki, and Kp, and removes the guessing, allowing you to determine when the system is optimally tuned. It removes dependence on feedback resolution, stage type, motor type, servo loop update rate, etc.

6.3.1. Setting up an Excitation

First, in order to determine the characteristics of the stage, the U500 must excite the system. It does this with a sinusoidal motion. The user must input the amplitude and frequency of this motion. This is done in the autotune screen. The amplitude is entered in mm or in. The frequency of excitation is entered in the Freq. (Hz) box (this is similar to running a signal generator into the controller).

In order for the software to successfully identify system parameters, the torque and velocity signals must be of reasonable amplitude. This means the torque signal should be greater than 1V PK when viewed in the tuning window. If the software responds with an error message, it may be because the amplitude or frequency is set too low.

Typical amplitude is 25-50mm (1-2in) and a frequency of 1Hz. Once the software responds with gain values, you can move on to the next step.



The U500 only displays velocity feedback and torque (for one axis) when in autotune mode.

6.3.2. Specifying Desired Performance

The second step is to specify how well you want the system to perform. This is done in terms of bandwidth and damping. The damping is usually set to .7. You should start with a low value of bandwidth (10Hz) and work up until the system becomes unstable. Then return to the next lower gain values.

6.3.3. Bandwidth and Damping

Bandwidth is the responsiveness of the system expressed in terms of frequency. Higher bandwidth systems are more desirable than lower bandwidth systems. They have less position error, can track better with lower times, contour better, have smaller settling times, etc. Systems with high bandwidth have high gains.

Damping is how oscillatory the system is. This is evident in how stable the system is when it comes into position at the end of a move. Air bearing systems should use a damping of .7, mechanical bearing systems should use .3.

6.3.4. Procedure

To autotune an axis, use the following procedure:

1. From the Tools menu of the Windows software, select the Axis Tuning... option. The Axis Scope screen will appear.
2. From the Tools menu of the Axis Scope screen, select the Gains option and the Auto Tune option. The Gains and Auto Tune toolbars will be displayed on the screen. See Figure 6-3.

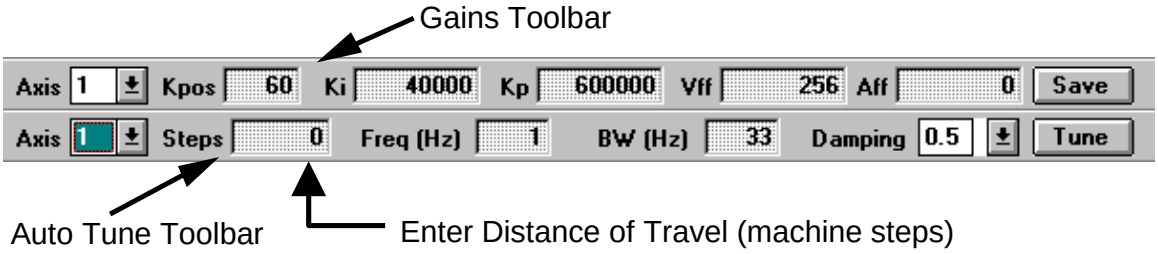


Figure 6-3. The "Gains" and "Auto Tune" Toolbars.

3. On the Gains panel, select the axis to be tuned. Make sure the gains are not set too high. Lower the Kpos value if the axis goes into oscillations or the axis traps during the autotuning procedure.
4. On the Auto tune panel, there are two text boxes that affect the motion of the axis. One of these boxes is the frequency or "Freq" box. Autotuning will excite the axis with a sinusoidal command of three different frequencies. The first frequency is displayed in the frequency box. The second frequency is twice that of the first frequency, and the third frequency is twice that of the second frequency. When first starting, set the first frequency to 1 Hz.
5. The text box with the label "mm" or "inch" is used to set up the distance the axis will travel for the first frequency. When first starting, set this value to approximately two revolutions of the motor shaft or for linear motors, 2 inches is a good starting point.
6. Set the values for BW and Damping appropriately. The BW term indicates the desired bandwidth of the velocity loop. The damping term affects how quickly the loop settles. A damping value of 0.7 configures the loop for a minimal amount of oscillation when settling into position. Lower damping values of 0.2 to 0.5 will allow the system to get to it's final position more quickly but with an increase in settling time. The higher the bandwidth value, the better the system will respond. A good starting point is a bandwidth of 30 Hz and a damping of 0.5. For systems with very little load or inertia, the BW term may be as high as 100. For heavier loads or greater inertia, the BW and damping may have to decrease. Typical damping factors for a mechanical bearing system are 0.2 to 0.5. A damping factor of 0.6 to 0.7 is recommended for air bearing systems.
7. When these values are set, hit the Tune button and the sinusoidal commands will be given to the axis. Following the completion of the motion, a pop up window showing the gains determined by the autotuning procedure should be displayed. Selecting OK will write the gains to the parameter file.
8. If an error occurs, examine the torque output. The output should be a sinusoid. If the torque appears to be clipped near 10V, lower either the number of steps or the

frequency and retry the procedure. If the torque is low with a peak of less than a volt, increase the number of steps or the frequency.

9. Following autotuning, determine if the response of the system is adequate. You may need to redo the autotuning procedure multiple times, changing the BW and damping terms, to achieve the system response desired.



The torque should be examined even if gains were determined. Autotuning should be run again to achieve a better tuned system, if the torque is not generally sinusoidal with a peak value greater than one volt.

Three autotuning plots are shown in the following figures. Figure 6-4 shows results of an autotune where "Steps" was set too low. The torque output is generally less than a volt and the commanded velocity plot is not sinusoidal. This stage should be retuned with the number of steps increased.

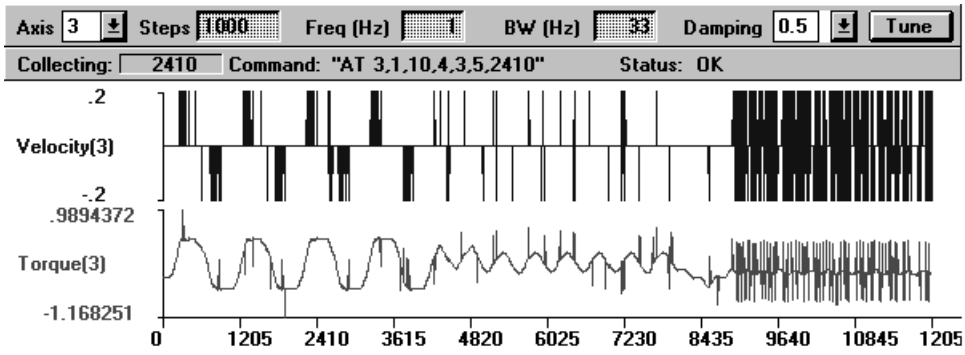


Figure 6-4. Autotune Plot where "Steps" Has Been Set Too Low

Figure 6-5 shows an autotune plot where there are too many steps. The plot shows that the torque output becomes clipped at 10V. This stage should be re-tuned with the number of steps decreased, so that the torque output is no longer clipped.

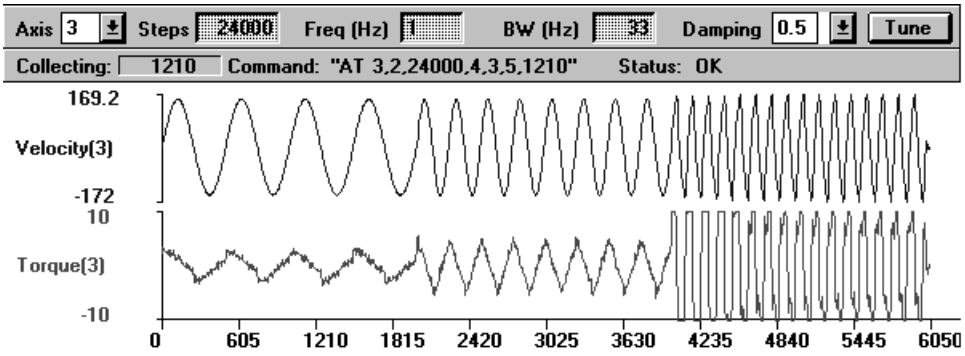


Figure 6-5. Autotune Plot where "Steps" Has Been Set Too High

Figure 6-6 shows a proper autotuning procedure. Both the torque and Velocity Command are sinusoidal and the torque output peak is greater than one volt. A resulting plot similar to this should identify the parameters of the motor/stage and produce gains that will allow the stage to be run adequately.

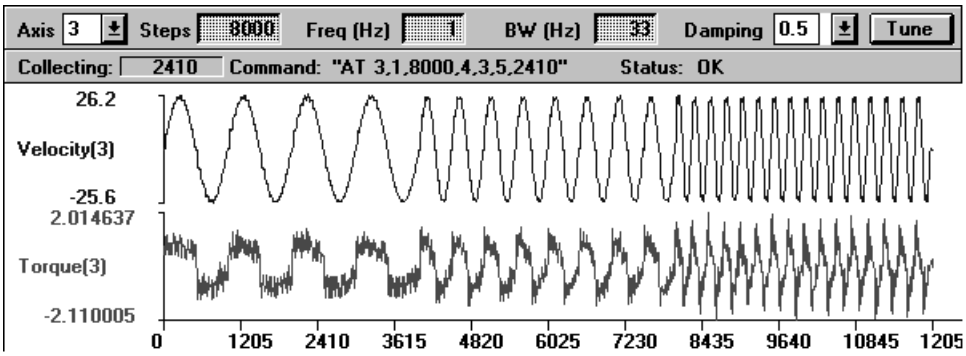


Figure 6-6. Autotune Plot Showing Proper Calibration

6.3.5. Dual Loop Systems

Autotuning may be performed on dual loop systems by temporarily configuring the U500 to run in single loop mode, i.e. from the velocity feedback transducer only. This is accomplished by setting the primary feedback channel parameter (axis parameter x38) to the velocity feedback device's channel and setting the secondary feedback channel (axis parameter x39) to 0. Autotuning will yield the correct values for Ki and Kp. The Kpos term will need to be manually adjusted when the U500 configuration is returned to dual loop mode.

6.3.6. Guidelines and Limitations

- It is usually necessary to defeat the velocity error and position error traps before tuning.
- If you do not have ballpark gains for a system, start out with low values, Kpos=10, Ki=1000, Kp=10,000.
- Most systems should be able to achieve a bandwidth of at least 30Hz.
- Care must be taken not to exceed the maximum tracking rate of the feedback device, especially resolvers/inductosyns. Large gimbals with inductosyns cannot use autotuning.
- Autotuning *will* work with vertical axis configurations.
- For systems with large mass or high inertia, it may be necessary to reduce the excitation frequency to .25 - .5Hz. For small systems, it may be necessary to increase the frequency to 2Hz.
- Autotuning can be run on unconnected motors and linear motors.
- When changing the servo loop update rate from 4KHz to 1KHz, Kp should be reduced by a factor of 4, Ki should stay the same, and Kpos should increase by a factor of 4.
- When changing resolution from 1000 lines to 2000 lines, Ki and Kp should be reduced by a factor of 2, Kpos stays the same.

6.4. Manual Tuning Procedure for Servo Loops

The following procedure can be used as a guide when manually tuning the UNIDEX 500 servo loop. Autotuning as discussed in section 6.3 is the recommended method of tuning. However, in some instances, when autotuning does not generate adequate gains, it may be necessary to adjust the servo loop gains manually. This procedure does not apply to motors with tachometers. Figure 6-7 shows the overall tuning process with the Axis Scope window. The tuning process discussed in this section was performed using the "X" Axis (Axis 1) of a X-Y stage. The user's system may behave differently and have different values for servo gains. However, the overall process is the same and the same process can be repeated for the other axes in the system. When adjusting each of the servo gains, the user will essentially be following the procedure below:

1. Press the Single button on the Control toolbar to step through a forward or reverse motion.
2. Observe the signal plots on the Axis Scope window.
3. Make a decision on whether to increase or decrease the value of the servo gain and if the observed signal is acceptable to move on to the next servo gain.
4. Repeat.

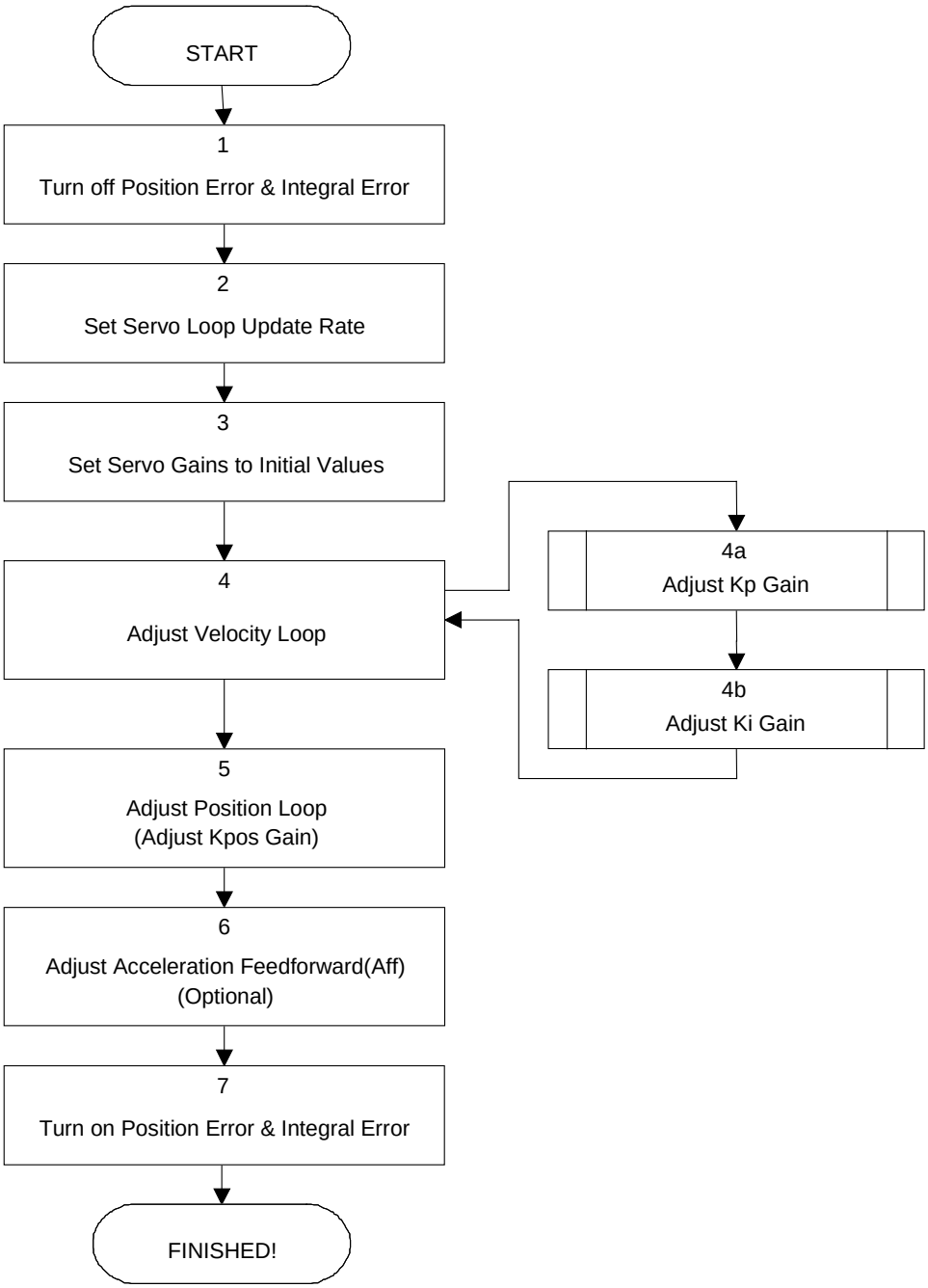


Figure 6-7. Flowchart of Overall Tuning Process

The following is a step-by-step procedure for tuning motors without tachometers.

IMPORTANT

Please read each step thoroughly before performing the task.

1. Turn off the “Position Error” and “Integral Error” Traps. In the UNIDEX 500 software, open up the Parameters Window via the EDIT menu and de-select the Position Error and the Integral Error in the parameter called “ Faults.” Refer to Figure 6-8.

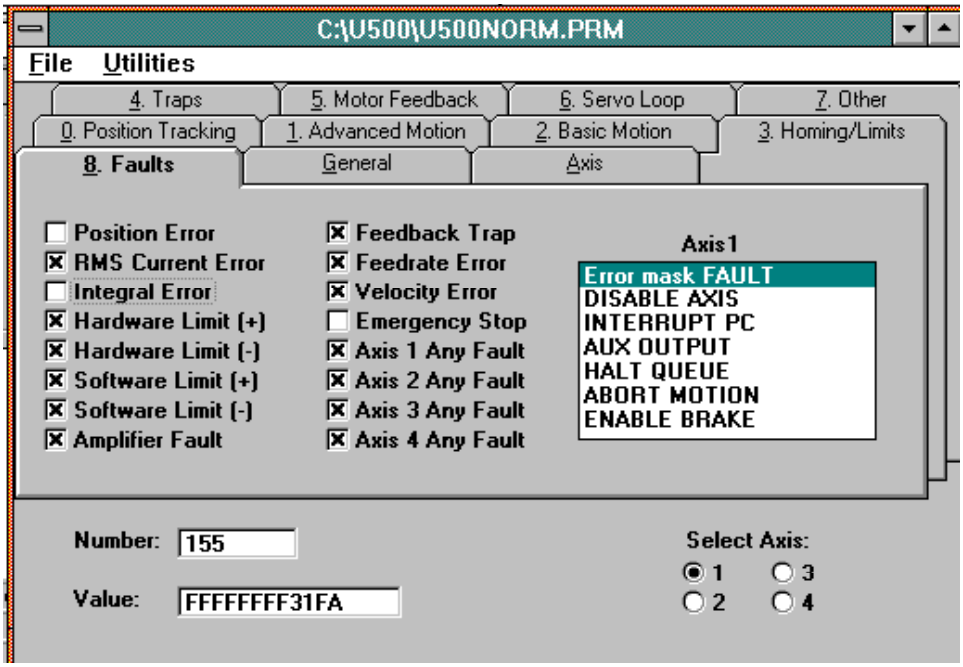


Figure 6-8. The Edit Parameters Window

2. The “*Servo loop update rate (1-100) x 0.25 ms*” parameter from the Servo Loop tab, shown in Figure 6-9, must be set appropriately in order to get optimal performance from the system. The default update rate is 0.25 ms (4 kHz) and the user would put a 1 as that parameter’s value. Another common choice is 1 ms (1 kHz) and the user would put a 4 as that parameter’s value. Some low resolution systems (500 line encoders, etc.), high inertia systems, or low velocity systems perform better at a lower update rate such as 1 kHz. If the user doesn’t know what to use for this parameter then an update rate of 4 kHz should be used. However, an update rate of 1 kHz can be used. If the update rate is changed, the tuning process must be repeated.

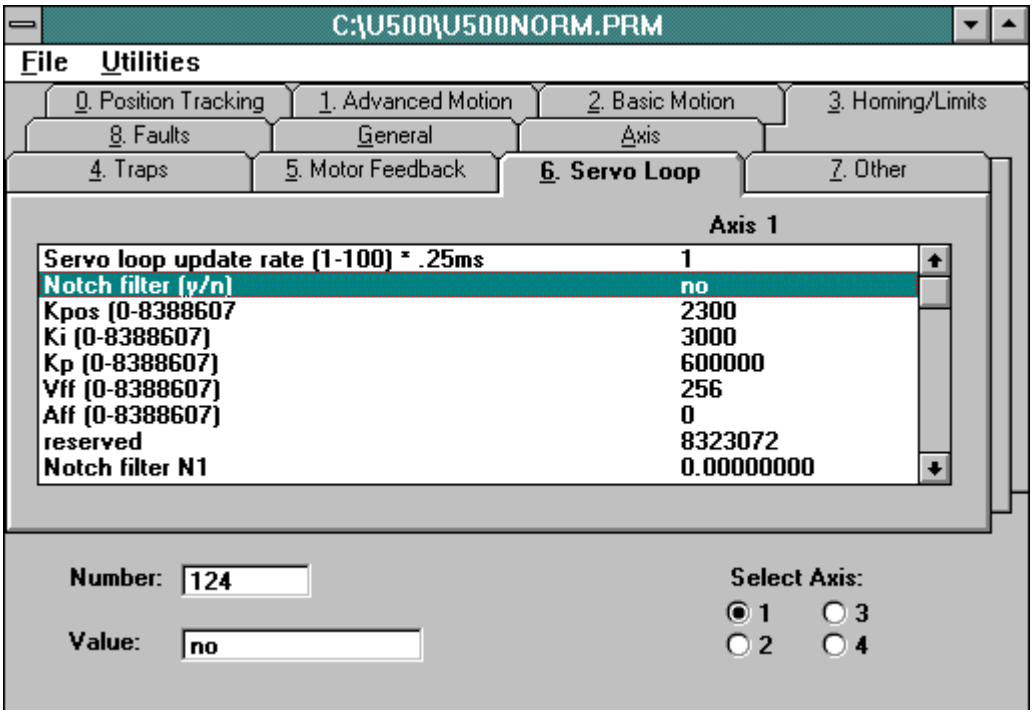


Figure 6-9. Servo Loop Tab of the Edit Parameters Window

3. Set servo parameters to initial value. While still in the Parameters window (refer to Figure 6-9), enter in the initial values for the servo gains. Table 6-1 has the initial values for these servo gains.

Table 6-1. Initial Servo Parameter Values

Kpos	Ki	Kp	Vff	Aff
0	0	≤10000	Derived value	0

Save the values set in the Parameters window and then exit this window. Reinitialize the UNIDEX 500 by pressing the Reset button.

4. Prepare the Axis Scope Screen for tuning by performing the following functions:

   a. Press the maximize button on the Axis Scope Window, shown in Figure 6-10 so the Axis Scope window fills the entire screen.



Figure 6-10. Maximize Button on the Axis Scope Window

   b. In the Collect menu select “2500 points.”

   c. In the Display menu select “2500 points.”

   d. In the Axis menu select Axis #1 (X Axis) or the axis that will be tuned.

e. In the Plot menu, select Velocity Command, Velocity Error, and Position Error.

f. In the Trigger menu, set the Forward Motion... and Reverse Motion... to a typical move such as *LINEAR X1 F180* for Forward Motion... and *LINEAR X-1 F180* for Reverse Motion... Also set the Sample Rate to 1.

g. In the Tools menu select Status, Control, and Gains.

When the "Single" button is pressed, Axis #1 will first move as specified by Forward Motion... When the Single button is pressed again, Axis #1 will move as specified by Reverse Motion...

5. Adjust the Velocity Loop using "Kp". The Kpos and Ki have been set to zero (0) to eliminate the Position Loop and half of the Velocity Loop. Thus, the only gain that is having an effect is Kp, which is the other half of the Velocity Loop.

Even though the user may only be concerned with how well the stage positions, the Velocity Loop cannot be overlooked as it is interrelated to positioning. The better a stage tracks velocity, the closer it will be to its correct position.



The objective in adjusting Kp, is to reduce the velocity error to 5-10 units. The velocity must not be eliminated completely because Ki will do that. Also, the user wants the axis to feel "snug". As Kp is increased, observe that the motor shaft (or ball screw) becomes increasingly snug. The user can turn the shaft manually and it won't move back to its original position, but it becomes harder to turn.

In between pressing the Single button, the operator should observe no screeching and howling from the motor. Noise means Kp is set too high causing it to oscillate. It may screech a little during the move, but not when at a standstill.

If the motor doesn't move, Kp is too low. Increase the value of Kp and try again by pressing the Single button.



If you are adjusting the gains that Aerotech has setup for your system, use the existing Kp as your starting point.



Once the motor is moving back and forth, the user should see a graph similar to Figure 6-11. From this graph, it is seen that there are 20 to 30 units of velocity error. "Kp" must be increased to reduce the amount of velocity error. After repeating this process a few times, the velocity error will look similar to Figure 6-12. From this graph the user can observe that the average velocity error during the move is about 6 units. Moreover the motor does not oscillate when it is stopped.

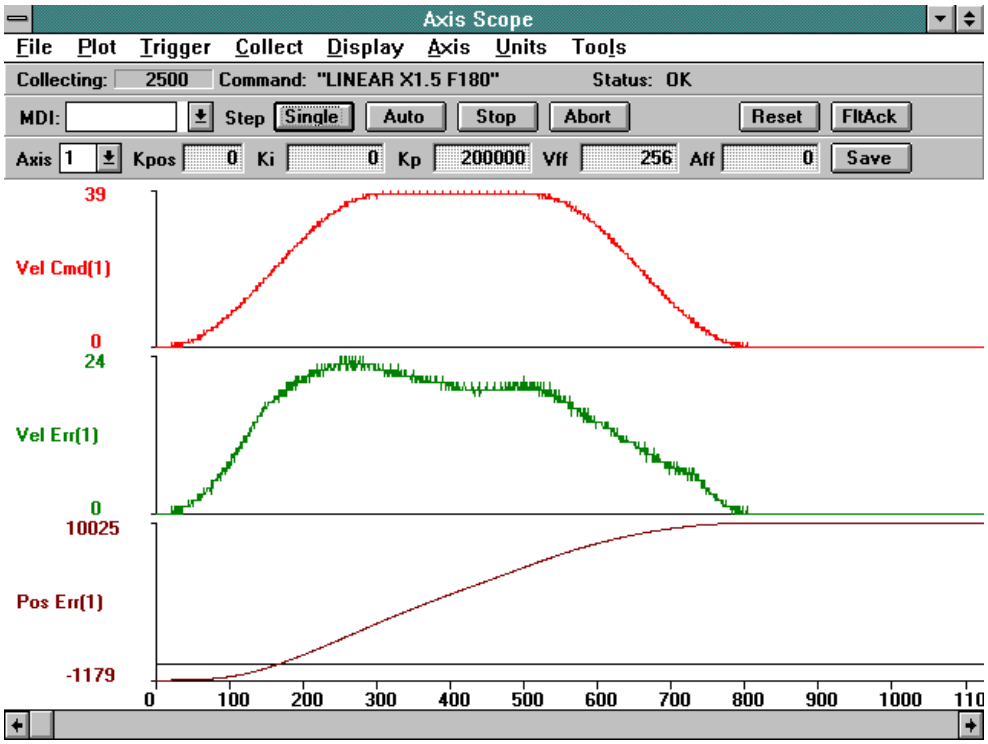


Figure 6-11. Unacceptable Velocity Error

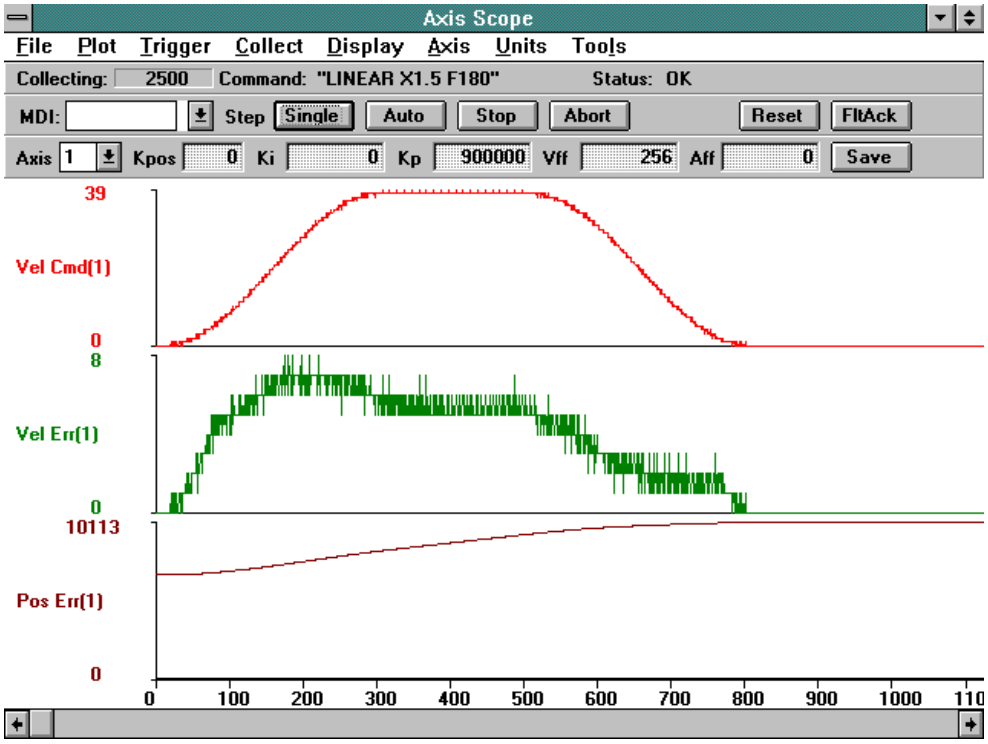


Figure 6-12. Acceptable Velocity Error (When Adjusting Kp)

The user can stop adjusting “Kp” and start adjusting “Ki”.

Use a starting value of 100 for Ki. The main objective in adjusting Ki is to reduce velocity error and position error. As Ki is increased, the error is reduced. However, the position error should not cross its “zero line”, otherwise it will increase the settling time of the axis. This is a point where the user can stop adjusting Ki. Also, a very large Ki will introduce low frequency oscillation in the position error. From the perspective of the stage, this is vibration. Lack of smoothness may or may not be acceptable to the user. Shown in Figure 6-13, is a plot showing Ki crossing its “zero line” and some oscillation.

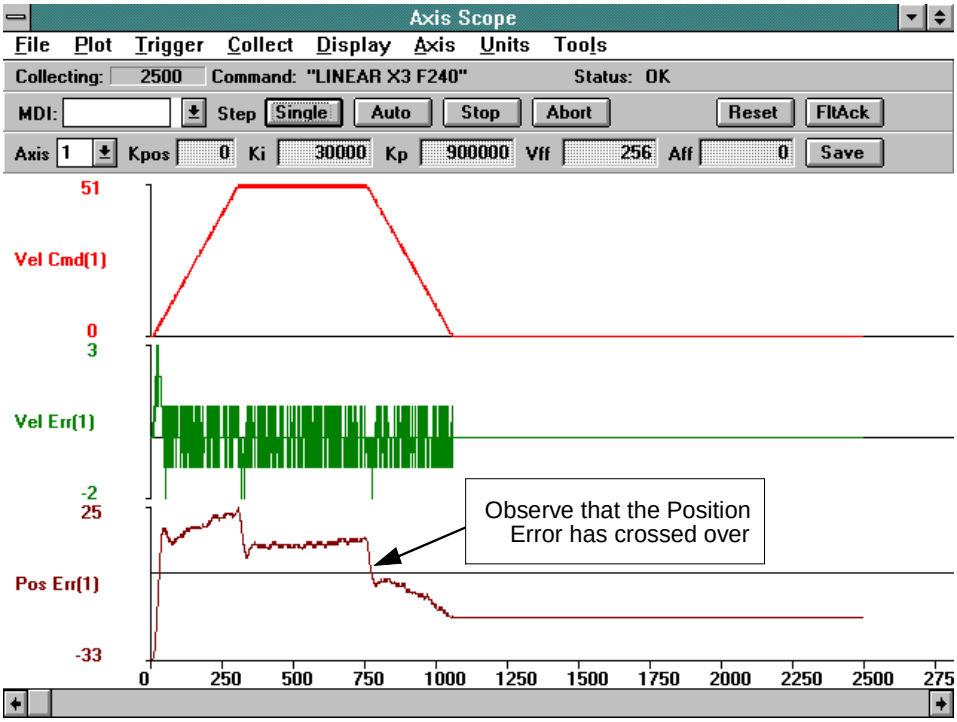


Figure 6-13. Unacceptable Position Error (When Adjusting Ki)

Depending on the application, the user may or may not be too concerned with vibration in the axis. However, the cross-over of the position error does mean the Velocity Loop is too tight, especially if maintaining position is important to the user. Due to this cross-over, the settling time increases and the Kpos (which is in the Position Loop) must work harder to keep the motor in position. The two gains start working against each other instead of complimenting each other.

Shown in Figure 6-14 is a graph with a much better Ki. As can be observed from the position error, it is smoother and the position error does not cross over the horizontal axis. Moreover, the velocity error has been reduced as well.

The user can stop adjusting “Ki” and start adjusting “Kpos.”

6. Adjust the Position Loop using “Kpos.” Use a starting value of 1-10 for Kpos. As the user increases Kpos, it will be observed that the position error is reduced. The main objective is to adjust Kpos until the position error is within user’s tolerance or starts to oscillate; whatever comes first.

   As previously mentioned, if Kpos is too high, the user will encounter a high frequency oscillation (stage vibrates strongly). This will cause the UNIDEX 500 to have a *RMS Current Trap*, which essentially means that too much current is being sent to the motor (the RMS Current Trap acts the same way as a fuse).

   Shown in Figure 6-14 is a plot of a good Kpos. From this graph it can be seen that there is little settling time. In other words, the position error ends near the same time the Velocity Command ends; so the move is “in position” at the end of the commanded move. For comparison, Figure 6-15 illustrates a plot where Kpos is too high.

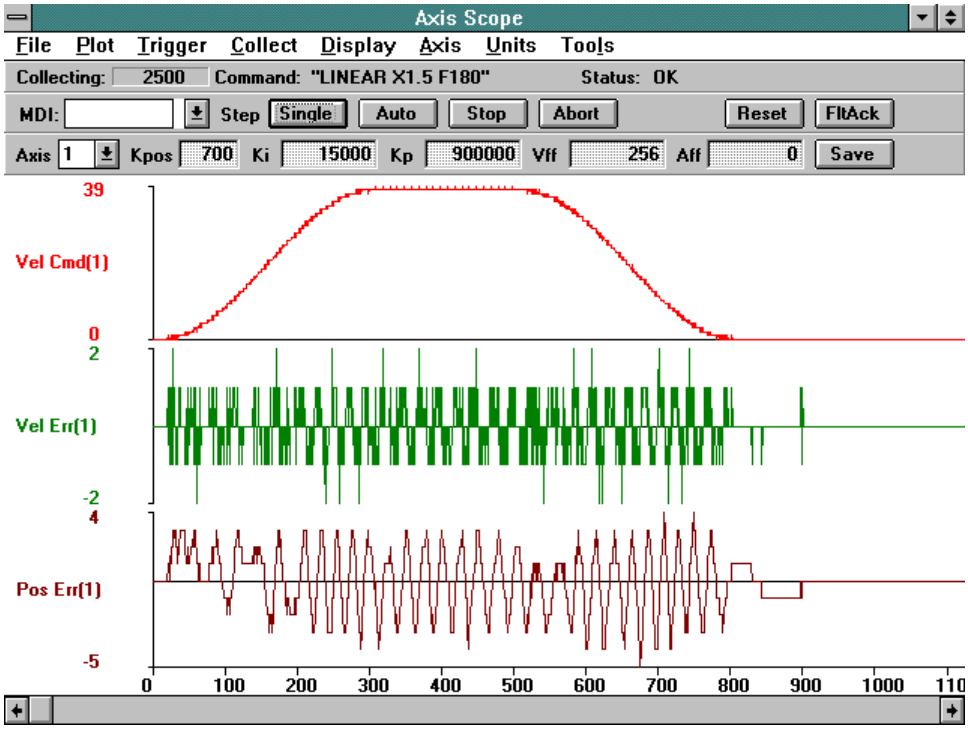


Figure 6-14. Plot Showing an Appropriate Value for Kpos

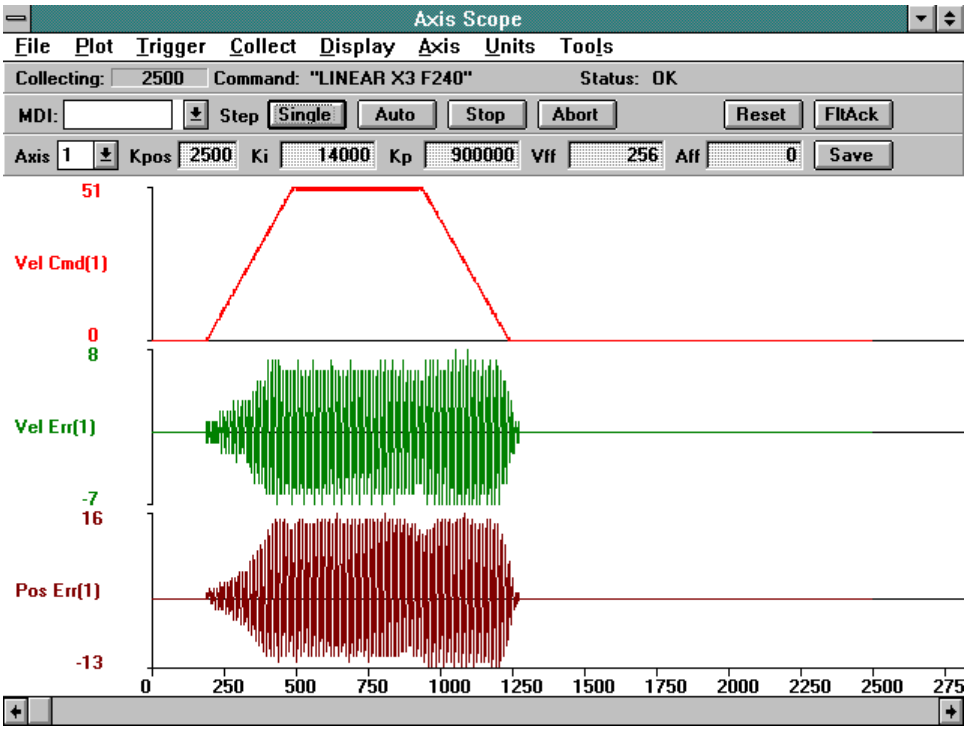


Figure 6-15. Plot Showing Overall Effects When Kpos is High

7. Adjust Acceleration Feedforward Loop using "Aff". The Acceleration Feedforward gain (Aff) attempts to remove position error during the acceleration and deceleration of a move.

   On the graph shown in Figure 6-16 the user can see for this XY stage that the Aff had little impact at all. Normal values for Aff tend to range in magnitudes of 100 to 500. The user can tell Aff is working when the system has a new grinding type noise. This noise is due to the fact that the UNIDEX 500 is working harder trying to keep "in position" at all moments of time.

Adjusting Aff is optional. The user's application may not need it.

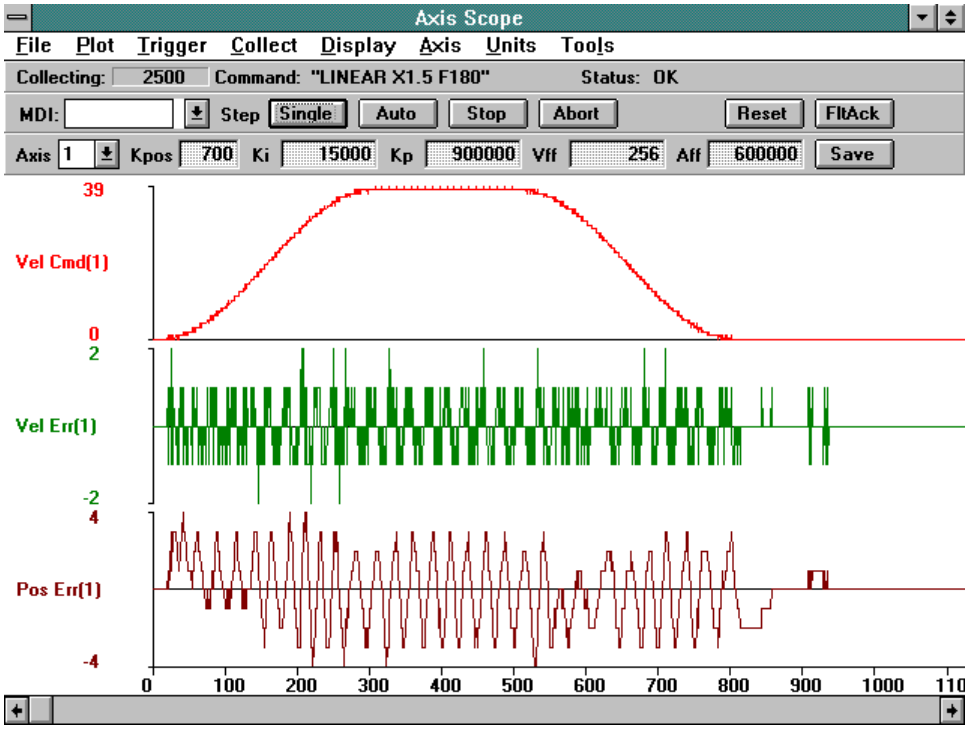


Figure 6-16. Plot Showing Aff Adjustment

8. Turn the Position Error and Integral Error Traps on by returning to the Parameters window and selecting the Parameter tab called “Faults.” Turn the “Position Error” and the “Integral Error” back on by checking the boxes. This will reactivate these traps.

   Save and exit the Parameters window.

   Reinitialize the UNIDEX 500.

6.5. Tuning Tips

Shown in Figure 6-17 is a tuning plot of an AC brushless motor. Notice the ripple effect during the move. This is normal since AC brushless motors usually have a larger torque ripple than DC brush motors. This ripple effect has been observed to be 10 machine steps peak with an unloaded AC brushless motor.

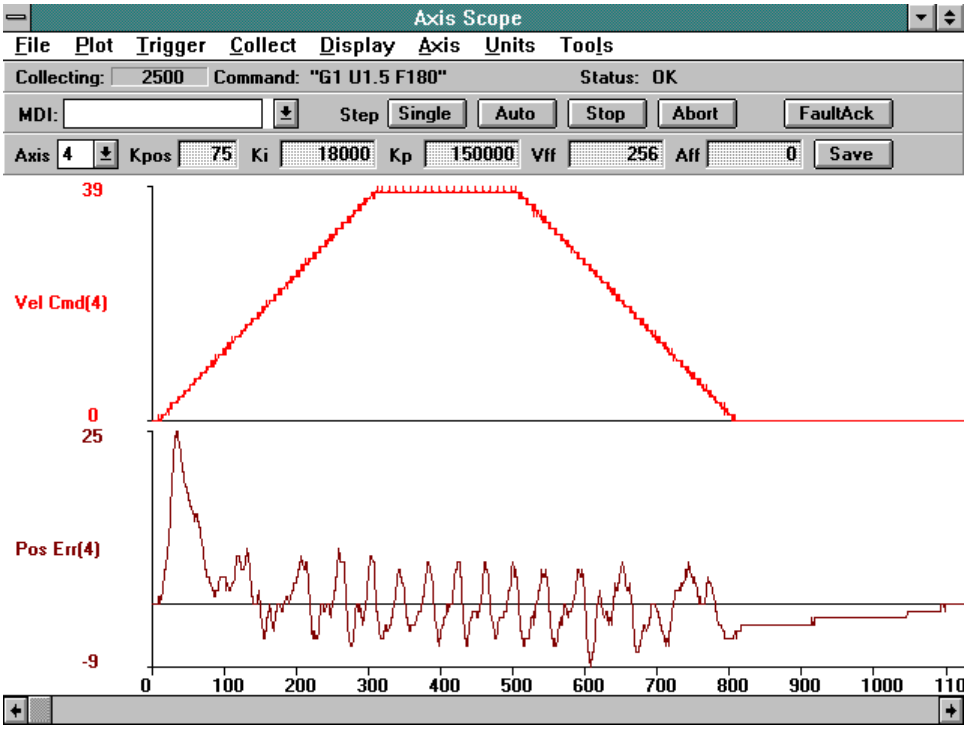


Figure 6-17. Tuning Plot of an AC Brushless Motor

6.6. Tachometer Feedback Basics

The UNIDEX 500 servo is easily configured for compatibility with external tachometers providing negative velocity feedback. To adapt to an external tachometer based Velocity Loop, the inherent digital Velocity Loop operation within the controller needs to be disabled. This is done by setting the digital servo loop proportional gain (Kp) to zero. The servo system's Velocity Loop then needs compensated by the tachometer/amplifier combination.

When configured this way, the analog outputs of the UNIDEX 500 that normally deliver current commands to amplifiers will now deliver Velocity Commands to amplifiers accepting tachometer feedback.

In this configuration, the servo system has the following characteristics:

- The amplifier is configured to accept tachometer based (negative) velocity feedback
- The amplifier controls the Velocity Loop of the servo system. Tuning for the Velocity Loop is accomplished in the Pre-amp section of the amplifier
- The proportional gain parameter (Kp) in the UNIDEX 500 controller's servo loop parameter set has been set to zero (0) for that axis, disabling its digital Velocity Loop functionality
- The controller is now commanding velocity to the amplifier instead of commanding torque to the amplifier.

6.6.1. In-Position Integrator

An in-position integrator can be enabled by setting Ki≠0. After Ki is set to a non-zero value, the UNIDEX 500 will attempt to remove steady-state position errors. This function also helps to reduce the effects of tachometer loop drift. In-position integration is accomplished at a rate that is directly proportional to the integral gain value that is set in Ki.



If a value for Ki is too large, it will induce oscillation into the position error and increase the settling time.

6.6.2. Velocity Feed Forward

The Following Error (position error) that occurs while the axis is moving may be reduced significantly by setting the velocity feed forward gain to a non-zero value. When the velocity feed forward function is enabled (i.e., Vff≠0), an added voltage is sent to the tachometer loop. This signal is proportional to the Velocity Command and the value of Vff. Vff is adjusted to minimize position error of the servo system.

6.6.3. Servo Parameter Setup for Tachometer Feedback

When configuring a servo loop containing external negative velocity feedback from a tachometer, the servo gain values shown in Table 6-2 are adjusted.

Table 6-2. Servo Gain Values

Parameter	Name	Value	Comments
Position Gain	*Kpos*	Adjust per application	Should be maximized for servo stability and acceptable position error (following error) levels.
Integral Gain	*Ki*	Optional	In-position integrator that helps reduce steady state position errors and the effects of tachometer loop drift.
Proportional Gain	*Kp*	Always zero	The digital Velocity Loop must be disabled so it does not conflict with an external tachometer providing velocity feedback.
Velocity Feed Forward	*Vff*	Optional	Minimizes following error (position error) of the servo system.
Acceleration Feed Forward	*Aff*	Always Zero	Use of this parameter also conflicts within external tachometer providing velocity feedback.

6.6.4. The Axis Scope Toolbars

The user will employ the same Axis Scope Toolbars shown in Figure 6-2 on page 6-2 when tuning with tachometer feedback. However, the gains on the Gain toolbar have slightly different meanings with tachometer feedback. The gain definitions are as shown below.

6.6.4.1. Kpos Position Gain

The Position Gain is the only gain in the Position Loop in the UNIDEX 500's Servo Loop. This gain reduces the amount of position error and decreases the settling time. It is the first gain to adjust.

6.6.4.2. Ki In-Position Integrator

This is the In-Position Integrator. Setting Ki to a non-zero value causes the UNIDEX 500 to attempt to remove steady-state position errors. This function also helps reduce the effects of tachometer loop drift. Ki is the second gain to adjust.

If the value for Ki is too large it will induce oscillation into the position error and increase the settling time.



6.6.4.3. Vff Velocity Feedforward Gain

The Velocity Feedforward Gain is the only gain in the Velocity Feedforward Loop in the UNIDEX 500's Servo Loop. This gain reduces the amount of position error for systems with a tachometer. It is the third gain to adjust.

6.6.4.4. Kp Proportional Gain

Kp is the proportional gain used in systems with tachometers. It is always set to zero (0).

6.6.4.5. Aff Acceleration Feedforward Gain

The Acceleration Feedforward Gain is the only gain in the Acceleration Feedforward Loop in the UNIDEX 500's Servo Loop. For systems with tachometers, it is always zero (0).

6.7. Tuning Tachometer Loops

The following procedure is a guide for tuning motors with tachometers. Figure 6-18 shows the overall tuning process with the Axis Scope window. The tuning process discussed here was performed using the "X" axis (Axis 1) of a X-Y stage. The user's system may behave differently and have different values for servo gains. However, the overall process is the same and the process can be repeated for the other axes. When adjusting the servo gains, the user will essentially be following the procedure below:

1. Press the Single button on the Control toolbar to step through a forward or reverse motion.
2. Observe the signal plots on the Axis Scope window.
3. Make a decision on whether to increase or decrease the value of the servo gain and if the observed signal is acceptable, move on to the next servo gain.
4. Repeat.

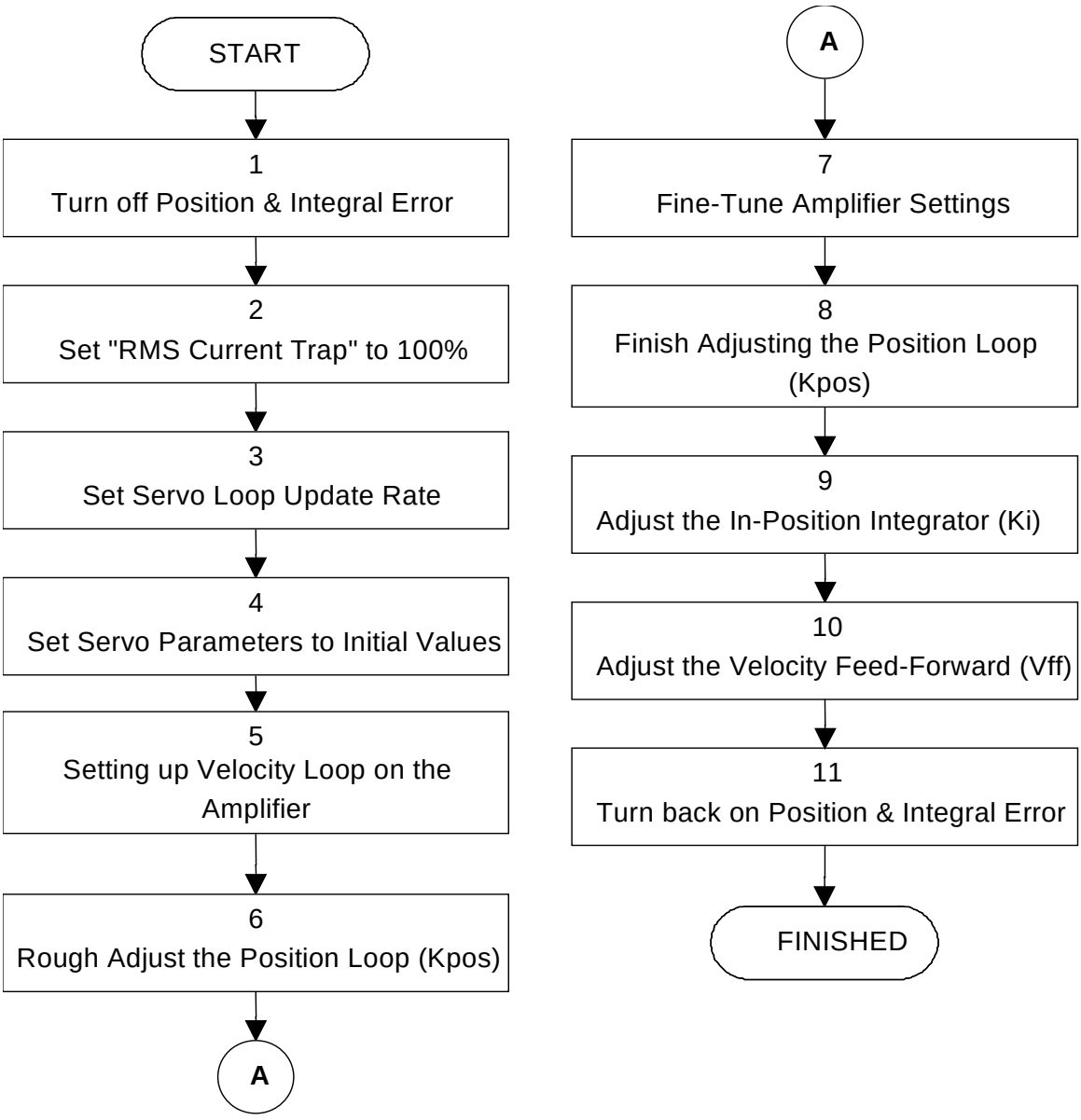
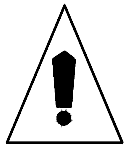


Figure 6-18. Flowchart of Overall Tach Tuning Process

The following is a step-by-step procedure for tuning motors with tachometers.



IMPORTANT

Please read each step thoroughly before performing the task.

1. Turn off the "Position Error," "Integral Error," and "Velocity Error" Traps. In the UNIDEX 500 software, open the Parameters Window via the Edit menu and de-select the Position Error and the Integral Error under the parameter tab called "Faults." Refer to Figure 6-19.

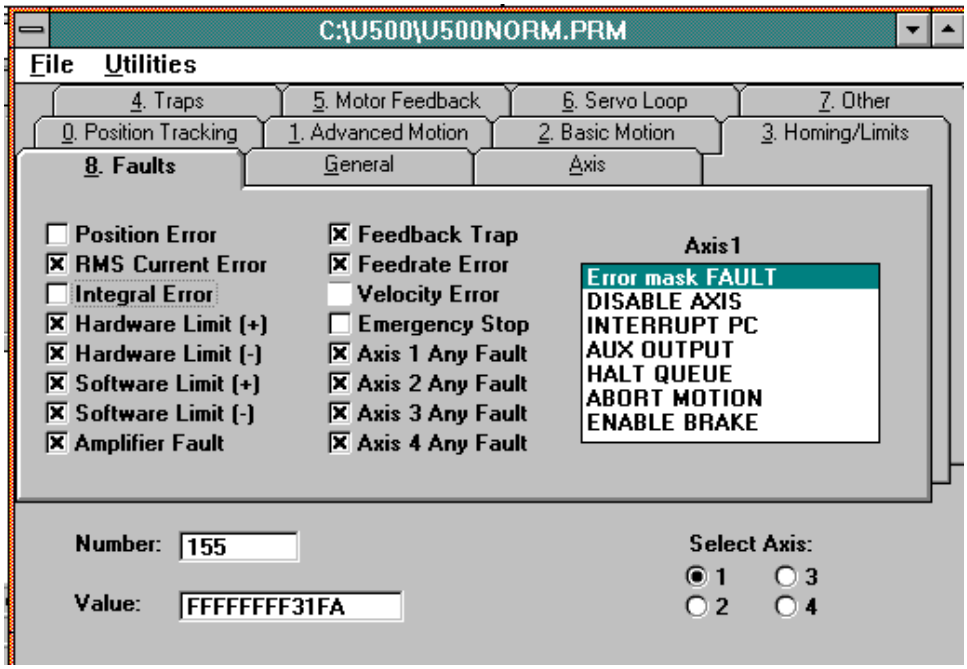


Figure 6-19. The Edit Parameters Window

2. Set the *RMS Current Trap* parameter to 100%. While still in the Parameters window, select the parameter tab called "Traps." Set the parameter, *RMS current trap (0-100%)*, to 100%.

3. The "*Servo loop update rate (1-100) x 0.25 ms*" Parameter from the Servo Loop tab, shown in Figure 6-20 must be set appropriately in order to get optimal performance from the system. The default update rate is 0.25 ms (4 kHz) and the user would put a 1 as that parameter's value. Another common choice is 1 ms (1 kHz) and the user would put a 4 as that parameter's value. Some low resolution systems (500 line encoders, etc.) or high inertia systems or low velocity systems perform better at a lower update rate such as 1 kHz. If the user doesn't know what to use for this parameter then an update rate of 4 kHz should be used. However, an update rate of 1 kHz can be used. If the update rate is changed, the tuning process must be repeated.

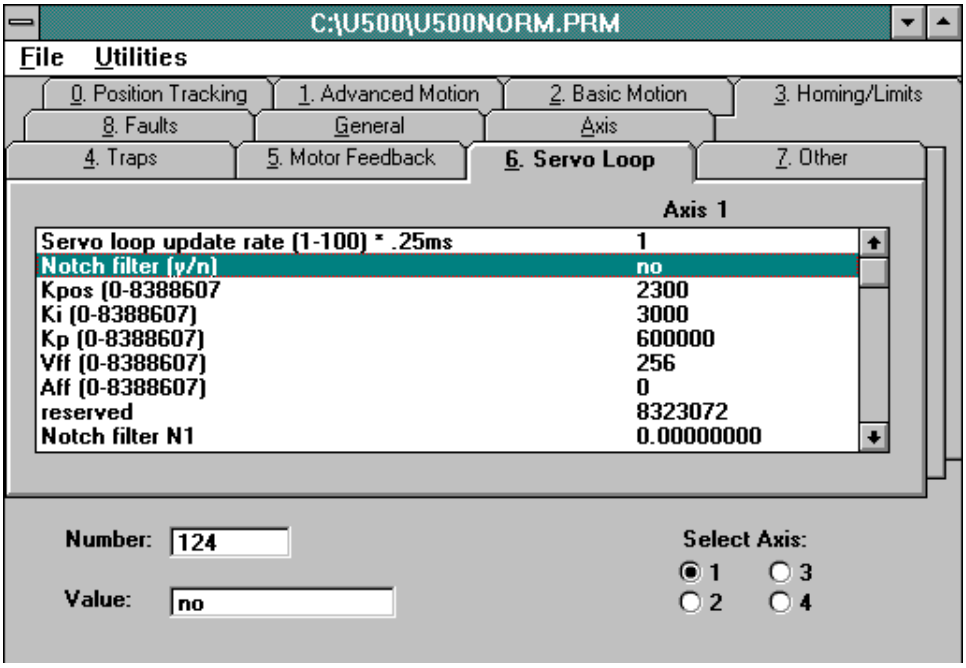


Figure 6-20. Servo Loop Tab of the Edit Parameters Window

4. Set servo parameters to initial value. While still in the Parameters window (refer to Figure 6-20) enter in the initial values for the servo gains. Table 6-3 has the initial values for these servo gains.

Table 6-3. Initial Servo Parameter Values - Tach Tuning

Kpos	Ki	Kp	Vff	Aff
0	0	0	0	Always 0

Save the values set in the Parameters window and then exit this window. Reinitialize the UNIDEX 500 by pressing the Reset button.

5. Set up the Velocity Loop on the amplifier.

If the user has a non-Aerotech amplifier, the manufacturer should provide information for setting the amplifier to Velocity Command and explain how to optimize the Velocity Loop.



If the user has a DS16020/DS16030 Aerotech amplifier, the Velocity Loop is adjusted the following way:

a. Select a fuse to protect the motor for the continuous current rating of the motor and insert it in the appropriate fuse holder of the amplifier. Refer to Figure 6-21 for location of the fuse holder.

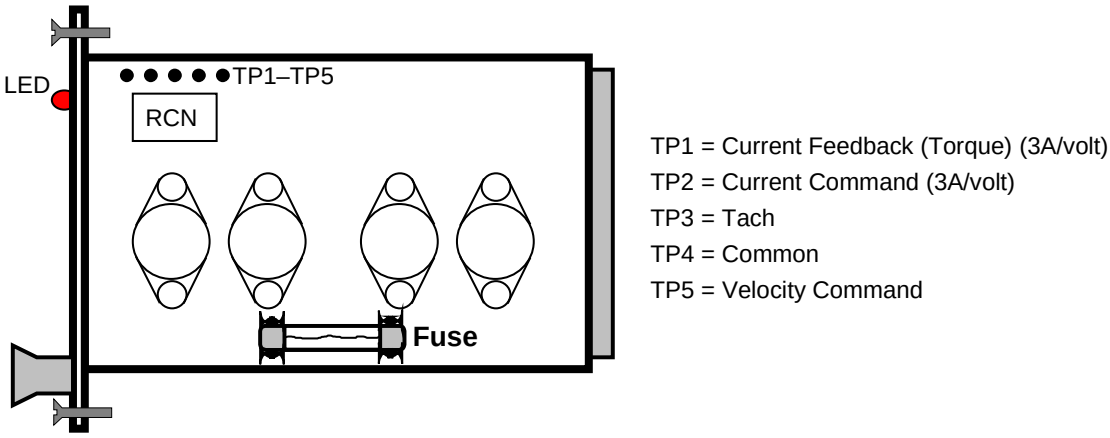


Figure 6-21. Cross-Section of the DS16020/16030 Amplifier

b. Make ballpark adjustments to the potentiometers on the Aerotech DS16020/16030 amplifier as shown in Figure 6-22.

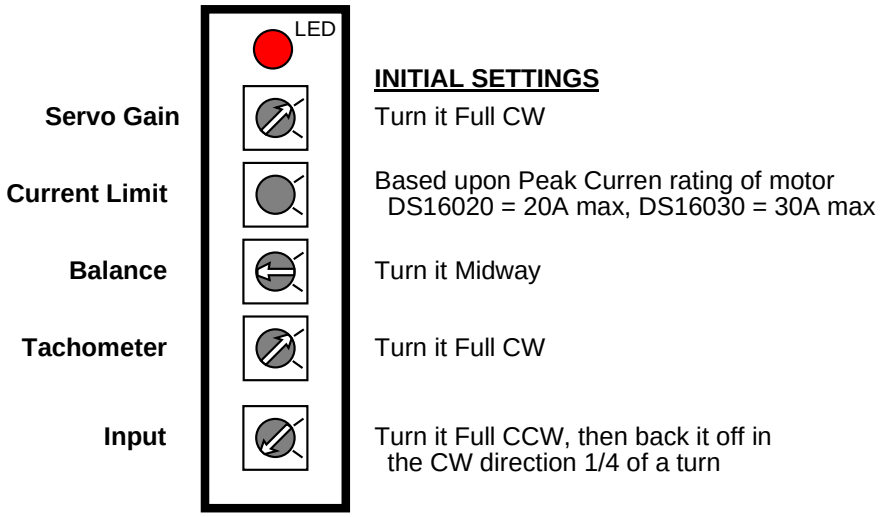


Figure 6-22. Amplifier Potentiometer Layout



The initial setting of the Current Limit potentiometer is based upon the peak current rating of the motor. If the user has a motor with a 10A peak current rating and a DS16020 that has a maximum current output of 20A, set the Current Limit potentiometer to midway for a representation of 10A. Then back it off 1/8 turn in the CW direction. Full CW allows the minimum amount of current through and full CCW allows the maximum.

c. Adjust the Servo Gain potentiometer on the amplifier by first enabling the axis and then turning the potentiometer CCW until the motor oscillates (i.e., the stage vibrates). The motor will produce a screeching sound when it oscillates. Back the gain off by turning it CW until the oscillation stops. Make another 1/8 turn CW from that position so it's not on the borderline of having the motor oscillate.

6. Prepare the Axis Scope Screen for tuning by performing the following functions:

   a. Press the maximize button on the Axis Scope Window so the Axis Scope window fills the entire screen.
   b. In the Collect menu select "2500 points."
   c. In the Display menu select "2500 points."
   d. In the Axis menu select Axis #1 (X Axis) or the axis that will be tuned.
   e. In the Plot menu select Velocity Command and Position Error.
   f. In the Trigger menu set the Forward Motion... and Reverse Motion... to a typical move such as *LINEAR X1 F180* for Forward Motion... and *LINEAR X-1 F180* for Reverse Motion... Also set the Sample Rate to 1.
   g. In the Tools menu select Status, Control, and Gains.

   When the "Single" button is pressed, Axis #1 will first move as specified by the Forward Motion.... When the Single button is pressed again, Axis #1 will move as specified by the Reverse Motion....

7. Adjust the Kpos position gain in order to get the position error to end at or near the same time the Velocity Command ends. The adjustment to Kpos is made by entering a value in the Kpos box on the Axis Scope window. Refer to Figure 6-2. If Kpos is set too high, the position error will visibly oscillate and the motor will vibrate. The user is not striving to reduce the position error, though that will happen. However the axis needs to be rough tuned because the following step will be to fine tune the potentiometers on the amplifier.



Since all Servo Gains are set to zero, the user must set the Kpos to a starting value, otherwise the stage won't move. Normal starting values are 10, 100, 1000, etc.

Once the stage is moving, the user should see a graph similar to the one in Figure 6-23. This graph illustrates that Kpos is too low. The stage moves slowly in the positive direction and then in the negative direction.

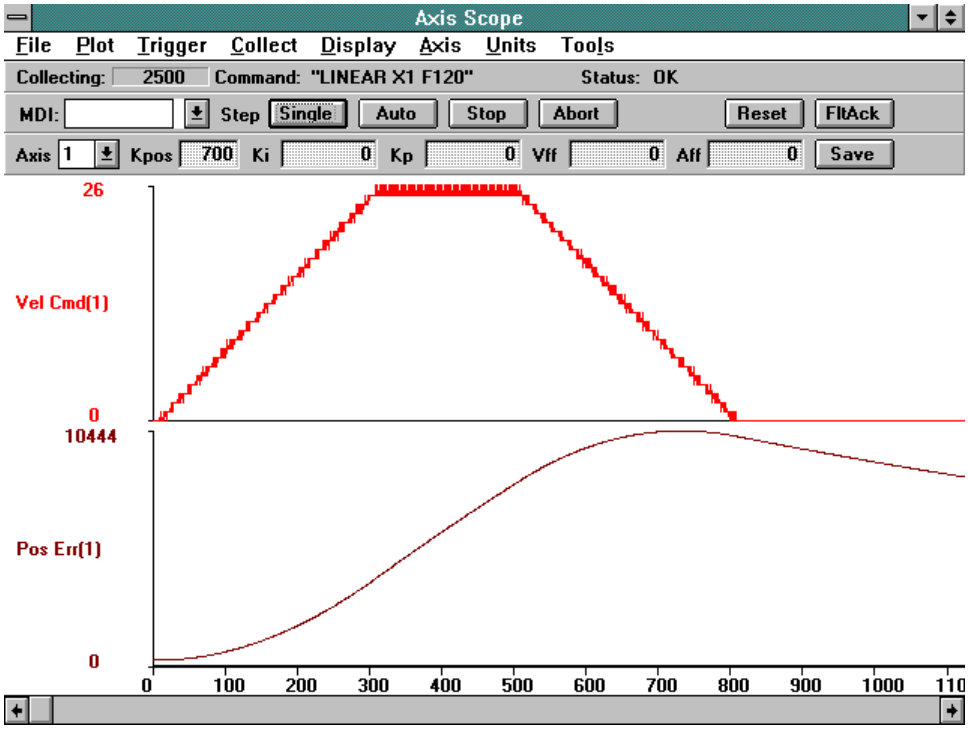


Figure 6-23. Axis Scope Window

If the motor doesn't move then Kpos is too low. Increase the value of Kpos and try again by pressing the Single button.

The stage may want to drift away on its own when it is enabled. Increasing Kpos will stop this.

If the user is adjusting the gains that Aerotech has provided for the system, use the existing Kpos as the starting point.

When Kpos is increased, the position error is beginning to end at or near the end of the commanded move as illustrated in Figure 6-24. The axis is roughly tuned, so continue with following step.

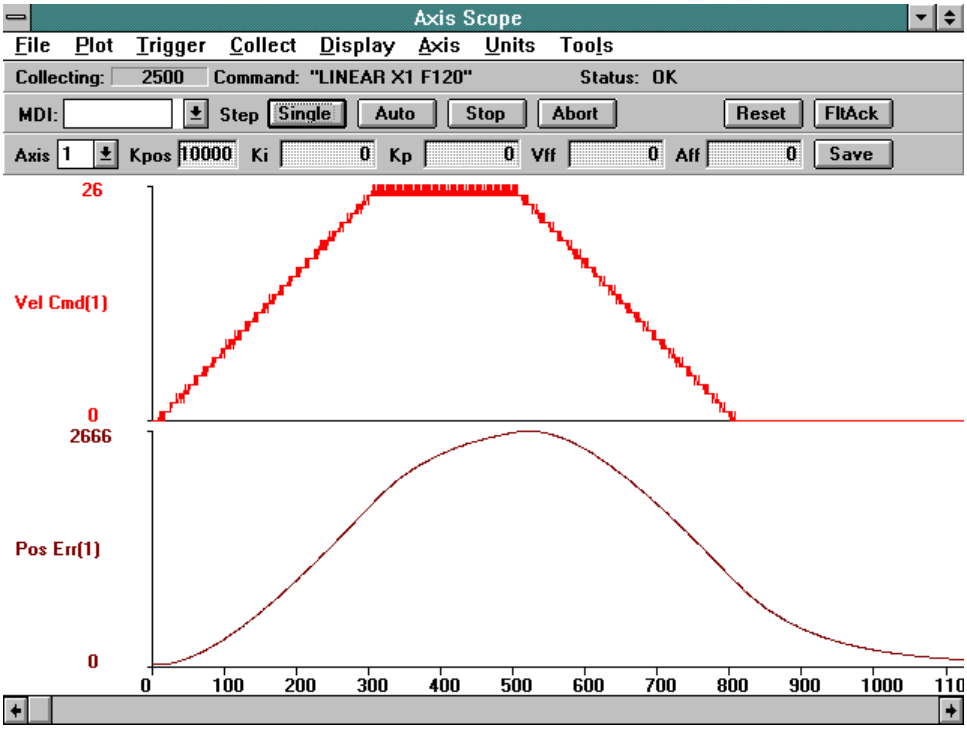


Figure 6-24. Plot Showing a Roughly Tuned Axis (When Adjusting Kpos)

8. This step requires fine tuning the amplifier settings. First, adjust the Balance pot on the amplifier in order to remove any DC offset in the position error. Press the Auto button to repetitively move the stage in the forward motion and reverse motion. While the stage is moving, adjust the Balance pot and remove any DC offset in the position error. Press the STOP button when the task is done. This is shown in Figure 6-25.

   Second, the user will fine tune the Current Limit pot on the Aerotech DS16020/16030 amplifiers after commanding the motor to move short, fast moves and observing the current feedback from TP1 on the amplifier with an oscilloscope. In order to do this, perform the following steps.

   a. Connect the O-scope leads to TP1 (current feedback) and TP4 (common) on the amplifier.
   b. Select the Trigger menu on the Axis Scope window and set up the Forward Motion... and Reverse Motion... to represent a short fast move.
   c. Press the Auto button and allow the stage to repetitively move in the forward and reverse motion.
   d. While the stage is moving, adjust the Current Limit pot to clamp the current to either 4 times the continuous current rating of the motor or the peak current rating of the motor, whichever is less.

   The current feedback on TP1 is 3 amps per volt, so a 2 volt signal on the O-scope would represent 6 amps. Press the Stop button when complete. Figure 6-26 illustrates what the user will see after one move.

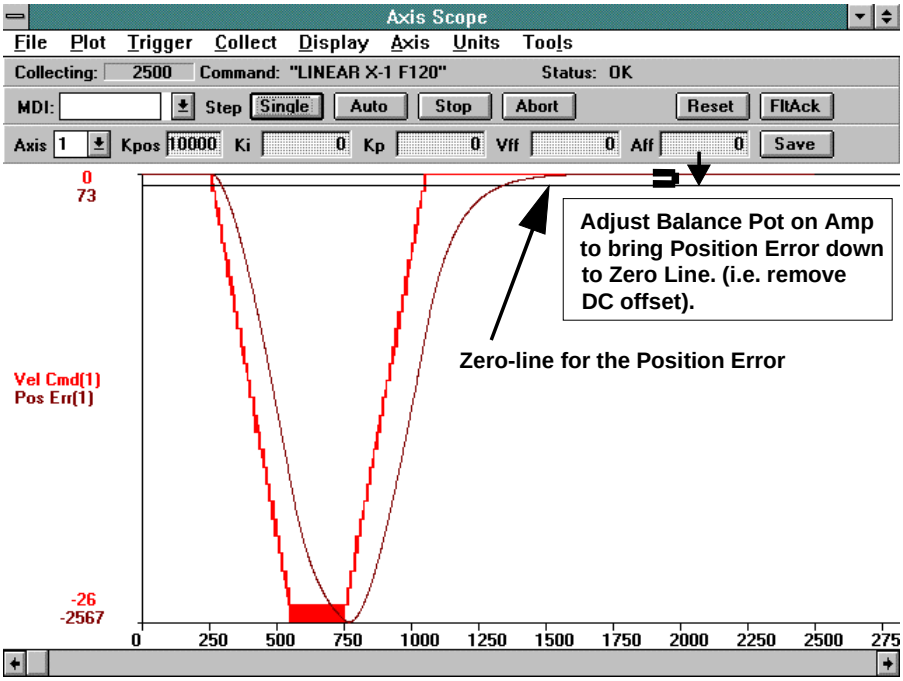


Figure 6-25. Plot Showing the Removal of DC Offsets in the Position Error

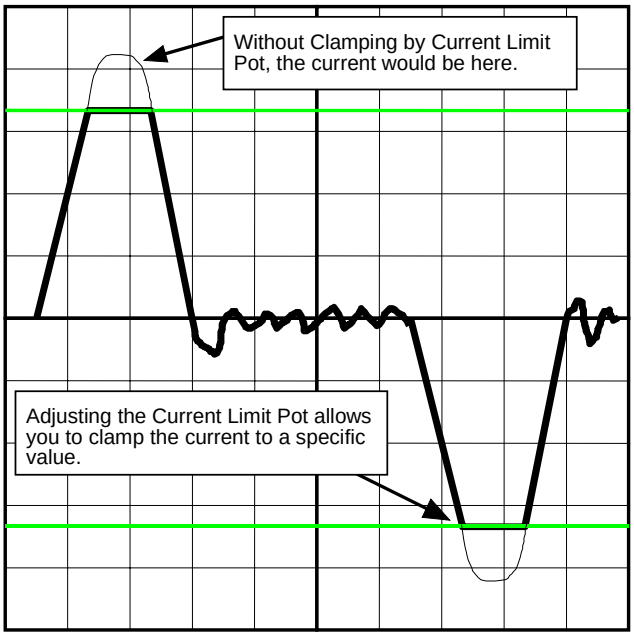


Figure 6-26. O-scope Showing Current Feedback for One Move

Third, if necessary, the user may have to fine-tune the Input pot if unable to achieve maximum speed for the motor. To fine-tune the Input pot, perform the following procedure:

- Connect the O-scope to TP5 (Velocity Command) and TP4 (common) on the amplifier.
- Select the Trigger menu on the Axis Scope window and set up Forward Motion... and Reverse Motion... to represent a move at 1/2 of the maximum speed.
- Press the Auto button and allow the stage to repetitively move in the forward and reverse motion.
- While the stage is moving, adjust the Input pot so that when the motor is moving at 1/2 speed the Velocity Command on TP 5 is 4 volts.
- Press the Stop button when completed.

9. Finish adjusting the Position Loop (Kpos) where the main concern is to strive for smoothness in the position error and to have the position error end at or near the same time the Velocity Command ends. After repeating the process of starting and stopping the axis and adjusting Kpos the graph should look like Figure 6-27.

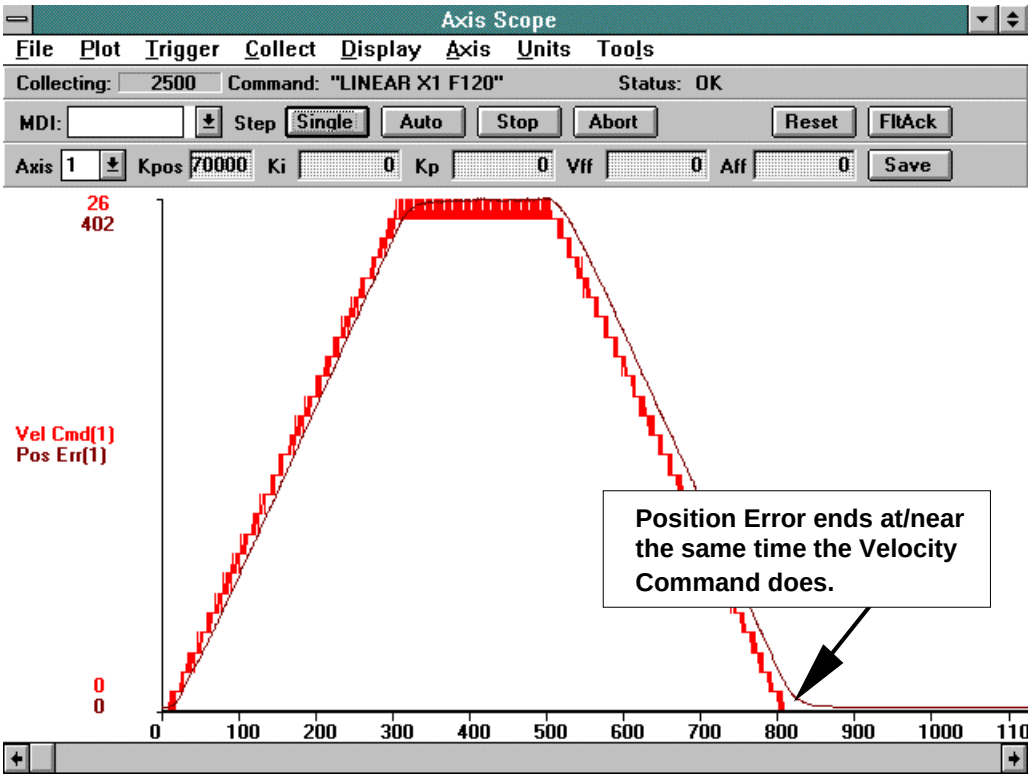


Figure 6-27. Plot Illustrating Smoothness in the Position Error

Referring to Figure 6-27, the position error ends at or near the end of the Velocity Command. The point where the user stops adjusting the Kpos depends upon how much settling time is allowed in the system.

If Kpos is too high, the motor will oscillate.

10. Adjust the In-Position Integrator (Ki) to remove any drift (DC offset) in the position error that might not have been removed with the Balance pot. Increasing Ki may help the position error to end closer to the end of the Velocity Command.

If Ki is too high, the settling time will increase as the position error begins to oscillate after the end of the Velocity Command.

Figure 6-28 is a plot that displays a case where Ki is too high and Figure 6-29 shows a plot of the position error with the appropriate Ki value.

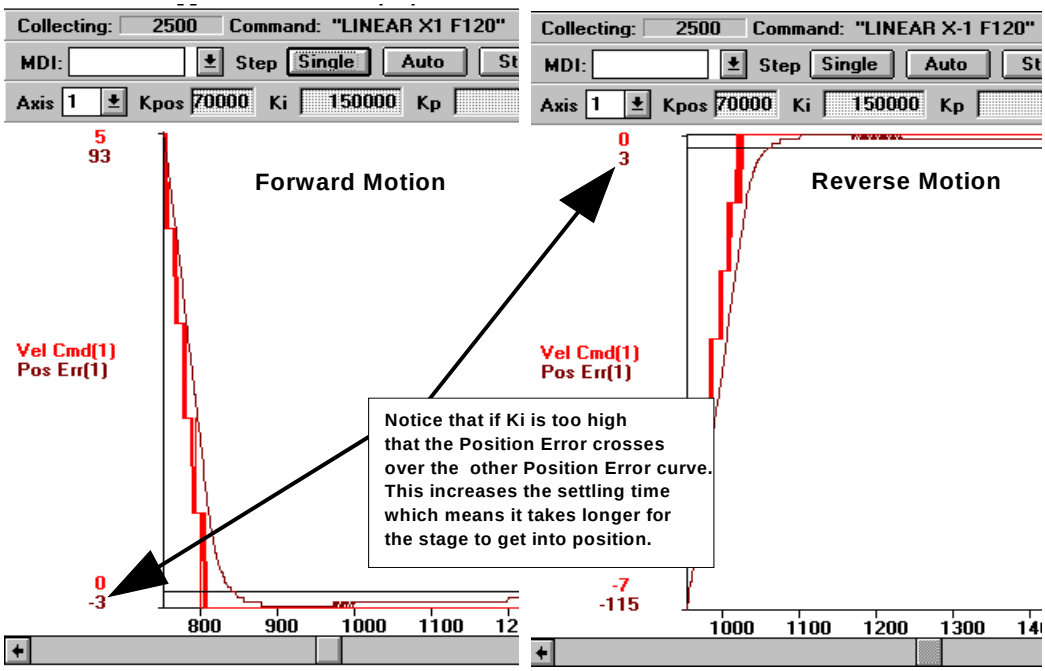


Figure 6-28. Plot Showing Effects on Position Error (When Ki is too High)

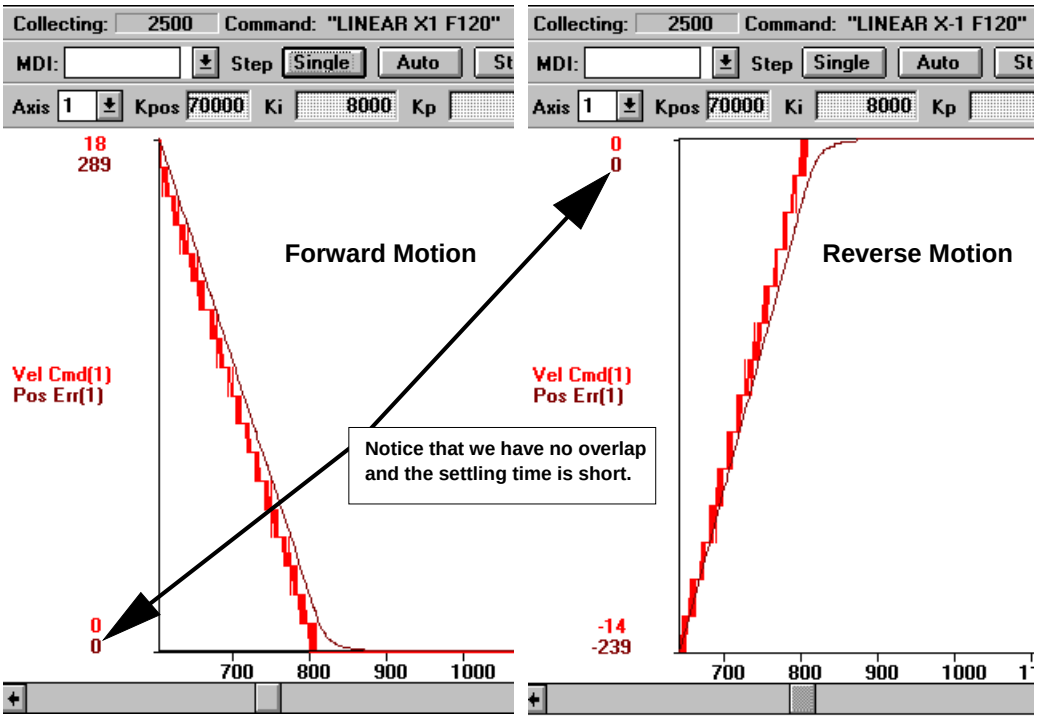


Figure 6-29. Plot of the Position Error With Appropriate Ki Value

11. Adjust the Velocity feedforward value to reduce the position error (if desired). When doing this, attempt to get the position error within 10 to 20 machine steps of error. Ensure the position error does not cross the zero line of the graph. If it does, Vff is set too high. The objective is to obtain smoothness in the Position Error and to get within 10 to 20 counts of error. Figure 6-30 illustrates that Vff has reduced the amount of position error. However, Vff still needs to be increased so the position error can be reduced some more. This is shown in Figure 6-31 where the position error has been reduced to within 10 counts of error. It is OK to allow some following error in the system.

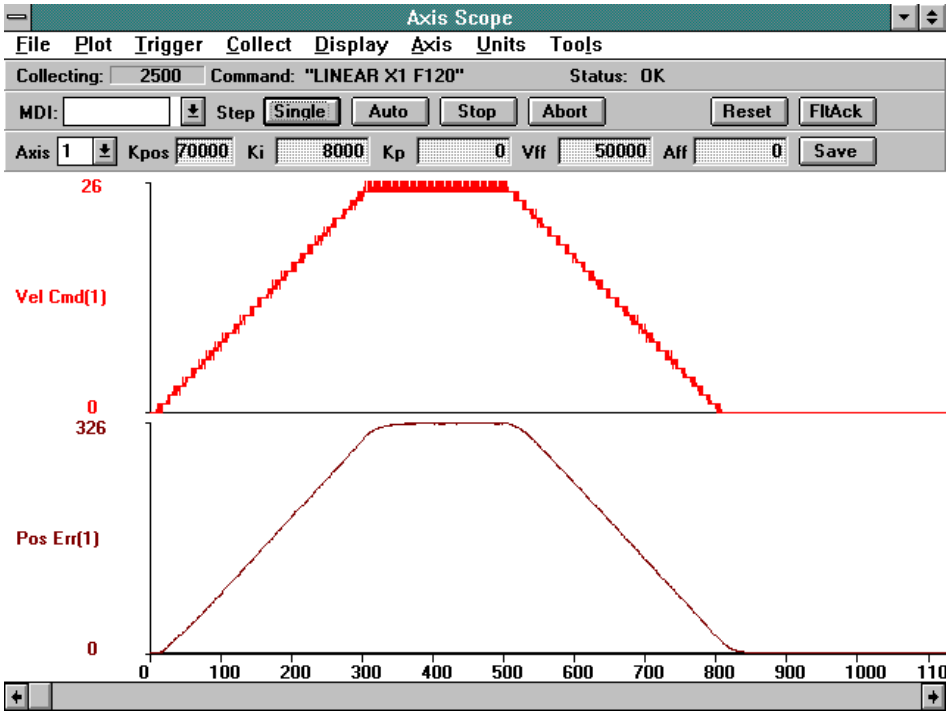


Figure 6-30. Position Error After Increasing Vff

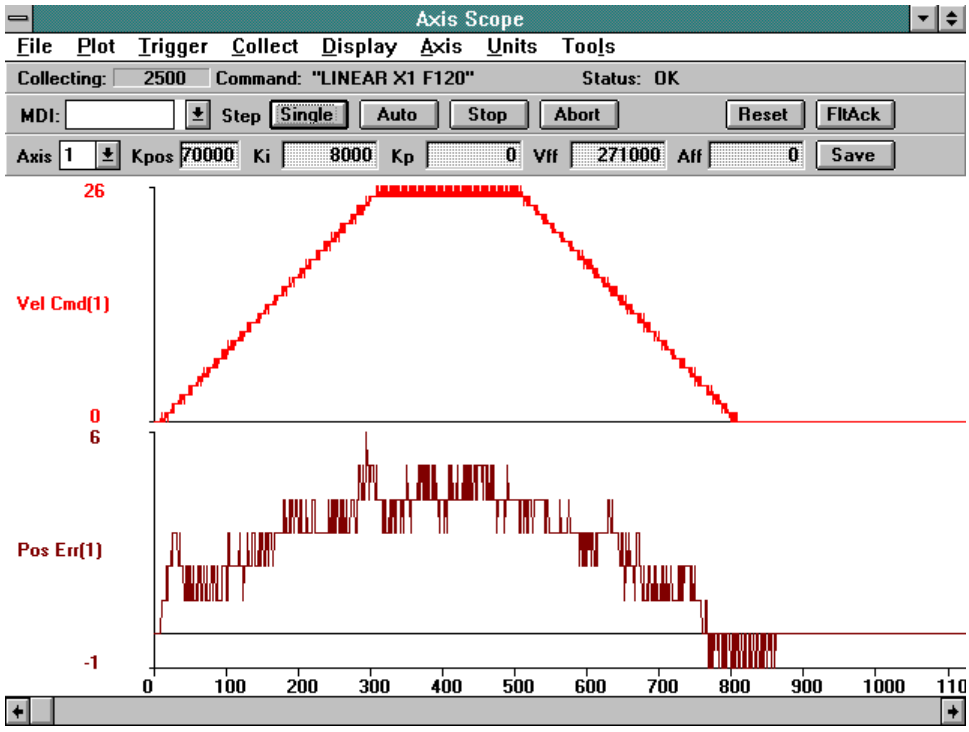


Figure 6-31. Position Error Reduced to Within 10 Counts of Error Using Vff

Shown in Figure 6-32 is plot of what happens when Vff is set too high and the position error crosses over the zero line. Notice that the Vff was not increased much and that the position error increased.

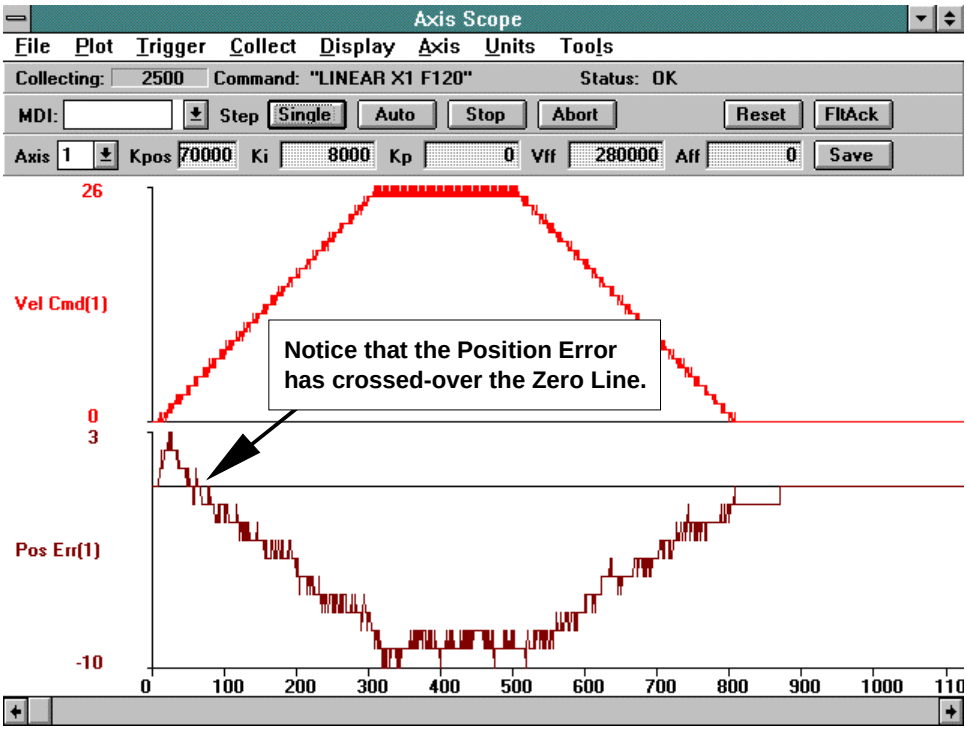


Figure 6-32. Plot of Position Error When Vff is too High

12. Turn the Position Error and Integral Error Traps on by returning to the Parameters window and selecting the Parameter tab called “Faults.” Turn the “Position Error” and the “Integral Error” back on by checking the boxes. This will reactivate these traps.

    Save and exit the Parameters window.

    Reinitialize the UNIDEX 500.

∇ ∇ ∇

CHAPTER 7: PROGRAMMING COMMANDS

In This Section:

- Introduction 7-1
- Program Execution Levels 7-2
- Mathematical Function Commands 7-3
- System Registers 7-6
- System Inputs $INP and $IN0-$INF 7-10
- iSBX I/O Interface (optional on U500 ISA only) 7-11
- 4EN Expansion Board I/O Interface (optional on U500 ISA only) 7-14
- Programming Commands 7-15

7.1. Introduction

The UNIDEX 500 decodes most programming commands by the first two characters, although the user may include more as desired for clarification. Other commands require a specific subset of letters or the entire command to be specified. The commands are not case sensitive. Throughout this chapter the commands appear in uppercase letters for easy recognition. Certain RS-274 codes may also be used to input certain commands. These are discussed later in this chapter.

This chapter uses the typographical conventions listed in Table 7-1. Required command letters are shown in uppercase bold, unnecessary command letters are shown lowercase, normal type, and may be included in your program for clarity.

Table 7-1. Programming Conventions Used in This Manual

Example	Description
INDEX	Uppercase bold letters are used to indicate terms used at the operating system command level.
distance	Words in italic indicate information that you **must** supply to validate the command.
[[option]]	Items between double brackets are optional.
BRAKE {*on*|*off*}	Braces and a vertical bar indicate a choice among two or more items. You must choose one of the items unless double square brackets [[{ }]] surround the braces.
MAP *plane,drive,axis...*	Three dots following an item indicate that more items having the same form may be included.
⋮	A column of dots indicates that part of an example program has been omitted.
ENTER	Capital letters signify names of keyboard keys.

Several command arguments are entered using a single character. These argument designators are listed in Table 7-2.

Table 7-2. Single Character Arguments for Programming

Argument	Meaning
X Y Z U	Each of the four axes names
C	Designates the center point for circular motions
F	Contour Feedrate
V	User's variable, V0 through V255

The UNIDEX 500 is available in three levels: BASE, PLUS, and ULTRA. Some commands and features are not available in the BASE and PLUS levels. Each command (listed in the following sections) will indicate its board level requirement.

7.2. Program Execution Levels

There are three levels of programming the UNIDEX 500 Series:

- Board level execution
- C library interface (QLB50032.DLL)
- C library ascii interface (WIN50032.DLL)

7.2.1. Board Level Execution

The U500 executes all motion commands at the board level. These commands are sent from the PC to the U500's queue FIFO. Commands are executed sequentially.

7.2.2. C Library Interface

The quick libraries (QLB50032.DLL) are the next higher levels of execution after the board level commands. All board level commands are processed and sent by the quick libraries to the board.

7.2.3. C Library ASCII Interface

The C library ascii interface (WIN50032.DLL) is the highest level of execution. This interface interprets the ascii level programming commands. The MMI and typical custom programs will interface with the WIN50032.DLL. This DLL, in turn, will make the necessary calls to the quick libraries and the U500 board to properly execute each command. Program flow commands, such as looping and branching, are examples of commands that are executed at the quick library level.

7.3. Mathematical Function Commands

The U500 library interpreter adds the functionality of variables, operators, and functions to ease motion program writing. These are described below.

7.3.1. Direct Variables (V0 through V255)

MMI permits the use of direct variables throughout a program and within functions. The format for these variables is:

V*n*, where $0 \le n \le 255$.

There are 256 general purpose double precision direct variables available, labeled V0 through V255. These variables are stored in the ascii interface C library (WIN50032.DLL).

These variables are initialized to zero after system initialization.



Numeric constants may be specified as *floating point*, *exponential*, or *hexadecimal* formats. Variables are automatically formatted in either floating point, exponential, or hexadecimal formats. For example:

```
V12 = 34.395        ;Floating point number
V200 = 0x3F         ;Hexadecimal integer format
V106 = 1.257e-7     ;Exponential format
```

7.3.2. Indirect Variables (VV0 through VV255)

Variables may be addressed indirectly. That is, the actual variable number is itself a variable. This is a very powerful feature, permitting treatment of variables as though they were a single dimensional array. The format for an indirect variable is:

VV*n*, where $0 \le n \le 255$

For example, assume that V12=56, then VV12 actually refers to V56. V56 may equal some other unrelated value. For example:

```
V35 = 10.237        ;Assigns 10.237 to variable V35
V0 = 35             ;Assigns 35 to variable V0
V1 = VV0            ;Assigns value of V35 to V1
```

In this case, the value of the variable number specified by V0 (e.g., V35, which equals 10.237) is assigned to variable V1.

7.3.3. Functions

The C libraries recognizes certain mathematical functions. The general format for these functions is:

FUNCTION_NAME (*arguments*)

Table 7-3 summarizes the functions that are executed by the C library ascii interface, as well as a description of each function and examples.

Table 7-3. Supported Functions

Function	Description	Examples
DEG(*radians)*	Converts radians into degrees	V0=DEG(0.25) ;V0=14.3293 degrees
RAD*(degrees)*	Converts degrees into radians	V2=RAD(35) ;V2=0.6108 radians
TAN*(angle)*	Calculates the tangent of *angle* (where *angle* is given in radians)	V17=TAN(0.785) ;V17=0.9992
ATN*(arg)*	Calculates the arctangent (inverse tangent) of argument (where *arg* is dimensionless, and the result is in radians)	V68=ATN(1) ;V68=0.7853981 V32=DEG(ATN(1)) ;V32=45 degrees
SIN*(angle)*	Calculates the sine of the term *angle* (where *angle* is in radians and the result is dimensionless)	V56=SIN(5) ;V56= -0.958924 V71=SIN(RAD(30)) ;V71=0.5
ASIN*(arg)*	Calculates the arcsine (inverse sine) of the argument (where *arg* is dimensionless, and the result is in radians)	V90=ASIN(-0.958924) ;V90=5 radians
COS*(angle)*	Calculates the cosine of the term *angle* (where *angle* is in radians and the result is dimensionless)	V22=COS(4) ;V22= -0.653643 V71=COS(RAD(30)) ;V71=0.8660254
ACOS*(arg)*	Calculates the arccosine (inverse cosine) of the argument (where *arg* is dimensionless and the result is in radians)	V38=ACOS(0.5) ;V38=1.0471975
SQR*(pos_num)*	Calculates the square root of *pos_num*	V34=SQR(36) ;V34=6
ABS*(number)*	Returns the absolute value of *number*	V84=ABS(-12.876) ;V84=12.876

7.3.4. Operators and Evaluation Hierarchy

Constants, functions, and variables may be combined using the mathematical operators listed in Table 7-4.

Table 7-4. Mathematical Operators and Their Evaluation Hierarchy

Operator	Function	Priority
( )	Grouping	Highest
* , / , ^ , | , &	Multiplication, Division, Exponentiation, Bitwise OR, Bitwise AND	↓
+ , - , || , &&	Addition, Subtraction, Logic OR, Logic AND	↓
= , < , > , <> , <= , >=	Equate or Assignment, Less than, Greater than, Not equal to, Less than or equal to, Greater than or equal to	Lowest

Below are some examples of using operators.

```
V0=SQR(V11)+ABS(COS(V23))-SIN(RAD(V23-12*V32))
V1=V1+1.5
V2=V3+V4+VV10*3
V0=SINP&0x3    ;V0=2, the bitwise AND of hex F and 2
V0=2|1         ;V0=3, the bitwise OR of 2 and 1
```

7.4. System Registers

The C libraries use predefined registers to designate axis positions.

7.4.1. Relative Position Registers

Relative position registers represent the commanded axis position with respect to the software home position. These registers can be set to any value using the SOFTWARE HOME or G92 command. This allows the user to define an offset for programming convenience.

The SOFTWARE POSITION command should be used to update the registers after an unsynchronized move such as SLEW or FREERUN is completed. Pressing the ABORT key or executing the ABORT command will also update the registers. The return value is in program steps. See the PROGRAM command for more information on "program steps." Refer to Table 7-5.

Table 7-5. Relative Position Registers

Register	Meaning
$XRP	X axis relative position, in program steps
$YRP	Y axis relative position, in program steps
$ZRP	Z axis relative position, in program steps
$URP	U axis relative position, in program steps

7.4.2. Absolute Position Registers

Absolute position registers represent the commanded axis position with respect to the hardware home position. These registers are cleared only after successfully executing a HOME command. The SOFTWARE POSITION command should be used to update the registers after an unsynchronized move such as SLEW or FREERUN is completed. Pressing the ABORT key or executing the ABORT command will also update the registers. The return value is in program steps. See the PROGRAM command for more information on "program steps." Refer to Table 7-6.

Table 7-6. Absolute Position Registers

Register	Meaning
$XAP	X axis commanded position referenced from the HOME position
$YAP	Y axis commanded position referenced from the HOME position
$ZAP	Z axis commanded position referenced from the HOME position
$UAP	U axis commanded position referenced from the HOME position

7.4.3. Real Time Feedback Position Registers

Real time feedback position registers represent the axis position from the feedback device (encoder, resolver, etc.) with respect to the software home position. This is the feedback position input to the servo loop. The difference between the Real Time Feedback Position and Real Time Commanded Position is Position Error. The value returned from the register is in program steps. See the PROGRAM command for more information on "program steps." Refer to Table 7-7.

Table 7-7. Real Time Feedback Position Registers

Registers	Meaning
$XFP	X axis real time feedback position referenced from the SOFTWARE HOME
$YFP	Y axis real time feedback position referenced from the SOFTWARE HOME
$ZFP	Z axis real time feedback position referenced from the SOFTWARE HOME
$UFP	U axis real time feedback position referenced from the SOFTWARE HOME

This position register does not need updated by the SOFTWARE POSITION command.



7.4.4. Real Time Commanded Position Registers

Real time commanded position registers represent the axis position that is commanded by the UNIDEX 500 with respect to the software home position. This is the real time position command to the servo loop. The difference between the Real Time Feedback Position and Real Time Commanded Position is Position Error. The value returned from the register is in program steps. See the PROGRAM command for more information on “program steps.” Refer to Table 7-8 for the real time commanded position registers.

Table 7-8. Real Time Commanded Position Registers

Registers	Meaning
$XCP	X axis real time commanded position referenced from the SOFTWARE HOME
$YCP	Y axis real time commanded position referenced from the SOFTWARE HOME
$ZCP	Z axis real time commanded position referenced from the SOFTWARE HOME
$UCP	U axis real time commanded position referenced from the SOFTWARE HOME

This position register does not need updated by the SOFTWARE POSITION command.



7.4.5. Understanding the Concept of Program Steps

The measuring unit called "Program Steps" is based upon the number of decimal digits that is displayed with the MMI software. For example, if the number of decimal digits displayed is 4, then the smallest move displayed in MMI's position display is 0.0001. The number 0.0001 is considered 1 "Program Step" since it is the smallest unit that can be programmed. Knowing this, a 1.0 in the position display is equivalent to 10000 "Program Steps." Also, if the user displays 3 decimal digits, the smallest step is 0.001 and is equivalent to 1 "Program Step." The following formula applies:

$$Program\ Units = \frac{Value\ from\ Position\ Register}{10^{Number\ of\ Decimal\ Digits}}$$

7.4.6. A/D Channel Registers

The A/D Channel registers are used to read the values of the 4 A/D Channels on the U500 ISA and the values of the 8 A/D channels on the U500 PCI.

The standard U500 ISA has an 8-bit A/D and can read a voltage between 0 and +5VDC. The U500 ISA also has an optional 12 bit A/D that can read voltage between –10 and +10 VDC Table 7-9.

Table 7-9. U500 ISA A/D Channel Registers

Register	Signal Name	Description	UNIDEX 500		DR500	BB500	BB501
			Connector	Test Point	Connectors		
$AD0	AIN 1–(AIN1)	MFO input	P1-96	TP38	J13-25	P5-19	TB1-4
$AD1	AIN 2–(AIN0)	Spare	P1-95	TP37	J13-24	P5-18	TB1-3
$AD2	AIN 3	Joystick–Vertical	P1-90	TP40	J12-6	P5-21	J12-6
$AD3	AIN 4	Joystick–Horizontal	P1-89	TP39	J12-3	P5-20	J12-3

On the U500 ISA, the A/D is engaged using the ENABLE AD command and disengaged using the DISABLE AD command.

The U500 PCI has a 12 bit A/D that can read voltage between –10 and +10 VDC. Refer to Table 7-10.

Table 7-10. U500 PCI A/D Channel Registers

Register	Signal Name	Description	UNIDEX 500		DR500	BB500	BB501
			Main Connector	Secondary Connector	Connectors		
$AD0	AIN 1	MFO input	P1-96		J13-25	P5-19	TB1-4
$AD1	AIN 2	Spare	P1-95		J13-24	P5-18	TB1-3
$AD2	AIN 3	Joystick–Vertical	P1-90		J12-6	P5-21	J12-6
$AD3	AIN 4	Joystick–Horizontal	P1-89		J12-3	P5-20	J12-3
$AD4	AIN 5	Spare	P100-96	P9-46	J13-25	P5-19	TB1-4
$AD5	AIN 6	Spare	P100-95	P9-45	J13-24	P5-18	TB1-3
$AD6	AIN 7	Spare	P100-90	P9-40	J12-6	P5-21	J12-6
$AD7	AIN 8	Spare	P100-89	P9-39	J12-3	P5-20	J12-3

EXAMPLE:

```
V0 = $AD0        ; return the value of A/D register 0 in volts
```

7.5. System Inputs $INP and $IN0-$INF

System input commands may be used to return a 16 bit word having a value that corresponds to the state of all inputs or specified inputs. The command syntax for system inputs is:

$INP Returns a 16 bit word, each bit corresponding to the state (0 or 1) of an input

$IN*n* Returns an individual bit value (0 or 1) that corresponds to the state of a particular input.

The UNIDEX 500 normally has 16 input lines. The state of the inputs may be read using the $INP command. The value returned by $INP will be between 0 (all inputs low) and 65,535 (all inputs high). The value of $INP is the decimal equivalent of the 16 bit binary number denoted by the state of the 16 inputs.



Unused inputs are pulled high, so they contribute to the values of the $INP value. To remove the effect of these unused bits in $INP, you can either pull them low with hardware, or mask them in software (using C or Visual BASIC, for example) using the appropriate Boolean AND function. For example, if the 8 most significant bits of $INP are unused (i.e., they appear as 1's in the bit map), you could eliminate their effect in $INP by ANDing the current $INP value with the bit mask 0000 0000 1111 1111 (00FF hex). Refer to the appropriate C or Visual BASIC programming manuals for the proper syntax for performing a bit-by-bit Boolean AND function.

When an input is high, that input line contributes the following amount to the $INP value:

$x = 2^n$ where n = the input number (0 to 15).

Individual bits can be read by the $IN*n* command. The term *n* is the hexadecimal number (0 through F) which represents the respective input number (0 through 15). The value of the input may be determined with the following logic:

If $IN*n* = 2^n	then that input is high
If $IN*n* = 0	then that input is low
$IN5	gives the binary state of input number 5
$INF	gives the binary state of input number 15

Multiple inputs may be checked using an additive process. For example:

V0 = $IN5 + $IN8 + $INA or V0 = $INP&0x520.

7.6. iSBX I/O Interface (optional on U500 ISA only)

The iSBX I/O board connects to connector P4 on the U500 ISA. There are 3 available ports on the iSBX board. The 8 lines of ports A and B are configured as group as either inputs or outputs, the 8 lines of port C can be separated into 4 bit group each group can be an output or an input.

If the iSBX board is of the 48 I/O version from WinSystems, Inc. Two 50 pin 3M style connectors are available on the board. Bank 0 of the 24 I/O bits are accessed on the J2 connector and are addressed by using the $0xx commands. Bank 1, the second set of 24 I/O, are on the J1 connector and are addressed by the $1xx commands.

The UNIDEX 500 commands to communicate with the iSBX board are:

$00 - $0f = iSBX 1 addresses

$10 - $1f = iSBX 2 addresses

$XXn = accesses individual bits

The syntax of the commands are as follows: the first digit following the "$" indicates which bank of 24 I/O bits to access, $0xx addresses the first 24 bits, $1xx addressed the second 24 bits. The second digit following the "$" indicates the port, which 8 bits within the bank to address, $x0x refers to Port A, $x1x is Port B, $x2x is Port C, $x3x addressed the control word which configures the ports as outputs or inputs. The third digit is optional and indicated which bit of the port to access, $xx0 is bit 0 up to $xx7 which is bit 7 of the port. For example, $123 addresses bit 3 of port C on the second bank of 24 bits.

To configure the ports to the desired basic (no handshaking) input/output configuration, a control word must be written to the control port. The control port is address $03 for bank 0 and $13 for bank 1. The control words are for interfacing to the Intel 82C55A Programmable peripheral interface IC. If a different interface chip is being used, consult the data sheet for that chip. The 16 possible control words (in hex) for configuring basic IO are (A = 8 bits of port A, B = 8 bits of port B, Cl = bits 0-3 of port C, Ch = bits 4-7 of port C):

Table 7-11. ISBX Control word Configurations

Control Word	Inputs	Outputs
80h	none	A, B, Cl, Ch
81h	Cl	A, B, Ch
82h	B	A, Cl, Ch
83h	B, Cl	A, Ch
88h	Ch	A, B, Cl
89h	Cl, Ch	A, B
8ah	B, Ch	A, Cl
8bh	B, Cl, Ch	A
90h	A	B, Cl, Ch
91h	A, Cl	B, Ch
92h	A, B	Cl, Ch
93h	A, B, Cl	Ch
98h	A, Ch	B, Cl
99h	A, Cl, Ch	B
9ah	A, B, Ch	Cl
9bh	A, B, Cl, Ch	none

After writing the control word, the inputs can be read to UNIDEX 500 variables and the outputs can be set. Port A is address $00 or $10, Port B is $01 or $11, port C is $02 or $12. On reset of the U500 card, the ports are set to input state and the bits are set to high impedance state. Once a control word is written and a port is set as an output, the bits are pulled low. To avoid undesired output states, after writing a control word, immediately follow it by setting the output ports properly.

The following is an sample program that sets outputs and reads inputs of the iSBX card.

```
$03=144              ;configure for port A as input, B and C as outputs
                     ;control word 144 = 90h
$01=255              ;set outputs of port B high
$02=85               ;turn on even outputs of port C
v0=$000              ;read bit 0 of port A
me di "bit 0 = %v0"  ; v0 = 0 or 1
v0=$001              ;read bit 1 of port A
me di "bit 1 = %v0"  ; v0 = 0 or 2
v0=$002
```

```
me di "bit 2 = %v0"   ; v0 = 0 or 4
v0=$003
me di "bit 3 = %v0"   ; v0 = 0 or 8
v0=$004
me di "bit 4 = %v0"
v0=$005
me di "bit 5 = %v0"
v0=$006
me di "bit 6 = %v0"
v0=$007               ; read bit 7 of port A
me di "bit 7 = %v0"   ; v0 = 0 or 128
```

If a Opto-24 board is being used, $x2 references bit 0 through 7 in reverse order, $x20 = bit 7, $x27 = bit 0. $x1 references bits 9 – 15 with $x10 = bit 15, $x17 = bit 9. $x0 references bits 16 - 23, with $x00 - bit 23, $x07 = bit 16.

7.7. 4EN Expansion Board I/O Interface (optional on U500 ISA only)

The "4EN" expansion board can be used with the UNIDEX 500 to provide additional I/O lines. Parameter 99 “Option Board Setup Code” must be set properly to use the expansion board. Refer to Chapter 5 “Parameters” for the parameter settings.

The 4EN expansion card occupies one PC-AT ISA bus slot. The PC bus connector is used for power only. Interface to a UNIDEX 500 card is through the P3 connector. This is a 50 pin ribbon cable which connects directly to the U500's P3 connector.

The 4EN card contains four sets of OPTO22 compatible I/O. Each set is referenced by a bank number. This bank number is used in the CNC I/O commands "OEn" and "INn" where n is the bank number. See Section 7.8.49. and Section 7.8.66. for more information.

Table 7-12. 4EN I/O Connector Interface

Connector	Direction	Bank Number	PB24 Module #
P7	24 outputs	1	0-23
P8	24 inputs	2	0-23
P9	16 inputs	3	8-23
	8 outputs	3	0-7
P10	8 outputs	4	0-7

The module type inserted into the OPTO22 board (PB24,16,8) determines the direction of the signal. The least significant bit of the "OEn"/"INn" commands corresponds to the lowest module number.

7.8. Programming Commands

The following section provides a reference for UNIDEX 500 commands. The commands listed in Table 7-13 are accompanied by their associated RS-274 codes (if applicable), a brief description, and the page number from this manual where additional information can be found. Most commands in the table are written with the first two characters capitalized. These represent commands that the U500 can recognize when only the first two letters are typed. The remaining commands are recognized only if the characters in capital letters are typed. Exceptions are the OEn and INn commands which use "n" to represent a number that must be typed in order for the command to work.

Table 7-13. U500 Programming Command Summary

Command	RS-274 Code	Description	Page Number
$IN0 to $INF	None	Individual input bit	7-10
$INP	None	16 bits of input data	7-10
$XAP, $YAP, $ZAP, $UAP	None	Signifies absolute commanded position of axes	7-6
$XRP, $YRP, $ZRP, $URP	None	Signifies relative commanded position of axes	7-6
$XFP, $YFP, $ZFP, $UFP	None	Represents the real-time axis position from feedback device with respect to the software home position	7-7
$XCP, $YCP, $ZCP, $UCP	None	Represents the real-time axis position that is commanded by the U500 with respect to software home position	7-7
$AD0, $AD1, $AD2, $AD3	None	Used to read the values of 4 A/D channels	7-8
ABS	None	Math routine producing the absolute value function	7-4
ABort	None	Abort motion command	7-22
ACceleration	None	Specifies the maximum acceleration/deceleration rates for each axis	7-22
AC PL (ACcel PLane)	None	Limit the acceleration during linear and circular moves by lowering the feedrate and adjusting the ramp time.	7-23
ACOS	None	Math routine producing the arccosine function	7-4
AFCO (AUTO FOCUS)	None	Enables a secondary position loop using the analog input for feedback.	7-23
AGain	M47	Sends program flow to the first line of the command set	7-27
Analog		Generates an output voltage proportional to the axis position or the velocity	7-28
Array		Stores array data for use with the FIRE command.	Chapter 8
ASIN	None	Math routine producing the arcsine function	7-4
AT	None	Sends a sinusoidal excitation signal to an axis	7-30
ATN	None	Math routine producing the arctangent function	7-4
BOard	None	Selects the UNIDEX 500 board that is active	7-31

Table 7-13. U500 Programming Command Summary (Continued)

Command	RS-274 Code	Description	Page Number
BRake	None	Engages or disengages the brake option	7-32
CAL	None	Downloads axis calibration data from a file	7-33
CALLDLL	None	Used from MMI program to call any DLL	7-34
CCw	G3	Counterclockwise contouring of one to four axes	7-37
CLRSCR (Clear Screen)*	None	Clears the message display window	7-42
CM (Contouring Mode)*	None	Used to set normal contouring mode (CM 0) or enhanced contouring mode (CM1)	7-42
COM FREE**	None	Disables the COM port in the software interface and allows other applications to use it	7-43
COM GET**	None	Used to define one or more system variables that will hold numerical data received by the COM port specified by COM INIT	7-43
COM INIT**	None	Used to initialize a selected COM port of the PC to specified parameters	7-45
COM RECEIVE**	None	Used to write all data that is received from the COM port to the buffer specified by the COM SET _RECEIVE_ function.	7-45
COM SEND**	None	The COM SEND function is used to send information to the COM port that was specified in the COM INIT function.	7-46
COM SET_RECEIVE_**	None	Used to set the receive buffer to be a given file name.	7-47
COM SET_STATUS_**	None	Used to define a system variable (v0 through v255) where subsequent COM function status information is to be stored.	7-48
COM SET_ TERMINATE_**	None	Used to define a termination string (of ASCII values) that is used between devices that are communicating over the COM port.	7-49
COM SET_TIMEOUT_**	None	Used to define time-out values for the COM GET and COM RECEIVE functions before a time-out error occurs.	7-50
COS	None	Math routine producing the cosine function	7-4
CS (Command Scope)*	None	Can be used to collect actual velocity, command velocity, or torque information from the U500 DSP	7-51

Table 7-13. U500 Programming Command Summary (Continued)

Command	RS-274 Code	Description	Page Number
Cutter Compensation Commands*	G40	Cutter comp. off	7-52
	G41	Cutter comp. on LEFT	7-52
	G42	Cutter comp. on RIGHT	7-52
	G43	Define cutter radius	7-52
	G44	Define compensated axes	7-52
CVI	None	Convert given value to integer	7-55
CYcle	None	Maps an input bit to the cycle start function	7-56
CW	G2	Clockwise contouring of one to four axes	7-37
DAc	None	Establishes current command output digital to analog conversion (DAC) number	7-57
DEG	None	Math routine to convert radians to degrees	7-4
DIsable	None	Disables the motor of the specified axis	7-58
DS (Display Servo)*	None	Provides real-time servo loop display through unused digital to analog conversion (DAC) channels	7-59
Dual		Tracks any combination of two axes and will generate an output pulse at the vector distance specified	Chapter 8
DY (Dynamic Gain)	None	The DY command is used to set the dynamic mode for the position loop gain, Kpos .	7-60
Dwell	G4	Inserts a delay of an established duration into the program	7-60
EC	None	This command will calculate the correction factor for the wavelength of light based upon the actual temperature, pressure and humidity.	7-61
Enable	None	Enables the motor of the specified axis	7-63
ERror	None	Defines a new system error mask	7-64
Exit	M2	Terminates program flow	7-65
FA (Fault ack.)	None	Performs same function as pushing FLTACK key	7-66
FL (Filter Time Constant)*	None	Used in conjunction with the alternate contouring mode. Activates an exponential filter on the specified axis	7-66
FIRE		Tracks the position of a single axis encoder channel and can generate an output pulse at a specified number of encoder counts.	Chapter 8
FREEDLL	None	Frees a dll from memory after the LOADDLL command was used.	7-67
FReerun	None	Produces background motion of a designated axis	7-68
GAin	None	Sets servo loop related values	7-69
GEar	None	Move axis based on feedback from another axis	7-70
GOto	None	Directs program flow to a previously defined label or another program	7-71
HAlt	None	Stops all activity for a specified contour plane	7-73

Table 7-13. U500 Programming Command Summary (Continued)

Command	RS-274 Code	Description	Page Number
Home	None	Sends specified axes to the hardware home position	7-74
IF	None	Signals a conditional GOTO	7-75
IMmediate	None	Used with large program files larger than 64K	7-77
INdex	G0	Point-to-point non-synchronized motion of any or all axes	7-78
INn (Extended input on U500 ISA only)	None	Reads the inputs of the U500 card or the 4EN encoder card	7-79
INT	None	Interrupt program execution	7-80
IO (U500 PCI only)	None	Reads and/or writes the 24 IO bits on the P4 connector of the U500 PCI.	7-81
IOSET (U500 PCI Only)	None	Configures the 24 IO bits on the P4 connector of the U500 PCI.	7-81
JOG	None	Calls the Jog screen from a program	7-82
: (Label Marker)	None	ASCII string defining entry point in a program	7-83
Linear	G1	Initiates motion where each axis adjusts its feedrate to keep a contour path	7-83
LOADDLL	None	Keeps a DLL loaded in memory during execution for subsequent uses of the CALLDLL command.	7-84
Loop	None	Signals the beginning of program blocks to be repeated a designated number of times	7-85
LVdt	None	Conditions UNIDEX 500 for using a LVDT sensor for positioning	7-86
M0 (M zero)	M0	Initiates a pause in program flow to wait for a press of the cycle start key (F9)	7-87
MComm (Motor Commutate)	None	Sets current command for AC brushless motor phasing (commutation)	7-88
MEssage	None	Sends a message to the display, printer or a file	7-89
MR (Memory Read)	None	Reads the value of a DSP memory location	7-90
MSet (Motor setup)	None	Sets current command of an AC brushless motor for resolver	7-91
MW (Memory Write)	None	Writes a data value to a DSP memory location	7-94
NExt	None	Specifies the endpoint of the group of program blocks making up the loop	7-95

Table 7-13. U500 Programming Command Summary (Continued)

Command	RS-274 Code	Description	Page Number
OEn (Extended output on U500 ISA only)	None	Sets output bits on either the U500 card or 4EN encoder card	7-96
OUtput	None	Sets high or low condition for output bits of the output port	7-97
Pause	None	Maps an input bit to the pause function	7-100
PARallel	None	If the angle between contour moves (G1,G2,G3) is greater than the angle specified with this command, the U500 will switch to G9 mode.	7-98
Plane	None	Selects the contour plane to receive commands	7-101
PLC	None	PLC program setup and control	7-102
POSITION	None	An output bit can be set when one or more axes are within a specified error band of their final positions.	7-105
PRogram		Establishes the programming environment	7-106
	G70	English mode	7-106
	G71	Metric mode	7-106
	G90	Absolute mode	7-106
	G91	Incremental mode	7-106
PRM (Parameter Read)	none	This command is used to read the value of a parameter from the current parameter file.	7-99
QUeue	None	Specifies actions of control for the queue buffer storage area	7-108
RAD	None	Math routine to convert degrees to radians	7-4
RAmp	None	The time it takes each axis to accelerate from zero velocity to steady velocity	7-110
REFerence	None	Moves axis to marker position	7-111
REturn	None	Signals the end of a subroutine and directs program flow back to the block after the calling block	7-111
ROTate	None	Rotates parts	7-112

Table 7-13. U500 Programming Command Summary (Continued)

Command	RS-274 Code	Description	Page Number
ROunding		Conditions the deceleration behavior of a contour type move	7-113
	G23	Corner rounding on	7-113
	G24	Corner rounding off	7-113
	G25	specifies axis pairs for enhanced corner rounding	7-113
	G26	Enhanced corner rounding on	7-113
	G27	Enhanced corner rounding off	7-113
SCope	None	Tells the scope window to collect one set of data	7-116
SCF (Overriding Scale Factor)*	None	Enlarges, reduces, or mirrors a part	7-117
SKey	None	Reprogram function keys	7-118
SIN	None	Math routine producing sine function	7-4
Slave	None	The slave command is used to link an axis (slave axis) to a feedback channel (master axis) and start tracking based on the UINT_N input.	7-119
SLew	None	Conditions the UNIDEX 500 for immediate positioning by an optional joystick	7-124
SOftware	G92	Configures the UNIDEX 500 for various Software functions.	7-126
SQR	None	Math routine producing square root function	7-4
SPline	None	Cubic spline fitting	7-128
STart	None	Used to activate planes that are currently under the HALT command	7-129
SUbroutine	None	Directs program flow to a previously defined label or another program (See RETURN statement)	7-130
SYnc	None	Pauses queue execution until all corner rounding/velocity profile moves have completed	7-130
TAN	None	Math routine producing tangent function	7-4
TD (Target Disable)*	None	Disable target tracking mode on a single axis	7-131
TE (Target Enable)*	None	Enable target tracking on a single axis	7-131
TP (Target Position)*	None	Set tracking position for a single axis	7-131
TRajectory	None	Used to specify whether the acceleration and deceleration ramp type will be linear or inverse-sine	7-132

Table 7-13. U500 Programming Command Summary (Continued)

Command	RS-274 Code	Description	Page Number
TRIgger	None	Starts planes that are currently halted	7-133
UMfo	None	Overrides MFO potentiometer	7-133
V*	None	Signifies a user variable, V0 through V255	7-3
VAR	None	Read and write user variables to files	7-134
VElocity		Used to blend consecutive motions into one continuous path (velocity profiling):	7-135
	G8	Vel Profiling – ON	7-135
	G9	Vel Profiling – OFF	7-135
VV*	None	Signifies user array, VV0 to VV255	7-3
X, Y, Z or U	None	Single character argument for each axis	7-2
Window		Counters that can be linked to separate axis encoders.	Chapter 8
WAit	None	Initiates a program pause until all of the previous commands in queue are completed	7-144
WHile/ENDWhile	None	Evaluates an expression and if true executes to the ENDW statement	7-145

Table Notes: *No English language command. Use the command abbreviations or G code to implement.

**Unlike standard program commands, COM functions cannot be abbreviated using the first few letters.

For additional information on programming, refer to the programming examples that follow. Additional programs can be found in Chapter 9: Sample Programs.



The UNIDEX 500 programming commands that follow are compatible with all three models of the UNIDEX 500 (Base, Plus and Ultra) unless otherwise noted.





7.8.1. ABORT

The AB command aborts motion of the axes and clears the queue buffer. All enabled axes will ramp to a stop using the *Max ac/de* parameter (x16). The software position registers will be updated with the new position. The abort command is similar to pressing the ABORT key.

SYNTAX:

AB

There are no arguments needed with the AB command. The abort command is a real time, non-queued command.



7.8.2. ACCELERATION

The AC command is used to specify the acceleration/deceleration rate for each axis. This command overrides, but does not change the setting of parameter x16 (*Max ac/de [machine steps/ms/ms]*). The rate established by this command remains in effect until updated by a subsequent AC entry or a system reset.

SYNTAX:

ACCELERATION *axis_rate…*

AC *axis_rate*

axis_rate	The *axis_rate* argument defines an axis (X, Y, Z, or U) as well as an associated acceleration/deceleration rate for that axis (given in machine steps/msec^2).



The INDEX and FREERUN commands use this command's data for ramp up and ramp down functions.

The acceleration rate that can be used with the AC command is a maximum of 2^{15} machine steps/msec^2. The use of the decimal point is optional (e.g., AC Z25 is the same as AC Z25.).

EXAMPLES:

```
AC X1 Y1 Z2 U5      ;The acceleration rate for axis X is 1 step/msec^2
AC Z25              ;The acceleration rate for axis Z is 25 steps/msec^2
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

INDEX, FREERUN

7.8.3. AC PL (ACcel PLane)



The ACCEL PLANE command will limit the acceleration during linear and circular moves by lowering the feedrate and adjusting the ramp time. One block look ahead is used to slow down before a circle or final move in a Velocity Profiling (G8) sequence. Look ahead is done only in the MMI or with the "::" command in a custom program.

Note: This command works for contour mode 1 (CM 1) only.

Note: Setting the acceleration value to 0 will turn this function off.

SYNTAX:

ACcel PLane = a

a = maximum acceleration in units/sec/sec or steps/sec/sec/

EXAMPLE:

```
PR UN          ; assume units are millimeters
AC PL=1000     ; set acceleration at 1000 millimeters/sec/sec for current plane
```

Related Commands:

G25/G26/G27/Rounding

7.8.4. AFCO (Auto Focus)



The AFCO command enables a secondary position loop in the U500 much like the primary position loop (using Kpos). Unlike the primary position loop which uses the encoder for feedback , this loop uses the analog input for feedback. The analog input is converted to a digital signal by the optional 12 bit A/D converter. The U500 commands the motor to move so that the analog input is at the user specified voltage.

The analog input number is specified by the "channel" argument. This should be between 1 and 4, or between 1 and 8 for the U500 PCI Ultra. A zero value tells the U500 to stop the tracking mode and return to normal operation.

The "pos" argument is the desired position (set point) of the analog input in terms of volts.

The "gain" argument is a number which multiplies the A/D error to give a correction speed. This gain number is like the Kpos argument in the normal servo loop. If the scaling of the analog input and encoder is 1:1, the "gain" value can be the same as the Kpos. This is assuming that the analog transducer is properly mounted.

The "speed" parameter is the maximum speed that the motor will move when the analog input is not at the desired position. The motor may not actually move at this speed if the gain of the loop is low. The units of the speed parameter are the same as the feedrate units of F,XF,YF,ZF,UF. The units can be English / metric, units / minute, units/second, program steps / minute, or program steps / second.

SYNTAX:

"**AFCO** *axis, channel, pos, gain, speed,* [*deadband, max, min, flags*]"

Arguments ([] indicates optional parameters)**:**

where:		units
axis	XYZU	
channel	A/D converter channel 1-4, or 0 = off	
pos	target A/D position (set point)	volts
gain	sets responsiveness of loop (Like KPOS)	
speed	maximum correction speed of motor	same as feedrate "F"
[dead-band]	dead band A/D counts	
[max]	maximum motor movement in positive direction	units
[min]	minimum motor movement in negative direction	units
[flags]	see text	

Optional arguments: (default to 0)

All optional parameters default to 0.

The "dead band" argument allows the user to specify a region about the target A/D position for which there will be no motor movement. A zero value here indicates no dead band.

The "max" argument is the maximum motor movement in units, that will be allowed. This number can be interpreted as absolute or incremental depending on the setting of bit #3 of the flags argument.

The "min" argument is the minimum motor movement in units, that will be allowed. This number can be interpreted as absolute or incremental depending on the setting of bit #3 of the flags argument. Absolute values are with respect to the hardware home position ; ie. the position displayed in the diagnostic window. Incremental values are with respect to the position of the motor when the command was given. Bit #2 of the flags argument, when set, enables this feature. The units of the max and min argument are the current programming unit, ie. mm, Inches, or program steps.

The "flags" argument allows the user to change certain characteristics of the auto focus loop. The bits in the following table can be OR-ed together. Reserved bits should be programmed as zero. See Table 7-14 for a list of optional arguments.

Table 7-14. Optional Arguments

Bit #	Value	Description	Hex
0	0	Command is queued, will be affected by wait mode. Considered done when the error is within the dead-band. bit #8-11 of status word 5 will be set until finished.	0x000000
	1	command will not be queued, not affected by wait mode. bit #8-11 of status word 5 will be not be affected by command is finished.	0x000001
1	0	command will continue to track when null position is found	
	1	function will automatically turn off after null position is found	0x000002
2	0	motor travel is not limited	
	1	enable motor travel limits (max/min arguments)	0x000004
3	0	max / min arguments are with respect to home (absolute)	
	1	max / min arguments are incremental	0x000008
2..22		reserved (program as 0)	
23	0	polarity of loop is not reversed.	
	1	reverse polarity of loop.	0x800000

Motion command such as G0,G1,etc cannot be issued to the U500 when in the focus mode. The focus mode will be exited by sending an AFCO command with the channel set to 0. The software position should be updated using the SOFTWARE POSITION command when the auto focus mode is stopped. The abort function will also terminate the focus mode and automatically update the software position registers.

The range of this input is –10V to +10V which gives a digital value of –2048 to +2047 respectively for the 12 bit A/D converter.

Example #1:

”AFCO X,1,0,100,1000,1,10,-10,0x80000F”

Description: track on analog input #1, zero position with a gain of 100. Maximum correction speed is 1000 mm / min. There is one A/D count of dead-band. The encoder will move +/- 10 mm (assuming metric mode) relative to the point where to command was given. The polarity of the tracking loop is reversed. The U500 will not wait for command execution to complete. Auto focus will shut off when the axis is within 1 count of the 0 position.

Example #2:

```
AFCO Y,1,-5,100,1000,1          ; tracking to -5V analog input level
DWELL 10                        ; wait for system to settle
AFCO Y,0                        ; turn focus off
WAIT ON                         ; wait for above commands to finish
SOFTWARE  POSITION  Y           ; update encoder positions
```

Table 7-15. Analog Input Locations

Channel	U500 P1	DR500	BB501	Other Functions
1	96 (AIN1)	J13-25 (MISC.IO)	TB1-4	MFO
2	95 (AIN2)	J13-24 (MISC.IO)	TB1-3	none
3	90 (AIN3)	J12-6 (JOYSTICK)	J12-6	joystick vertical
4	89 (AIN4)	J12-3 (JOYSTICK)	J12-3	joystick horizontal

Programming Notes:

An unused channel on the U500 can be setup to display the analog input value (See Table 7-15). This is done by setting the axis parameter 38 to 45-48 for analog channel 1-4 respectively. This is useful for initial setup of the analog sensor and for polarity verification. If the polarity of the sensor is incorrect, the motor will run in wrong direction. This may result in damage to the system. On initial setup of the tracking loop, it is recommended that the max correction speed be set low and the max / min encoder movement parameters be utilized.

General parameter #99 should have bit #2 set (ie. set it to 4) to signal that the U500 should read the 12 bit A/D converter. This software feature requires V5.09 or higher and a U500 ULTRA board with the optional 12 bit A/D converter. The firmware file "U500CN.JWP" should be used. The joystick and MFO analog inputs are not available when using the 12 bit A/D converter.

The ABORT function / button will stop the auto focus mode. The axis cannot be homed when in auto focus mode.

The A/D target position can be changed by sending a new AFCO command with a different target voltage. This can be done even if a AFCO command is currently active.

It is highly recommended that a dead-band of at least one A/D count is used for applications where the user is monitoring the completion of the cycle.

7.8.5. AGAIN



The AG command is used to send the program flow to the first line of the program.

SYNTAX:

AGAIN

AG

M47

EXAMPLE:

```
OUTPUT 2,1          ;Set output bit 1
G1 X10 Y10 F100     ;Linear move of X and Y at the designated feed rate
OUTPUT 2,0          ;Clear output bit 2
AGAIN               ;Rerun program
```

C library level.

Related Commands:

QUEUE, LOOP, NEXT

ANALOG

7.8.6. Analog Vector Tracking Mode

The U500PCI can generate an output voltage proportional to the axis position or velocity. In velocity mode, the U500PCI calculates the vector velocity as the square root of the sum of the individual velocities squared. This number is then multiplied by a velocity-to-voltage conversion factor and output to the specified D/A channel. If the vector velocity exceeds the maximum velocity specified, the output voltage will remain at the maximum value.

In position mode, the analog voltage output is proportional to the vector distance moved. The distance is referenced to the location where the command ANALOG command is given.

Note that output voltage is not direction dependent. A positive move will result in the same output voltage as a negative move provided the velocity (or position) is the same.

The U500PCI calculates the vector velocity or position of all specified axes every millisecond.

The ANALOG command is a queued command and is executed in sequence with other commands given.

SYNTAX:

"ANALOG [STEPS] [*ON|OFF***] XYZU VEL/POS** *vel_pos* **DAC** *number***,** *min_voltage*, *max_voltage***"**

Table 7-16. Arguments for Analog Command

Argument	Description
STEPS	Specifies that the velocity (or position) is to be interpreted in encoder steps not current programming units. This argument is optional and must be specified before any other argument if used.
XYZU	Specifies axes to track, one to four axes may be specified here.
VEL/POS	Specifies velocity tracking mode (VEL) or position tracking mode (POS)
vel_pos	specifies the maximum velocity or position for the output voltage to equal "*max_voltage*". In velocity mode, this number is expressed with units of distance/sec. The maximum value of this number is 8,388,607 encoder counts.
DAC *number*	D/A output channel number 1-10. Min and Max voltage also required.
min_voltage	voltage at 0 velocity in velocity mode or starting position in position mode
max_voltage	voltage at maximum velocity or position
OFF	immediately turns off tracking and sets analog output to "*min_voltage*". This argument should be used by itself.
ON	restarts tracking based on previous command parameters. This argument should be used by itself.

EXAMPLES:

```
ANALOG STEPS XY VEL 10000 DAC 9,0,10
                                ; generate an output voltage ;between 0V and
                                ; 10V on D/A  channel #9 proportional to the
                                ; vector speed of X and Y axes with
                                ; maximum output voltage at 10000 encoders
                                ; counts / second.
ANALOG XYZU POS 10 DAC 1,-10,5
                                ; track vector position of all axes. Maximum
                                ; output voltage ; will occur at 10 units
                                ; (mm,in,etc.) relative to current
                                ; location of axes.
ANALOG OFF                      ; stop tracking axes and set analog output
                                ; channel to min voltage specified in above
                                ; command.
ANALOG ON                       ; restart tracking based on previous command
                                ; parameters.
```



7.8.7. AT (Autotune)

The AT command is usually called automatically by the U500 MMI or Toolkit software as an excitation signal for the autotuning function (see autotuning in Chapter 6). This command, however can be also be invoked like any other command to generate a sinusoidal excitation to a specified axis. This can be used for simple frequency response calculations.

Note that this command also specifies the data collection parameters for the automatic autotuning function. The U500 software contains the algorithms that analyze this data and fit gains to the servo system. When the command is invoked alone, these data are ignored. The command WILL NOT directly calculate servo loop gains.

SYNTAX

AT *axis,startfreq,amplitude,cycles*,[[numfreq]],[[samptime]],[[numsamp]]

startfreq	Frequency of oscillation (.1-100).
amplitude	Peak-peak position displacement in units.
cycles	Number of cycles to generate.
[[numfreq]]*	Number of multiples of "startfreq" to generate (frequency is doubled and the position displacement is halved.)
[[samptime]]*	Sample time in ms (reserved for MMI software use only)
[[numsamp]]*	Number of samples to collect (reserved for MMI software use only.)

* = parameter not required

EXAMPLE:

```
AT 1,1.2,100,10      ;This command generates 10 cycles of a sinusoidal
                     ;velocity command to axis 1 with a maximum
                     ;displacement of 100 mm and a frequency of 1.2 Hz
```

7.8.8. BOARD



The BO command is used to select the UNIDEX 500 PC board that is to receive the forthcoming commands. This is only applicable to configurations with more than one U500 board installed. The board selection remains in effect until the next board command is issued. Only one board may be selected at a time.



If you are writing your own program in "C" or Visual Basic and have to initialize more than one board, then execute the following sequence:

1. Send the BOARD 1 command
2. Initialize board no. 1
3. Send the BOARD 2 command
4. Initialize board no. 2
5. Continue sequence if using more than two U500 boards

SYNTAX:

BOARD *number*

BO *number*

number Designates the UNIDEX 500 board number (1, 2, 3, 4, 5, or 6) to receive the commands.

EXAMPLE:

```
BOARD 1          ;or
BO 1             ;Selects board number 1 to receive commands
```

Library Level. Executed by the host processor.



The U500 cannot generate a precise contour type move between separate boards.



7.8.9. BRAKE

The BR command manually engages or disengages the UNIDEX 500 brake output. A braking relay and power supply are available options for the DR500 drive rack. This command can be used to manually turn the brake on for an axis just before disabling it. This may be useful in vertical axis applications when axis movement cannot be tolerated. The U500 does not immediately release the brake on a linked axis when enabled. There are approximately 100 ms before the brake is released.

SYNTAX:

BRAKE *state*

BR *state*

state The *state* argument can be set to one of two options:
ON.....Engages the brake
OFF .....Disengages the brake

EXAMPLE:

```
BRAKE ON        ; Optional brake is now engaged
BR ON           ; Same result using the abbreviated syntax
BRAKE OFF       ; Optional brake is now disengaged
BR OFF          ; Same result using the abbreviated syntax
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

ENABLE AXIS, DISABLE AXIS, Fault Mask Parameters, Enable Brake Mask Parameter



The brake output must be *linked* to only one axis with parameter x61 (*Enable brake mask*). The brake output is deactivated (unclamped) when the axis is enabled, and activated (clamped) when the axis is disabled. This command only works when the axis is enabled.

7.8.10. CAL (Load Calibration File)



The CAL command downloads axis calibration data from a file. This can be used to dynamically down load calibration files during a user program. Calibration data does not become active until the axis is homed.

SYNTAX:

CAL *"filename"*

"filename" The file containing the axis calibration data.

EXAMPLE:

CAL "C:\U500\MMI\scan1.cal"

Programming level: Interpreter.

Related Commands: (none)



7.8.11. CALLDLL

The CALLDLL function can be used from an MMI program to call any DLL. This command can be used to call any existing DLL, or can be used to call a function to perform a task that can not be accomplished from within a U500 program.

SYNTAX:

v# = CALLDLL *"dllname", "function", "string"*

v#	U500 user variable (V0 - V255), function return value.
dllname	Name of user DLL to be called.
function	Function in the user DLL to be called.
string	Optional string to pass to DLL function.

USAGE:

This command is used to call any user DLL. The prototype of the function that can be called from a U500 program is:

double FuncName (double *varlist, char *str)

The function will return a double into the user variable specified in the CALLDLL command. The argument pointer to a double is the location of U500 user variable V0. The U500 user variables are stored sequentially as doubles. Thus, any user variable will be available to the called DLL function. The string passed to the DLL is optional and can be used if the function being called requires a string. Unless this prototype is the same for the DLL function to be called, a wrapper function and DLL will need to be created to call the desired function.

The CALLDLL function loads the DLL, performs the procedure or function call, and then unloads the DLL from memory. This may not be desirable in cases where the DLL requires a significant amount of time to load or where variables are stored in the DLL and need to be accessed by future calls into the DLL. For these cases, two commands have been added to manually load (LOADDLL) and unload (FREEDLL) the DLL. The sequence of commands would be to call LOADDLL to load the DLL into memory, make all necessary function calls using CALLDLL, then call FREEDLL to unload the DLL from memory.

SYNTAX:

v# = **LOADDLL** *"dllname"*

v#	(v0-v255) returns the handle of the dll
"dllname"	Name of the DLL to load into memory.

SYNTAX:

v#1 = **FREEDLL v#2**

v#1	the return code, 0 is successful
v#2	Handle of the dll that is loaded in memory (returned by LOADDLL call)

EXAMPLE:

This is an example showing how to call the U500 DLL, WIN50032.DLL, using the CALLDLL command. The following is the C code of the DLL for three different functions; the DLL created is named U500USER.DLL.

```
#include <math.h>
#include "c:\u500\ntlib\qlib\aerotype.h"
#include "c:\u500\ntlib\qlib\build.h"
#include "c:\u500\ntlib\wapi\wapi.h"

double __declspec(dllexport) _stdcall ReadPos75( double *pUserVar, char *str )
{
/* This function stores the real-time position of the X axis in V75. */
  *(pUserVar+75) = WAPIAerReadPosition(8);
  return(0);
}

double __declspec(dllexport) _stdcall SendCmd( double *pUserVar, char *str )
{
/* This function sends a command to the U500. */
  return( WAPIAerSend(str) );
}

double __declspec(dllexport) _stdcall SolveQuadratic( double *pUserVar, char *str )
{
/* This function calculates the roots of the quadratic equation.  The a, b, and c
    terms are in V10, V11, and V12, respectively.  The roots will be calculated and
    returned in V0 and V1.   */
  double a=*(pUserVar+10);
  double b=*(pUserVar+11);
  double c=*(pUserVar+12);

  if ( b*b-4*a*c<0 )  //imaginary result
    return(-1);

  *pUserVar = (-b+sqrt(b*b-4*a*c))/(2*a);
  *(pUserVar+1) = (-b-sqrt(b*b-4*a*c))/(2*a);
  return(0);
}
```

To build a DLL that can be called externally requires a DEF file that lists the functions used. The following is the associated DEF file:

```
LIBRARY u500user

EXPORTS
        ReadPos75
        SendCmd
        SolveQuadratic
```

The following is the U500 program, which calls the different functions in the U500USER.DLL:

```
v5=calldll "u500user.dll","ReadPos75"
me di "X axis position = %v75"

v1=calldll "u500user.dll","SendCmd","di x"
me di “WAPIAerSend return code = %v1”

v10=1
v11=4
v12=3
v5=calldll "u500user.dll","SolveQuadratic"
me di "ret=%v5,+root=%v0,-root=%v1"
```

The function ReadPos75 calls the WAPIAerReadPosition function within the WIN50032.DLL. The return value of the function, the real-time position of the X axis, is returned into user variable V75. The SendCmd function passes the string, “di x,” to the WAPIAerSend function of WIN50032.DLL. The return value of the WAPIAerSend function will be passed back to the U500 program into user variable V1. The other function, SolveQuadratic, is a basic function performing the operation of finding the roots of a quadratic equation. The coefficients, a, b, and c, of the quadratic equation are stored in V10, V11, and V12, respectively. If imaginary roots will result, the function will return a -1 into V5, otherwise, the equation is solved and the roots will be stored in V0 and V1.

Related Commands:

FREEDLL, LOADDLL

7.8.12. CLOCKWISE and COUNTERCLOCKWISE CIRCULAR INTERPOLATION



The clockwise (CW) or counterclockwise (CCW) circle commands initiate circular contour-type motion (i.e., circles or arcs). The axis pair assigned to the circular motion automatically adjusts its path and feed rate to maintain a circular contour path.

The UNIDEX 500 uses the contour plane's ramp time to ramp both of the axes up to steady speed and then down to the target distance (the ramp can be linear or inverse sine-type).

SYNTAX:

For clockwise rotation:

CW_CIRCLE Axis*end1* **Axis***end2* **I***c1* **J***c2*

G2 Axis*end1* **Axis***end2* **I***c1* **J***c2*{**F***feedrate*|**FT***feedtime*}

G2 Axis*end1* **Axis***end2* **C***c1,c2*{**F***feedrate*|**FT***feedtime*}

G2 Axis*end1* **Axis***end2* **R***rad* {**F***feedrate*|**FT***feedtime*}

For counterclockwise rotation:

CCW_CIRCLE Axis*end1* **Axis***end2* **I***c1* **J***c2* {**F***feedrate* | **FT***feedtime*}

G3 Axis*end1* **Axis***end2* **I***c1* **J***c2*{**F***feedrate*|**FT***feedtime*}

G3 Axis*end1* **Axis***end2* **C***c1,c2*{**F***feedrate*|**FT***feedtime*}

G3 Axis*end1* **Axis***end2* **R***rad* {**F***feedrate*|**FT***feedtime*}

Axis	X, Y, Z, or U
end1	Defines the first axis (X, Y, U, or Z) that is involved in motion, and the first end point (or present point).
end2	Defines the second axis (X, Y, Z, or U) involved in motion and the second end point (or present point).
c1,c2	Defines the center point of each arc. Always incremental with respect to the starting point.
rad	A radius can be specified instead of discrete center points. By specifying endpoints and a radius, there are two different arcs that can be traced. If the radius is positive, the shorter of the 2 arcs are generated. Specifying the radius as negative will generate the longer arc.

The center point is always specified incrementally from the starting point of the circle, regardless of the programmed mode.





Complete circles can not be generated when using the **R** syntax.

feedrate Upon initializing the U500, the default units for feedrate are in units/min. The units are defined by the English and metric conversion factors. The units for feedrate can be changed by the PROGRAM command. The maximum feedrate that can be used for a contour type move is 2^{15} machine steps/msec. If a feedrate is specified which is different than the current feedrate, the vector velocity will ramp to the new feedrate within the current ramp time.



If a feedrate subcommand is missing, the UNIDEX 500 will use a previously programmed feedrate or the feedrate established by general parameter numbers 022, 040, 058, and 076.

feedtime Defines the time (in seconds) that is allocated to complete the contour move. The UNIDEX 500 calculates the contour feedrate based on the contour path. The feedtime is valid for the current block only.

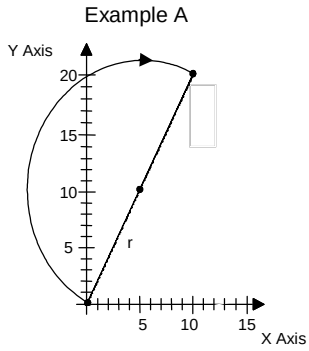
The contour feedrate is clamped at the feedrate established by general parameters 027, 045, 063, 081 - *Clamp feedrate (prog. steps/ms).*

The corner rounding, noncorner rounding, and velocity profile programming options may be used in conjunction with these circular moves. See the ROUNDING (G23 and G24), and VELOCITY (G8 and G9) command descriptions in this chapter.

Circles and Arcs

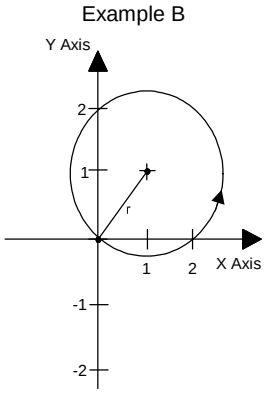
EXAMPLES:

The following commands can be used to create a CW arc ending at point (10,20) with a center point of (5,10) incrementally away from the starting point. The previously set feedrate is assumed by its absence (Example A).




```
CW_CIRCLE  X10 Y20 I5 J10   ;end point X=10, Y=20, radius =√125; or
CW  X10 Y20 I5 J10          ;end point X=10, Y=20, radius =√125; or
G2  X10 Y20 I5 J10          ;end point X=10, Y=20, radius =√125
```

Each of the following commands can be used to create one full CCW circle with a radius of √2 (Example B).

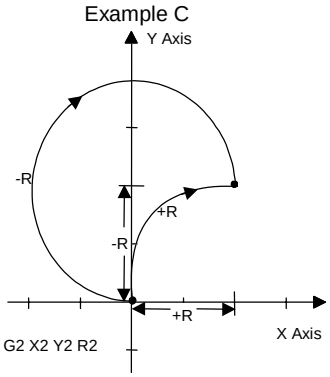



```
CCW_CIRCLE  X0 Y0 I1 J1     ;full circle with radius=√2; or
CCW  X0 Y0 I1 J1            ;full circle with radius=√2; or
G3 X0 Y0 I1 J1              ;full circle with radius=√2
```

The following command can be used to create one full CW circle with a radius of √2 and a center point of (1,1), assuming incremental mode (Example C).

```
CW  X2 Y2 R2                ;shorter path taken; or
G2  X2 Y2 R-2               ;longer path taken
```

Any one of the following commands can be used to create CW circular motion with end points and center points defined in variables. The previously set feedrate applies.




```
CW_CIRCLE X=V1 Y=V2 C=V3,V4 ;or
CW X=V1 Y=V2 C=V3,V4        ;or
G2 X=V0+10 Y=V1+2 I=V3 J=V4 ;variables used to specify parameters
```

Radius Adjustment

When using the I, J or C method of specifying a circle, it is possible that the user specify a combination of end point and center point which does not give equal radii at the start and end of the circle. In this case, the U500 will calculate a new radius so that a smooth arc will be generated and the programmed end point is preserved.

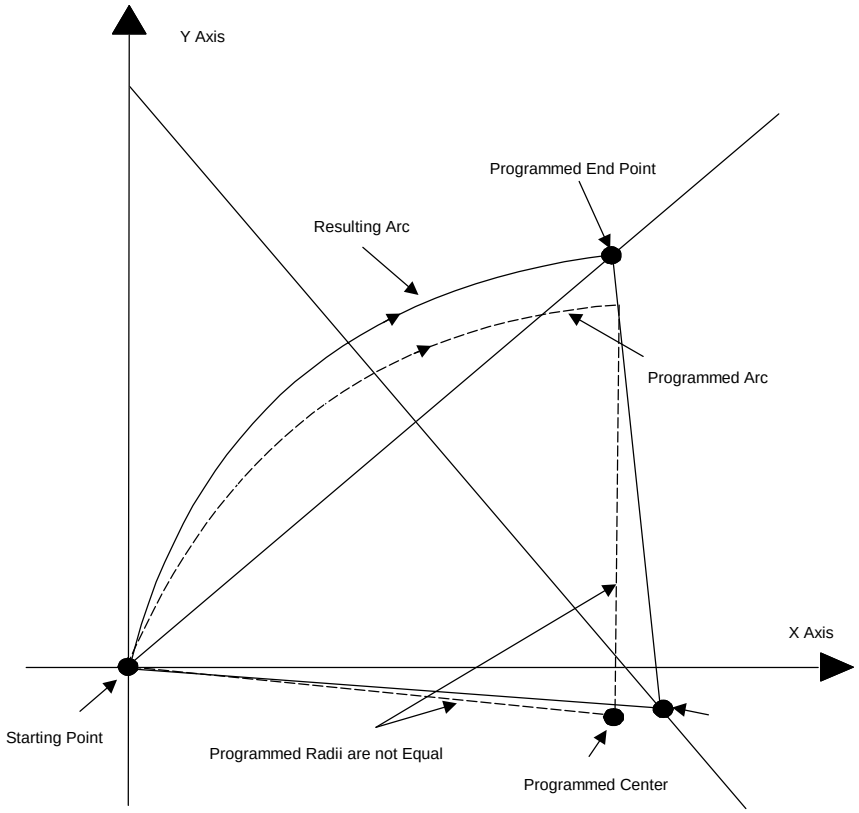


Figure 7-1. Circle Center Adjustment

Helix Motions (Circular plus Linear)

Helix motion incorporates linear motion and circular motion. The syntax for clockwise and counterclockwise helix motion is similar to regular circular motion, with the addition of the **LINEAR** term and its associated arguments. Variations of the command syntax for helix motions are shown below.

SYNTAX:

CW_CIRCLE Axis*end1***Axis***end2* **I**1 **J**1 {**F***feedrate* | **FT***feedtime*}
LINEAR Axis{**F***feedrate* | **FT***feedtime*}

CW Axis*end1* **Axis***end2* **I**1 **J**1 {**F***feedrate* | **FT***feedtime*}
LINEAR Axis{**F***feedrate* | **FT***feedtime*}

CW Axis*end1* **Axis***end2* **I**1 **J**1 {**F***feedrate* | **FT***feedtime*}
LI Axis{**F***feedrate* | **FT***feedtime*}

G2 Axis*end1* **Axis***end2* **I**1 **J**1 {**F***feedrate* | **FT***feedtime*}
G1 Axis{**F***feedrate* | **FT***feedtime*}

EXAMPLES:
In the following example, axes Z and U do clockwise circular motion. A new contour feedrate of 100 is specified. Axes X and Y do linear motion.

CW_CIRCLE Z20 U20 I1 J1 F100 LINEAR X10 Y20

CW Z20 U20 I1 J1 F100 LI X10 Y20

CW Z20 U20 I1 J1 F100 G1 X10 Y20

This example shows a helical path beginning with linear move of the X axis, CCW circular move of the Y and Z axes to endpoint of 5,5. Entire move to take 10.5 seconds.

G3 Y10 Z10 I5 J5 FT10.5 G1 X10

Dual Circular Motions (spherical motion)

Dual circular motion commands use the combined syntax of two circular motion commands. Variations are shown below.

SYNTAX:

CW_CIRCLE Axis*end1* **Axis***end2* **I***c1* **J***c2* {**F***feedrate* | **FT***feedtime*}
CW_CIRCLE Axis*end1* **Axis***end2* **I***c1* **J***c2* {**F***feedrate* | **FT***feedtime*}

CW Axis*end1* **Axis***end2* **I***c1* **J***c2*{**F***feedrate* | **FT***feedtime*}
CW Axis*end1* **Axis***end2* **I***c1* **J***c2*{**F***feedrate* | **FT***feedtime*}

G2 Axis*end1* **Axis***end2* **I***c1* **J***c2* {**F***feedrate* | **FT***feedtime*}
G2 Axis*end1* **Axis***end2* **I***c1* **J***c2*{**F***feedrate* | **FT***feedtime*}

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

LINEAR, RAMP, ROUNDING, VELOCITY, TRAJECTORY, PROGRAM, G70/G71, G90/G91, G23, G24, CM

This command is available only on Plus and Ultra models of the UNIDEX 500.





7.8.13. CLRSCR (Clear Screen)

The CLRSCR command clears the message display window.

SYNTAX:

CLRSCR

There are no arguments needed with the CLRSCR command.



7.8.14. CM (Contouring Mode)

The contouring mode command (CM) is used to set the default contouring mode (CM 0) or enhanced contouring mode (CM1). During velocity profiling, normal contouring mode blends moves together by combining the deceleration of one move with the acceleration of the next move. The enhanced mode does not and requires that the last move be preceded by a G9 command (velocity profiling off) if in G8 mode.

For profiles with velocity profiling off (G9 mode), CM0 and CM1 function identically.

The Filter Time Constant command (FL) can be used with enhanced contouring mode to produce corner rounding effects or to smooth transitions between non-tangential moves.

The default contouring mode can also be set by general parameters 31, 49, 67, and 85. The CM command is modal and will remain in effect until reset.

CM 1 mode is recommended for profiles that consist of many short moves or moves with non-tangential vectors. See the VELOCITY command for more information. The maximum allowed MFO value in CM1 is 100%.

SYNTAX:

CM #

#	0 for normal contouring mode. 1 for enhanced contouring mode.

EXAMPLE:

```
CM 1                    ;set new contouring mode
```

Programming level: Interpreter.

Related Commands:
FL, VE, G8, G9

7.8.15. COM FREE

The COM FREE function disables the COM port in the software interface. It frees the COM port and allows other applications to use it.



SYNTAX:

COM FREE

No arguments are needed for this command.

Refer to Chapter 9 “Sample Programs” for an example of the U500 Rs232 COM functions.



The COM functions are supported only by the MMI software of the UNIDEX 500. These functions can be run from parts programs that are run under MMI.



Unlike standard program commands, COM functions cannot be abbreviated using the first few letters.



7.8.16. COM GET

The COM GET function is used to define one or more system variables (v0 through v255) that will hold numerical data that is received (one character at a time) by the COM port specified in the COM INIT function. Numerical characters (that is, a subset of characters from the set {+, -, 0-9, and .} are read from the COM port until the terminate sequence (defined in COM SET _TERMINATE_) is encountered. Once the termination sequence is read, the individual numerals are converted into a number and the resulting number is stored in the specified variable.



SYNTAX:

COM GET *var1*, *var2*, ..., *varN*

var#	A variable number (v0 through v255) that will hold the numerical value that was received from the COM port.

If more than one number string is to be received from the COM port, the termination string must separate the numbers. For example, if you are expecting three numerical values, you might use the syntax COM GET v100, v101, v102. The information received by the COM port might look like the following.

- 1 0 0 . 2 3 <termination string> 5 . 7 <termination string> 6 . 0 4 <termination string>



Refer to Chapter 9 “Sample Programs” for an example of the U500 Rs232 COM functions.



The number is received from the most significant digit to the least significant digit. For example, the number -100.23 is sent as - 1 0 0 . 2 3 <termination string>.



If an invalid character is received, the current string of numerals being read from the COM port is considered to be bogus. In this case, the string (containing bogus data) not converted into a number, the variable defined in the COM SET _STATUS_ function is set to error code 5 (invalid character received) and the corresponding variable defined in the COM GET function is left untouched. Valid characters for the GET function are the numerals 0-9 (any number of these numerals in any order), a single plus (+) or minus (-) character per number, and a single decimal point (.) per number. All others (excluding the termination sequence) are considered to be invalid.



The COM GET function is used when numerical data is expected. To receive alpha-numeric data (i.e., numbers, letters, special characters, etc.), use the COM RECEIVE function.



The COM functions are supported only by the MMI software of the UNIDEX 500. These functions can be run from parts programs that are run under MMI.



Unlike standard program commands, COM functions cannot be abbreviated using the first few letters.

7.8.17. COM INIT



The COM INIT function is used to initialize a selected COM port of the PC to specified parameters. The number of data bits, stop bits, the type of parity and the COM port baud rate are all set through this command.

SYNTAX:

COM INIT *nPortNum*, *nDataBits*, *nStopBits*, *cParity*, *nBaud*

nPortNum	The COM port number to be initialized (1-9).
nDataBits	The number of data bits (7 or 8).
nStopBits	The number of stop bits (1 or 2).
cParity	The desired parity (N, E or O): N = None E = Even O = Odd
nBaud	The desired baud rate (110, 150, 300, 600, 1200, 2400, 4800 or 9600).

Refer to Chapter 9 “Sample Programs” for an example of the U500 Rs232 COM functions.



7.8.18. COM RECEIVE



The COM RECEIVE function is used to write all data that is received from the COM port (specified in the COM INIT function) to the buffer specified by the COM SET _RECEIVE_ function. Information is read from the COM port until the terminate sequence (defined in COM SET _TERMINATE_) is encountered. Once the termination sequence is read, the file specified by the COM SET _RECEIVE_ function may be read and interpreted as needed.

SYNTAX:

COM RECEIVE

No arguments are needed for this command.

Refer to Chapter 9 “Sample Programs” for an example of the U500 Rs232 COM functions.





7.8.19. COM SEND

The COM SEND function is used to send information to the COM port that was specified in the COM INIT function. This function can send a text string, terminating characters, value of a user variable (v0 - v255) or single characters specified by their ascii value.

SYNTAX:

COM SEND "string"

string = Double quote(") delimited ASCII text (i.e. "Hello")

The string may contain the following escape sequence:

/v{format specifier}[0..255]	insert value in variable 0..255
/t	insert terminate sequence
/c[Number]	insert value for character
//	send the / character

The format specifier is used to specify the format of the variable being accessed, it has the following format:

(%[type] {.precision})

where,

type - L (l) for long integer format, f for floating point

precision - the number of decimal places (default 6)

EXAMPLE:

The following example sends the given values to the com port that was specified with COM INIT.

```
v200=56.453
com send "Var200=/v200/t"                ;sends -> Var200=56.453000[_terminate_]
com send "Var200=/v(%f.3)200/t"          ;sends -> Var200=56.453[_terminate_]
com send "Var200=/v(%f.0)200/t"          ;sends -> Var200=56.[_terminate_]
com send "Var200=/v(%l)200/t"            ;sends -> Var200=56[_terminate_]
com send "/c52 /c53 /c0x38/t"            ;sends -> 4 5 8 [_terminate_]
```



Refer to Chapter 9 "Sample Programs" for an example of the U500 Rs232 COM functions.

7.8.20. COM SET _RECEIVE_



The COM SET _RECEIVE_ function is used to set the receive buffer to be a given file name. Two arguments are used with this function. The first argument specifies the target file name. The second argument specifies whether the text is to be written to a new file or appended to an existing file.

SYNTAX:

COM SET _RECEIVE_ *filename, cFileMode*

filename	The name of the target file where text is to be written
cFileMode	The mode switch that defines how the file is to opened:
	n = open a new file
	If the file already exists, its contents are overwritten.
	a = append to a file
	If the file already exists, the characters are appended to the file.

If the file specified in the filename argument does not exist, then it will be created automatically.

If the file specified in the filename argument already exists and the “open new file” (n) argument is selected, then the contents of the existing file will be overwritten.

If the file specified in the filename argument already exists and the “append to a file” (a) argument is selected, then the text will be appended to the existing file.

Refer to Chapter 9 “Sample Programs” for an example of the U500 Rs232 COM functions.

Be sure to observe the space characters that surround _RECEIVE_ (i.e., use the syntax COM SET _RECEIVE_ filename, cFileMode, not the improper syntax COM SET_RECEIVE_filename, cFileMode.



7.8.21. COM SET _STATUS_

The COM SET _STATUS_ function is used to define a system variable (v0 through v255) where subsequent COM function status information is to be stored. After any COM function is executed, the result status of the COM function is stored in the specified system variable.

SYNTAX:

COM SET _STATUS_ V#

= the variable number (v0 through v255) where status information is to be stored.

After any COM is executed, the appropriate system variable (V0 through v255, as specified in the variable argument) can be checked (read) to determine if an error has occurred. The error code found in the system variable can have one of five possible values with the following meanings:

0 = no error has occurred (INIT, GET, RECEIVE and SEND commands)

1 = a time out error has occurred (GET and RECEIVE commands)

2 = a framing error has occurred (GET and RECEIVE commands)

3 = a parity error has occurred (GET and RECEIVE commands)

4 = an overrun error has occurred (GET and RECEIVE commands)

5 = an invalid character was recvieved (GET)



Refer to Chapter 9 “Sample Programs” for an example of the U500 Rs232 COM functions.



Be sure to observe the space characters that surround _STATUS_ (i.e., use the syntax COM SET _STATUS_ variable, not the improper syntax COM SET_STATUS_variable.

7.8.22. COM SET _TERMINATE_



The COM SET _TERMINATE_ function is used to define a termination string (of ASCII values) that is used between devices that are communicating over a COM port. The arguments for this function consist of one or more ASCII values separated by commas. Instead of an ASCII value, each argument of this function can consist of a system variable (v0 through v255) that contains an ASCII value.

SYNTAX:

COM SET _TERMINATE_ val1, val2, ..., valN

val1 = the 1st value of a termination string consisting of an ASCII character value (for example, 13) or a variable (v0 through v255) that contains an ASCII value

val2 = the 2nd value of a termination string consisting of an ASCII character value (for example, 13) or a variable (v0 through v255) that contains an ASCII value

valN = the last value of a termination string consisting of an ASCII character value (for example, 13) or a variable (v0 through v255) that contains an ASCII value

If an argument uses a numerical value, the MMI software will assume that the number is in decimal format (e.g., 13), unless it is preceded by 0x (e.g., 0x0D), in which case it is assumed to be in hexadecimal format.

Refer to Chapter 9 “Sample Programs” for an example of the U500 Rs232 COM functions.



Be sure to observe the space characters that surround _TERMINATE_ (i.e., use the syntax COM SET _ TERMINATE _ val1, val2, ..., valN, not the improper syntax COM SET_ TERMINATE_val1, val2, ..., valN.





7.8.23. COM SET _TIMEOUT_

The COM SET _TIMEOUT_ function is used to define time-out values for the COM GET and COM RECEIVE commands before a time-out error occurs. This function has two arguments that specify the number of tries to attempt to GET/RECEIVE information, and the delay time (given in milliseconds) between each attempt.

SYNTAX:

COM SET _TIMEOUT_ nTries, nDelay

nTries = the number of tries to attempt to GET/RECEIVE information

nDelay = the delay time in ms between each attempt.



Refer to Chapter 9 “Sample Programs” for an example of the U500 Rs232 COM functions.



Be sure to observe the space characters that surround _ TERMINATE _ (i.e., use the syntax COM SET _TERMINATE_ nTries, nDelay, not the improper syntax COM SET_ TERMINATE_nTries, nDelay.

7.8.24. CS (Command Scope)



The Command Scope (CS) works identically to calling the aer_scope_command function from the libraries. It can be used to collect actual velocity, command velocity, or torque information from the U500 DSP. See WAPIAerScopeCommand in the Programming Tools section.

SYNTAX:

v#=CS *####*

#	Variable number.
####	Defines the cmd argument to the aer_scope_command. It returns the return code of the aer_scope_command into the specified variable.

EXAMPLES:

```
V0=CS 0x40000+1          ;sets time base to 1 ms
V0=CS 0xb0000+7500       ;sets "long" number of samples
V0=CS 0xc0000            ;collect data
:wait
V0=CS 0xd0000            ;return number of points to collect
if v0>0 :wait
```

Programming level: Interpreter

Related Commands: (none)



7.8.25. Cutter Compensation Commands

Cutter compensation offsets programmed moves to compensate for the size of the cutting tool. The UNIDEX 500 implements cutter compensation with the following G code commands:

G40 - cutter comp. off
G41 - cutter comp. on, LEFT
G42 - cutter comp. on, RIGHT
G43 - define cutter radius
G44 - define compensated axes.

Cutter compensation operates only on contoured motions, G1, G2, and G3. It modifies the endpoint of a move based on the next move. Therefore, a move will not begin executing until the next move has been commanded. Commands that occur between contours are stored until the next contoured motion is sent. No more than 5 commands should be placed between contours.

Contoured moves are modified either by making the move shorter or by adding a circle with the same radius as the cutter, between the moves.

The first move is assumed to be a move on to the part. This move is the first move after a G41 or G42 command and can be linear or circular. The end point of this move is adjusted so that it is normal to the second move's starting point, offset by the tool radius. Refer to Figure 7-2.

The last move is assumed to be a move off of the part. This contour occurs after the G40 command. The end point of the move preceding the G40 command (last move on work piece) is adjusted to be normal to the move's programmed endpoint. Refer to Figure 7-3.

In the case of a circular move on to the work piece, the programmed end angle is preserved. For a move off of the part, the circle's start angle is preserved.



Abort and reset clears cutter compensation. Any buffered commands are lost.

SYNTAX:

G40	Turns off cutter radius compensation. The contour following this command moves off of the work piece.
G41	Turns on cutter radius compensation LEFT. A tool radius and the axes to be compensated must first be specified.
G42	Turns on cutter radius compensation RIGHT. A tool radius and the axes to be compensated must first be specified (default).
G43 R*units*	Set cutter radius.
G44 *axis axis*	Defines axes for compensation: X, Y, Z, or U.
units	Cutter radius. Units are the same as current programming mode: English units, metric units, or steps.
axis	Defines axes for compensation: X, Y, Z, or U.

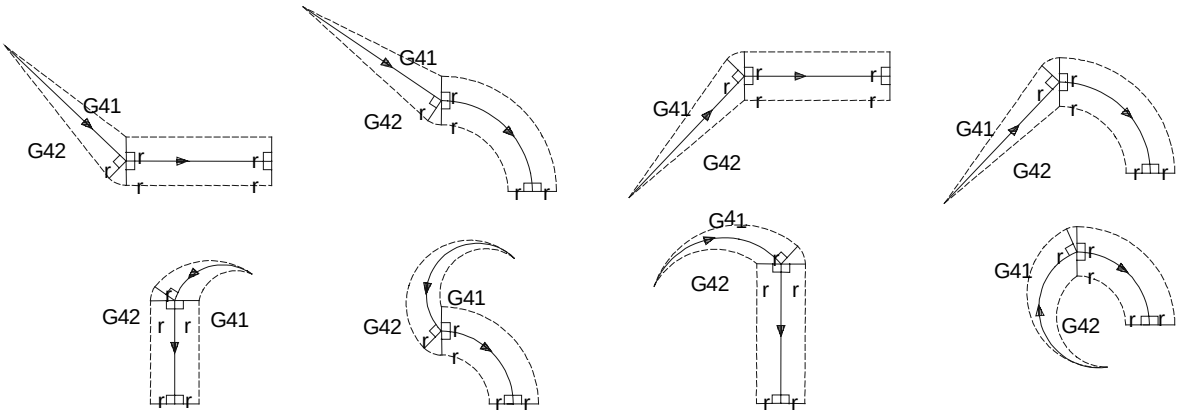


Figure 7-2. Startup Moves

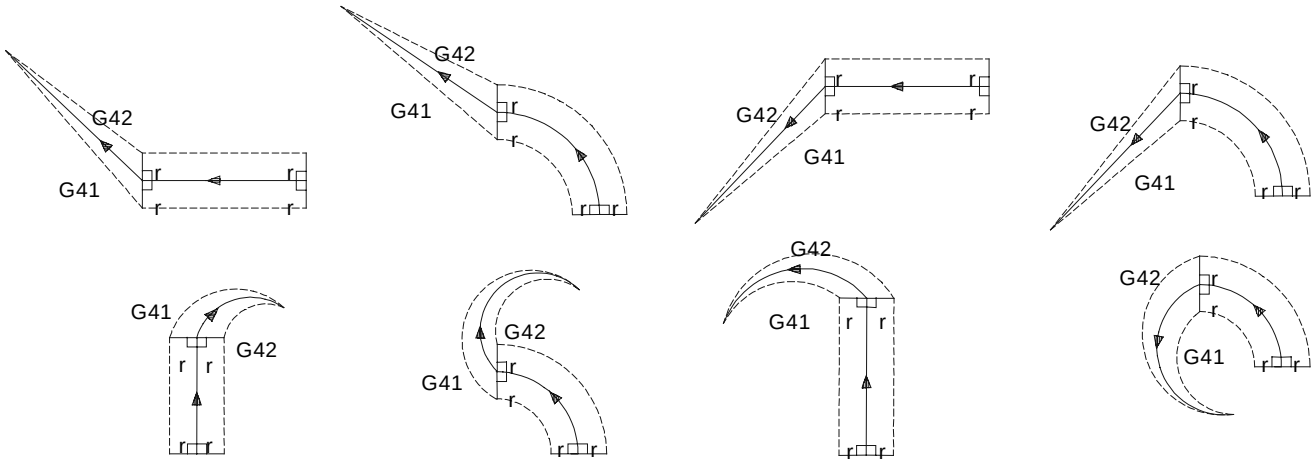


Figure 7-3. Ending Moves

EXAMPLE:

The following example program demonstrates the cutter compensation commands. See Figure 7-4.

```
; ##### CUTCOMP1.PRG #####
;
; - demonstrates cutter compensation while doing a square
;
ENABLE X Y              ; Enable axes
HOME X Y                ; Home axes
PROGRAM UNITS UNITS/MIN     ; Set to units (default)
G91                     ; Set Incremental mode
G70                     ; Set English mode
Ramp 100

; ##### 1st, do square with no cutter comp ####
G1 X1 Y1 F10            ; Move on to part
G1 Y1                   ; Side
G1 X1                   ; Side
G1 Y-1                  ; Side
G1 X-1                  ; Side
G1 X-1 Y-1              ; Move off
G4 500                  ; Dwell 1/2 sec

; ##### 2nd, do square with cutter comp ####
G43 R.25                ; Set tool radius to .25"
G44 X Y                 ; Define cutter comp axes
G41                     ; Cutter comp left
G1 X1 Y1 F10            ; Move on to part
G8 G1 Y1                ; Side
G1 X1                   ; Side
G1 Y-1                  ; Side
G9 G1 X-1               ; Side
G40                     ; Cutter comp off
G1 X-1 Y-1              ; Move off
```



Cutter compensation is available only on the UNIDEX 500 ULTRA.

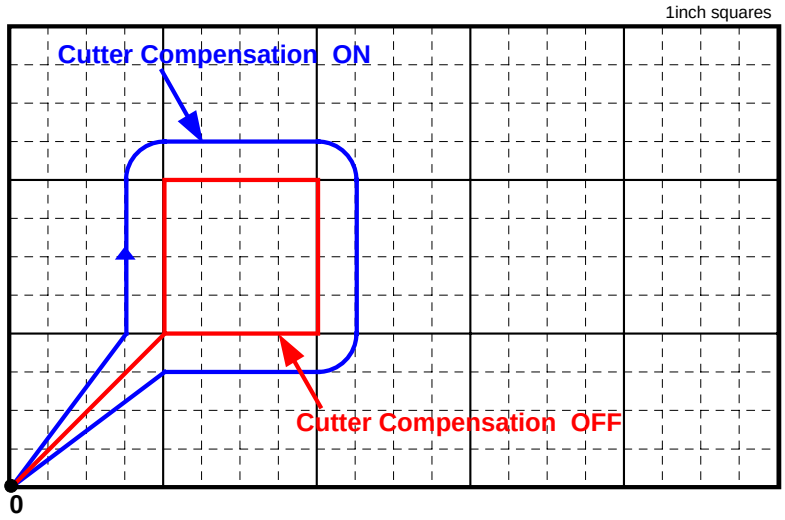


Figure 7-4. Cutter Compensation Example

7.8.26. CVI (Convert to Integer)



The CVI command is used to convert a given value to an integer. The fractional portion of the argument is truncated.

SYNTAX:

v#=**CVI**(*expression*)

v#	Any user variable (v0 - v255).
expression	Any mathematical expression.

EXAMPLE:

```
v10=CVI(3.97)          ;result v10 = 3
v15=CVI(v15*3.24)      ;result v15 = int value of v15*3.24
```

C library level.

This command is only available with the MMI software interface.



7.8.27. CYCLE

CY

The CYCLE ON command is used to map an input bit to the cycle start function. While a program is running, the MMI scans the input bit. If the input bit value equals the bitstate specified in the command, the cycle start function is called. This command is identical to the "CYCLE START" button in the Load Program window. Refer to the example in Figure 7-5. CYCLE OFF stops the scanning of the input bit. This command also can be used with the SBX-IO48 board.

SYNTAX:

CYCLE ON, *inputbit,bitstate*

CYCLE OFF

CY ON, *inputbit,bitstate*

CY OFF

inputbit	Input bit number (0 to 15) or valid SBX-IO48 input bit ($000 to $127).
bitstate	Bit value to send cycle start, either 1 or 0.

EXAMPLE:

```
CYCLE ON, 4,1        ;Checks input bit 4 for a logical 1. If the value is present,
                     ;calls cycle start
```

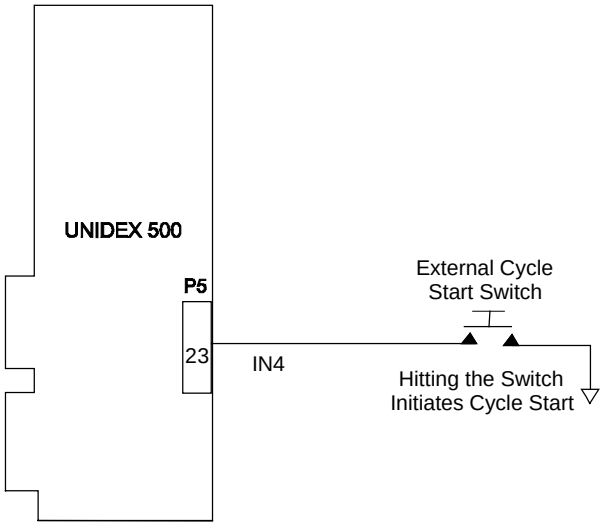


Figure 7-5. CYCLE START Function

This command is only available with the MMI software interface.

Related Commands:
QUEUE INPUT

7.8.28. DAC (D/A Output)



Digital to analog conversion (DAC) channels are normally used for axis servo loop current commands. DAC channels 5-8 are normally used as the second current command phase when using AC servo or stepper motors. Unused DAC channels may be used as analog outputs and have ranges of +10 VDC to -10 VDC.

The U500 PCI has two additional DAC outputs located on the secondary axis interface connector (P9). The are both ± 10 VDC analog outputs. The connector for channel 9 is P9-41 and the connector for channel 10 is P9-42.

SYNTAX:

DAC *number, volts*

DA *number, volts*

number	Output number (1 2 3 4 5 6 7 8).
volts	Voltage output (-10 to 10 volts).

The DAC command should not be issued to a channel that is being used for servo or stepping motor operation.



The following channel-signal relationship exists:

DA Channel	Signal	U500 ISA Test Point
1	ICMD1B	TP16
5	ICMD1A	TP20
2	ICMD2B	TP17
6	ICMD2A	TP21
3	ICMD3B	TP18
7	ICMD3A	TP22
4	ICMD4B	TP19
8	ICMD4A	TP23

EXAMPLE:

```
DA 1,2.5          ;Sets DAC #1 to 2.5 volts
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.



7.8.29. DISABLE

The DISABLE command is used to disable one or more axes. When axes are disabled, the U500 continues to track the position, but the current command output remains at zero, so motion is stopped. The amplifier(s) is also disabled. The command is also used to disengage the reading of the A/D registers.

SYNTAX:

DISABLE *axis*

DI *axis*

DI AD

axis	Defines the axes (X, Y, Z, or U) to be disabled.

EXAMPLE:

```
DISABLE Y Z          ;or
DI Y Z               ;Motors for axes Y and Z are disabled

ENABLE X Y AD        ;engage the reading of the A/D registers,
                     ;enable X, Y axes as well
V0=$AD0              ;read the value at A/D channel #0
V1=$AD1              ;read the value at A/D channel #1
V2=$AD2              ;read the value at A/D channel #2
V3=$AD3              ;read the value at A/D channel #3
DISABLE AD           ;disengage the reading of the A/D channels
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.



An axis should not be enabled until the initial system checks (described in Chapter 3) have been completed.

Related Commands:
ENABLE

7.8.30. DS (Display Servo Loop Data)



Digital-to-analog conversion (DAC) channels are normally used for axis servo loop current commands. DAC channels 5-8 are normally used by the servo processor as the second current command phase when using AC servo or stepper motors. The DS command provides real-time servo loop display through unused DAC channels (using, for example, a scope connected to the appropriate test points on the U500 board). This can be useful when tuning servo loops since it provides dynamic, real-time feedback (as opposed to the tuning display window [provided in the MMI software] which shows a time sample).

SYNTAX:

DS *axis, variable, scale, DAC_chan*

axis	Axis number, where: 0 = no display 1-4 = Axes 1-4
variable	Data to be displayed, where: PC = Position Command - mach steps VA = Actual Velocity - mach steps/0.25 ms PA = Actual Positions - mach steps VE = Velocity Error - mach steps/0.25 ms PE = Position Error - mach steps TQ = Torque Output - ± 10V (scale doesn't apply here) VC = Velocity Command - mach steps
scale	Voltage per bit (0 to 10 volts/bit).
DAC_chan	DAC channel number (1-8).

The DS command should not be issued to a channel that is being used for servo or stepping motor operation.



The following channel-signal relationship exists:

A Channel	Signal	P1 Connector	U500ISA Test Point
1	ICMD1B	79	TP16
5	ICMD1A	80	TP20
2	ICMD2B	81	TP17
6	ICMD2A	82	TP21
3	ICMD3B	83	TP18
7	ICMD3A	84	TP22
4	ICMD4B	85	TP19
8	ICMD4A	86	TP23

EXAMPLE:

```
DS 1,TQ,10,5        ;Displays torque output information of axis 1's servo loop,
                    ;to DAC 5 (ICMD1A) within the range of 0-10 volts
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.



7.8.31. DWELL

The DWELL command establishes a time delay (in milliseconds) of a programmed duration. The DWELL command must occupy it's own block within a program.

SYNTAX:

DWELL *time*

DW *time*

G4 *time*

time Duration of dwell in msec.

The dwell time may be set at zero to 2^{23} msec.

EXAMPLE:

DWELL 100	;Program execution delayed for 100 msec
G4 V10	;Program execution delayed for the time stored in variable ;10

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

GAIN, WAIT, QUEUE INPUT



7.8.32. DY (Dynamic Gain)

The DY command is used to control the position loop gain, Kpos . When the axis has a non-zero velocity command, Kpos will be set to the gain specified in the argument. When there is no commanded velocity and the timeout (specified in milliseconds) has been exceeded, Kpos will again have the value set in axis parameter 25.

SYNTAX:

DY axis time Kpos

axis: X,Y,Z,U

time: the amount of time after the velocity command is 0, that Kpos is set to the value of axis parameter 25

Kpos: the position loop gain

EXAMPLE:

DY X 20 1 ; set Kpos for the x axis to 20 with a 1 ms time-out.

Related Commands:

GAIN

7.8.33. EC



This command will calculate the correction factor for the wavelength of light based upon the actual temperature, pressure, and humidity. It is typically used with Aerotech's LZR1100 environmental compensator. This correction factor will affect the actual distance traveled during a G0, G1, G2, or G3 command. This command is modal and will affect the specified axes until the environmental compensation is turned off. Before each move for any axis that is specified in the "EC" command, the sensor values are sampled and converted to calculate the correction factor.

The Temperature, Pressure, and Humidity are read as voltages from the LZR1100 sensor to the UNIDEX 500 analog inputs. Temperature is converted to units of degree Celsius. Pressure is converted to absolute millimeters of mercury. Humidity is converted to a percentage, 50 represents 50%.

The sensors must be connected to the analog inputs as follows:

Sensor	Analog Input
Temperature Input 1	0
Pressure Input	1
Humidity Input	2
Temperature Input 2	3

The offsets and scaling constants for the sensors are stored in a calibration file. In the calibration file, any text after a semicolon is considered a user comment. Up to 8 values will be read from the file and the values must be in the order used in the following example calibration file:

```
; Sample LZR1100 – U500 environmental compensation calibration file
12.004      ; Temperature #1 Input Scale      not used = 1
15.222      ; Temperature #1 Input Offset     not used = 0
72.536      ; Pressure Input Scale            not used = 1
535.74      ; Pressure Input Offset           not used = 0
1           ; Humidity Input Scale            not used = 1
0           ; Humidity Input Offset           not used = 0
1           ; Temperature #2 Input Scale      not used = 1
0           ; Temperature #2 Input Offset     not used = 0
```

The calibration file should be specified in the "System Options" section of the MMI. Refer to Chapter 4 Section 4.3.4. for details. The calibration file is usually created by Aerotech.

SYNTAX:

EC *on/off axis analog_mask*

on/off	Turns compensation mode on or off. If turning compensation off, no other arguments are required. If turning compensation on, both the AXIS and ANALOG_MASK arguments are required
axis	X, Y, Z, U …. any combination of axes may be specified
analog_mask	Bit mask that determines which analog inputs are being used: bit 0 = 1 if Analog Input 0 (A/D 1) is used bit 1 = 1 if Analog Input 1 (A/D 2) is used bit 2 = 1 if Analog Input 2 (A/D 3) is used bit 3 = 1 if Analog Input 3 (A/D 4) is used

EXAMPLES:

EC ON X Y 3	; turn compensation on for axis x and y using analog inputs 0 ; and 1
EC ON Z 1	; turn compensation on for axis z using analog inputs 0
EC OFF	; turn compensation off

Note: The axis and analog_mask parameters are cleared (set to zero) when the next EC ON command is issued. For example, the following command sequence:

EC ON X Y 3

EC ON Z 3

will cause only the Z axis to be affected by Analog Inputs 0 and 1.

7.8.34. ENABLE



The ENABLE command is used to enable one or more axes. The U500 turns on the amplifier and starts to close the servo loop. It is also needed to enable scanning of the A/D channel registers.

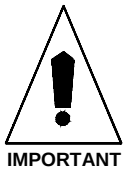
SYNTAX:

ENABLE *axis* [[**AD**]]	Enable axes and optionally enable A/D registers.
EN *axis*	Enable axes.
EN AD	Engage the reading of the A/D registers and put the values in the A/D registers.
axis	Defines the axes (X, Y, Z, or U) to be enabled.

EXAMPLE:

```
ENABLE X Y Z        ;or
EN X Y Z            ;motors for axes X, Y, and Z are enabled
ENABLE X Y AD       ;engage the reading of the A/D registers
                    ;enable X, Y axes as well
V0=$AD0             ;read the value at A/D channel #0
V1=$AD1             ;read the value at A/D channel #1
V2=$AD2             ;read the value at A/D channel #2
V3=$AD3             ;read the value at A/D channel #3
DISABLE AD          ;disengage the reading of the A/D channels
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.



An axis should not be enabled until the initial system checks (described in Chapter 3) have been completed.

Related Commands:

DISABLE



7.8.35. ERROR

The ERROR command is used to change the current fault mask. This overrides but does not change the fault mask parameter defined in the parameter file.

SYNTAX:

ERROR *axis, masknumber, fault_bit*

axis	X, Y, Z, or U
masknumber	0-6, where: 0 = Global Fault Mask 1 = Disable 2 = Interrupt 3 = Aux Output 4 = Halt 5 = Abort 6 = Brake
fault_bit	BA#: 0 = Position Error 1 = RMS Current Error 2 = Integral Error 3 = Hardware Limit + 4 = Hardware Limit - 5 = Software Limit + 6 = Software Limit - 7 = Amplifier Fault 8 = Feedback Device Error 9 = Velocity Feedback Device Fault 10 = Hall Feedback Fault 11 = Reserved 12 = Feedrate greater than Max Error 13 = Velocity Error 14 = Emergency Stop 15 = Reserved 16 = Axis # 1 Any Fault 17 = Axis # 2 Any Fault 18 = Axis # 3 Any Fault 19 = Axis # 4 Any Fault



Bits 11 and 15 are reserved.

EXAMPLE:

```
ER X,0,071FF          ;Set X Axis error mask as hex data 071FF
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

Parameter x*55*

An axis should not be enabled until the initial system checks (described in Chapter 3) have been completed.



7.8.36. EXIT



The EXIT command is used to terminate program flow. If this command is not used in a program, program flow will continue to the end of the program file.

SYNTAX:

EXIT

EX

M2

EXAMPLE:

```
;*******************************
;Main Program
;*******************************
     :                  ;Body of main program
     :                  ;Additional commands
SUB  :SUB1              ;Go to subroutine SUB1 then return
SUB  :SUB2              ;Go to subroutine SUB2 then return
     :                  ;Additional commands
     :                  ;Additional commands
EXIT                    ;End of Main Program (Don't fall through to subroutines)

:SUB1                   ;Subroutine SUB1
     :                  ;Additional commands
RETURN                  ;Return to main program

:SUB2                   ;Subroutine SUB2
     :                  ;Additional commands
RETURN                  ;Return to main program
```

Interpreter level.



7.8.37. FAULT ACKNOWLEDGE

The FAULT ACKNOWLEDGE command performs the same function as pushing the "FLTACK" soft key. This will clear any axis faults such as position error, RMS current error, and similar errors. If an axis is in a limit, the FA command will cause the UNIDEX 500 to move out of the limit.

SYNTAX:

FA

There are no arguments with the FA command.

EXAMPLE:

FA

Related commands:

ABORT



7.8.38. FL (Filter Time Constant)

The Filter Time Constant (FL) command is used in conjunction with the alternate contouring mode. This command activates an exponential filter on the specified axis. The time constant of the filter is given in milliseconds. The primary use of the filter is to smooth a trajectory that consists of non-tangential moves in G8 (velocity profiling) mode. The filter should also be used in the alternate contouring mode if feedhold or MFO is desired. A low filter value (10 ms) is sufficient in these cases. A filter time constant of 0, turns the filter completely off. A parameter setting of 1 dissipates the filter contents with no filter affect. If you are not planning to use the filter, the time / parameter should be set to 0.

SYNTAX:

FL X#, **Y**#, **Z**#, **U**#

#	Defines the filter time constant in ms.

EXAMPLE:

FL X10 ;10 ms filter

Programming level: U500

Related Commands:

CM

7.8.39. FREEDLL



The FREEDLL command will free a dll from memory after the LOADDLL command was used.

SYNTAX:

v# = **FREEDLL** v#

where,

v# (argument) is the handle of the dll that is open

v# (return) returns 0.

Related Commands:

CALLDLL , LOADDLL



7.8.40. FREERUN

The FREERUN command is used to produce background motion of designated axes. Freerun motion is completely unsynchronized to contoured motion.

SYNTAX:

FREERUN *axis±feedrate,distance*

FR *axis±feedrate,distance*

axis	Defines the axes (X, Y, Z, or U) under Freerun control.
± feedrate	Defines direction and velocity of the axes under freerun. If the feedrate equals zero, freerun will stop. Upon initializing the U500, the default units for feedrate are in units/min. The units are defined by the English and metric conversion factors. The units for feedrate can be changed by the PROGRAM command.
distance	Defines the distance of the freerun. If the distance is not included, the axis will run until commanded to stop.



The FREERUN command must be used with one axis at a time.



Following freerun of an axis, it is advisable that the SOFTWARE POSITION command be executed to update the position registers with the current position of the axes. See Section 7.4.

Trajectory for freerun motion is linear only. Ramping is based on maximum acceleration. The corner rounding and velocity profiling options are not available to axes under freerun control.

EXAMPLES:

FR X100	;The X axis will run continuously in the positive direction ;at a feedrate of 100
FR Y-100,2000	;Enables the Y axis for freerun in the negative direction ;with a feedrate of 100. Freerun will stop at 2000
FR X0	;Stops X axis freerun

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

INDEX, LINEAR, CW_CIRCLE, CCW_CIRCLE, ACCELERATION, SOFTWARE POSITION, PROGRAM, G70, G71, G91, G92

7.8.41. GAIN



The GAIN command is used to set servo loop related values. These values override, but do not change the corresponding axis parameter values. Refer to Chapter 5 of this manual for a detailed explanation of all of the gain values.

SYNTAX:

GAIN *axis param_name&val*

GA *axis param_name&val*

axis — Defines the axis (X, Y, Z, or U) that receives the gain value change.

param_name&val — Specifies a servo loop parameter name followed by an associated value. Valid options for *param_name&val* are:

[[**KPOS***val*]] [[**KI***val*]] [[**KP***val*]] [[**VFF***val*]] [[**AFF***val*]]
Used to enter new values for the servo loop gain values (x25, x26, x27, x28, and x29). One or more of these parameters may be used in succession.

[[**N0***val*]] [[**N1***val*]] [[**N2***val*]] [[**D1***val*]] [[**D2***val*]]
Defines new notch filter coefficient values (x31, x32, x33, x34).

NOTCH{0|1} — Used to enable (1) or disable (0) the notch filter function (x24). NOTCH can be abbreviated NO.

DEADBAND*val* — Used to enter a new value (in machine steps) to be used as a positional deadband (x35). DEADBAND can be abbreviated DE.

EXAMPLES:

GAIN X KPOS1500 **KI**500 **KP**600000 **VFF**256 **AFF**0
;Sets all servo loop gains

GAIN Y N0=.0055 **N1**=.0111 **N2=**.0055 **D1=-**1.778 **D2**=.8
;Sets notch filter coefficients for the Y axis

GAIN Y NOTCH1 ;Turns the notch filter ON for the Y axis

GAIN Z DEADBAND10 ;Sets the Deadband for 10 machine steps

GAIN X KPOS1000 ;Sets Kpos servo loop gain for the X axis to 1000

GAIN Y VFF0 ;Sets Velocity feedforward of Y axis to zero

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.



7.8.42. GEAR

The GEAR command moves an axis (the slave) based on the feedback of another axis (the master). The slave axis follows the motion of the master axis from the time the command is given until the gearing is released. The master axis can be a handwheel or other feedback device. The master axis is not required to be a closed loop servo.

SYNTAX:

GEAR *slave_number,master_number,slave_ratio,master_ratio*

GE *slave_number,master_number,slave_ratio,master_ratio*

slave_number Defines the slave axis (1,2,3, or 4).

master_number 0 = disengage gear.
1,2,3,4 = axis.
S1 = use iSBX encoder board.

slave_ratio and master_ratio
Positive or negative integer value. Maximum value is 8,388,608.

These optional parameters specify the scaling to be used. If they are not used, a one-to-one linkage is generated.

The ratio of these two arguments (slave_ratio/ master_ratio) represents the scaling of slave counts to master counts. A negative number may be specified to provide a reverse direction of motion to the slave axis.

EXAMPLES:

```
GE 2,1          ;Link axis 2 to the motion of axis 1 with a 1:1 ratio
GE 2,1,-3,10    ;Link axis 2 to axis 1. The axes will move with the
                ;following ratio:
                  ;-3 slave steps : 10 master steps (= -3). The slave will
                  ;move -3 machine steps for every 10 machine steps of
                  ;the master
GE 1,S1,1,4     ;Link axis 1 to the iSBX encoder input with handwheel
                ;scaling (1:4)
GE 2,0          ;Disengages the gear
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

SOFTWARE POSITION



To engage and disengage a handwheel, the GEAR command should be used with the handwheel connected to one of the feedback channels (1,2,3,4,S1).

After using the GEAR command, the position registers need updated with the SOFTWARE POSITION command. See Section 7.4.

7.8.43. GOTO



The GOTO command is used to direct program flow to a previously defined label or another program. Variable labels are accepted for branching using the "**:%v**###" syntax. The RETURN command is not used in conjunction with the GOTO command. Refer to Figure 7-6.

SYNTAX:

GOTO *:label*

GO *:label*

GOTO *program*

GO *program*

GO :%v*###*

GO LINE *##*

:*label*	Specifies a label name within a program (program flow will go to the specified label and then continue from there).
program	Specifies a program name (program flow will go to another program). The called program must be identified using the format *Drive:\Directory\Filename.ext*. When the called program is finished, control is returned to the original program. A RETURN command is not used in conjunction with the GOTO statement.
###	A U500 variable 0 through 255.
##	Sets program to line number ##.

EXAMPLES:

```
GO :Sec1            ;Program flow goes to label :SEC1 and begins execution
   :                ;⎫
   :                ;⎬These lines are skipped
   :                ;⎭
:SEC1               ;Program execution continues here...

GOTO C:\Prog1.prg   ;Program flow goes to the "Prog1.prg" file and begins
                    ;processing
   :                ;When finished, program control returns here and
                    ;continues
v25 = 700
GO :%v25            ;program execution will jump to label :700
...
:700
ME DI "program jumped to here"
```

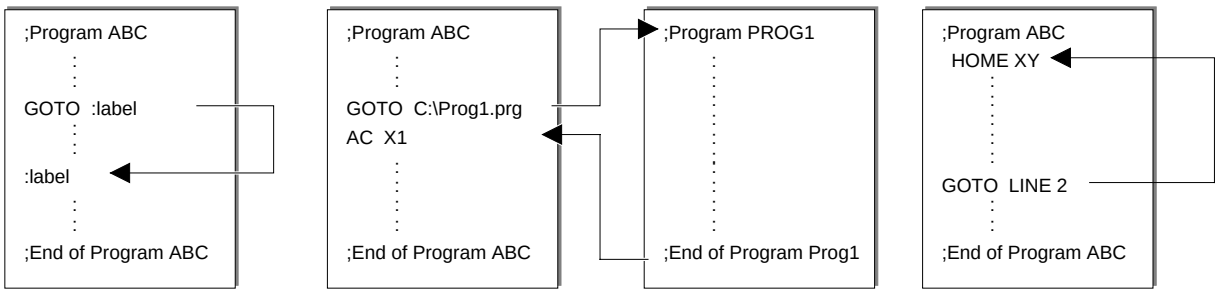


Figure 7-6. Sample Uses of the GOTO Command

C library level.

Related Commands:

SUBROUTINE, LOOP, IF, WHILE

The GOTO command is only available with the MMI software interface.

7.8.44. HALT



The HALT command is used to stop execution in the current contour plane. When a HALT command is initiated, the UNIDEX 500 retains all commands in an internal queue buffer. These commands are not executed until the halt condition is removed by a START or TRIGGER command.

The HALT command affects the current plane only. The UNIDEX 500 ensures that all planes do not process the HALT command simultaneously.

SYNTAX:

HALT

HA

EXAMPLE:

```
HALT            ;Stops processing all commands in the current contour
                ;plane
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

WAIT, START, QUEUE, TRIGGER



7.8.45. HOME

The HOME command is used to move specified axes to the hardware home position. The home sequence is described in Chapter 5 of this manual.

SYNTAX:

HOME *axis*

HO *axis*

axis	Defines the axis (X, Y, Z, or U) to send to the home position.

EXAMPLES:

HOME X	;*or*
HO X	;X axis sent home
HOME X Y Z U	;*or*
HO X Y Z U	;All axes sent home

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

SOFTWARE HOME, G92

7.8.46. IF



The IF command is used to signal a conditional GOTO. First, two values are compared. If the result of the comparison is true, then the program flow will go to a designated label or program. Otherwise, the next sequential command is decoded. The available comparison operators are shown in Table 7-17 below.

SYNTAX:

IF *val1operatorval2 destination*

IF *(expression)* **THEN** For 32 bit software only.
…
ELSE
…
ENDIF

IF *(expression)* **THEN** For 32 bit software only.
…
ENDIF

val1	First value of comparison. The value may be in the form of an integer or variable.
val2	Second value of comparison. The value may be in the form of an integer or variable.
operator	Comparison operator. Refer to Table 7-17.
destination	Label to go to when condition is true, or subroutine to go to when condition is true. Destination can either be a label or a subroutine. A label is specified by a ":" symbol. A subroutine is specified by "SU" followed by the label of the subroutine. When the subroutine is finished, the U500 executes the next command after the IF statement.
expression	Any legal U500 comparison.

Table 7-17. Comparison Operators

Operator	Function
=	equal to
<	less than
>	greater than
<>	not equal to
>=	greater than or equal to
<=	less than or equal to

EXAMPLE:

```
:LOOP                          ;Label
IF $IN1=2 :LOOP                ;Waits for input bit 1 to go low before continuing
IF $IN4 = 0 SU :CUTHOLE        ;If input #4 is low, then execute subroutine called
                               ;CUTHOLE
IF (V100 > V150+50) THEN
  …
ENDIF

IF ($INP &0x01) THEN           ;check input is bit # 0 level
    ME DI “INPUT HIGH”
ELSE
    ME DI “INPUT LOW”
ENDIF
```

C library level.

Related Commands:

SUBROUTINE, GOTO, LABEL MARKER “:”, WHILE



This command is only available with the MMI software interface.

7.8.47. IMMEDIATE



The Immediate (IM) command is used with large program files that are too big to be loaded into the PC's memory. The command should be the first line in a program. A program with this command line in it cannot contain any looping GOTO or other conditional statements such as IF, WHILE, etc. When running in this immediate mode, the program list box is not filled. Instead, the program is executed by sending a command directly to the U500 board from the PRG file.

SYNTAX:

IM

There are no arguments needed with IM command.

This command is only available with the MMI software interface.





7.8.48. INDEX

The INDEX command specifies point-to-point non-synchronized motion of any or all axes. The ramping of each axis is based on its individual maximum acceleration rate. Sinusoidal ramping is used for both acceleration and deceleration. INDEX-type moves always decelerate to a stop at the end of the move.

SYNTAX:

INDEX *axis distance axis feedrate ...*

G0 *axis distance axis feedrate ...*

axis	Defines the axis involved in the motion (X, Y, Z, or U).
distance	Defines the length of the movement.
feedrate	Defines the INDEX feedrate for the specified axis (XF, YF, ZF, or UF). Upon initializing the U500, the default units for feedrate are in units/min. The units are defined by the English and metric conversion factors. The units for feedrate can be changed by the PROGRAM command.



If the target distance is insufficient for ramping, the UNIDEX 500 automatically calculates and implements the shortest path to the target.

All axes specified with this command start motion at the same time but stop relative to their individual axis' target distance and feedrate. When the target distance is reached for the specified axes, the UNIDEX 500 executes the next command block.

Since each axis motion is done independently, the corner rounding or velocity profiling motion options are not available for use with this command.

The maximum distance that can be specified by the INDEX command is 2^{31} machine steps. The maximum feedrate that can be specified by the INDEX command is 2^{15}.



If an indexing feedrate subcommand is missing for an axis, the UNIDEX 500 will use a previously programmed feedrate or the feedrate established by general parameters 023-026, 041-044, 059-062, and 077-080 (*X, Y, Z, and U point-to-point feedrate [program steps/ms]*).

EXAMPLES:

```
INDEX X10 XF100 Y20  ;Index move axis X 10 using the specified feed rate for
                     ;axis X. Index move axis Y 20 using the default or
                     ;previous index feed rate
G0 X10 Y-20 Z-30 U40 ;Index move X 10, Y -20, Z -30, and U 40 using the
                     ;previously set feed rate used by each of these axes
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

LINEAR, CW_CIRCLE, CCW_CIRCLE, FREERUN, ACCELERATION, PROGRAM, G70/G71, G90/G91

7.8.49. INn (Read Inputs)



This command reads the inputs of the U500 card or the 4EN encoder card. IN0 is similar to $INP command, however, the IN0 command reads the inputs directly. The $INP command input value is updated every 100 ms when running from MMI. The "n" in the INn command must be filled by a number from 0 to 4 in order for it to be distinguished from the INdex command.



General parameter number 99, bit no. 0 should be set to 1 when using the 4EN option board's I/O.

SYNTAX:

INn	where "**n**" is a number from 0 to 4.
INn, *bit*, *bit*	
n	0: for U500 card 16 inputs. 1: 4EN option board - read back OU1 bits. 2: 4EN option board (24 inputs). 3: 4EN option board (16 in, 8 out). 4: 4EN option board (12 in, 8 out).
bit	0-23 for bit number to read.

EXAMPLES:

```
V0=IN0          ;read UNIDEX 500 input bits
V0=IN2,0,1      ;read bits 0 and 1 of 4EN option board P8 connector
```

Programming level: Real time status information executed by the host processor and the UNIDEX 500.

Related Commands:

OU, OUn, $INP, OEn



7.8.50. INTERRUPT

The INT command is used to interrupt program execution. It does this by scanning an input bit for a defined logic level. If the input bit level is the same as the defined level, a dump is made to a label in the program. This function is useful for implementing push button "boxes" used as operator interfaces. The inputs are scanned every 100 ms.

SYNTAX:

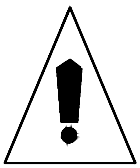
INT *board, nLevel, nInputBit, nOnOff, Label*

board	Board to set interrupt on (1-6).
nLevel	Level of interrupt where: 0 = turn off 1 = jump to label (abort all motion) 2 = jump to label (don't abort motion) 3 = jump to label, turn off interrupt, (no abort) 4 = jump to label, turn off interrupt, (abort motion)
nInputBit	Input line to trigger interrupt–(0 - 15 for U500 input bits or $000 - $127 for SBX-IO48 input bits).
nOnOff	0 = int when *nInputBit* low. 1 = int when *nInputBit* high.
Label	Label to jump to on interrupt.

EXAMPLES:

INT *1,1,0,0,:turnoff*	;Interrupt on board 1, input 0. Jump to :turnoff when input ;bit 0 goes from high to low. Abort all motion
INT *1,4,8,1,:exit*	;Interrupt on board 1, input 8. Jump to :exit when input bit ;8 goes from low to high
INT *1,0,10*	;turn off int for board 1, input 10

C library level.



This function only works when a user program is executing.

7.8.51. IO COMMAND



This command is only valid with the PCI version of the U500 board.

The U500PCI contains an additional set of 24 I/O bits. The I/O are located on the P4 connector. They are arranged as 3 ports of 8 bits. Each port is configurable as outputs or inputs. The direction of each port is set with the "IOSET" command. I/O data is read and written using the "IO" command.

The active state of all outputs is low. Therefore, a programmed logic "1" will result in a low impedance to GND for that output. A programmed logic of "0" will result in a high impedance state for that output. All I/O bits become inputs (high impedance) during reset.

SYNTAX:

IO*port value*

OR **IO***port bit#*, 0/1,...

port	Port of 8 X 3 I/O connector, 0-2 (A, B, or C).
value	Specifies the 8 bit output data 0-255 or 0xFF.
bit#	Specifies the bit number that is affected.

EXAMPLE:

```
IO0 0,1,1,1,2,0      ;Set bits 0 and 1 to 1, bit 2 to 0 of port 0
V0=IO1               ;Read 8 bit inputs of port 1
```

7.8.52. IOSET COMMAND



This command is only valid with the PCI version of the U500 board.

The IOSET command configures the direction of each port of the 8 X 3 I/O bus available on the U500 PCI card at connector P4. This bus is configurable in groups of 8 bits as inputs or outputs. All ports are set as input after a hardware reset. Output data can be written to this port (using the IO command) before the direction is configured.

SYNTAX:

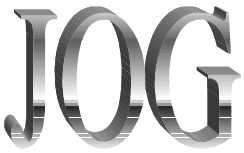
IOSET port,dir,port,dir,port,dir

port	Port 0 through 2 (Port 0 = A, Port 1 = B, Port 2 = C).
dir	0 for input, 1 for output.

EXAMPLE:

```
IOSET 0,0,1,0,2,0    ; set all ports as inputs
IOSET 0,1,1,1,2,1    ; set all ports as outputs
IOSET 0,1            ; set port 0 as output, other ports unchanged
```

See Table 12-36 in Chapter 12 for connector pinouts.



7.8.53. JOG

The JOG command calls the Jog screen from a program. When this occurs, the program execution pauses and the Jog screen appears so the operator can jog the axes. The user presses the Quit button in the Jog screen to return to the program.

SYNTAX:

JOG

JO

There are no arguments needed with the JOG command.

EXAMPLE:

```
ENABLE XY
HOME XY
   :
JOG
   :
```

Related Commands:

FREERUN, SLEW



7.8.54. Label Marker (:)

A label is an ASCII string that may be used to define an entry point within a program. The label command must occupy it's own program block.

SYNTAX:

:*label*

label	Specifies an ASCII string defining the label.

EXAMPLE:

```
:SECT1          ;Inserts the label "SECT1" into the program. Label names
                ;are arbitrary.  Label markers may be used as an entry
                ;point for a GOTO or SUBROUTINE command
```

C library level.

Related Commands:

GOTO, SUBROUTINE, IF



This command is only available with the MMI software interface.

7.8.55. LINEAR



The LINEAR motion command initiates contour motion in which each axis adjusts its own feedrate to maintain a contour path. All specified axes start and stop at the same time. The UNIDEX 500 uses the current contour plane's ramp time to produce either a linear or sine-type ramp to get all axes to steady speed. The same ramp time is used to ramp down to the target distance.

The contour feedrate is clamped at the rate set by general parameter 027, 045, 063, or 081 (*Clamp feedrate [prog. steps/ms]*). If a new feedrate is specified in velocity profiling mode (G8), the U500 will ramp to it at the beginning of the move.

SYNTAX:

LINEAR *axis distance...* {**F***feedrate*|**FT***feedrate_time*} [[{SAn|SDn}]]

G1 *axis distance...* {**F***feedrate*|**FT***feedrate_time*}

axis	Defines one or more axes (X, Y, Z, U) for motion.
distance	Defines the distance of the move.
feedrate	Defines a new contour feedrate. Upon initializing the UNIDEX 500, the default units for feedrate are units/min. The units are defined by the English and metric conversion factors. The units for feedrate can be changed by the PROGRAM command.
feedrate_time	Defines the time (in seconds) that is allocated to complete the contour move. The UNIDEX 500 calculates the contour feedrate based on the contour path. The *feedrate_time* value is valid for the current block only.
SAn	Starts specified plane "n" when acceleration is complete.
SDn	Starts specified plane "n" at the beginning of deceleration.



If the target distance is insufficient for ramping, the UNIDEX 500 preserves the programmed acceleration.

When using the UNIDEX 500-Plus or the UNIDEX 500-Ultra, the corner rounding or velocity profiling motion options may be used in conjunction with this command.



If a feedrate subcommand is missing, the UNIDEX 500 will use a previously programmed feedrate or the feedrate established by general parameter # 022, 040, 058, 076 (*Contour feedrate [program steps/ms]*).

EXAMPLE:

```
LINEAR X10 Y20 F100   ;Contour move of  X and Y axes at contour feedrate
G1 X10 Y-20 Z-30 U40  ;Contour move of all four axes at a previously set feedrate
LINEAR X=V10 F=V11    ;Contour move of X axis with the distance and feedrate
                      ;contained in variables 10 and 11
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

INDEX, CW_CIRCLE, CCW_CIRCLE, FREERUN, RAMP, ACCELERATION, ROUNDING, VELOCITY, TRAJECTORY, PROGRAM, G70/G71, G90/G91, CM, EF

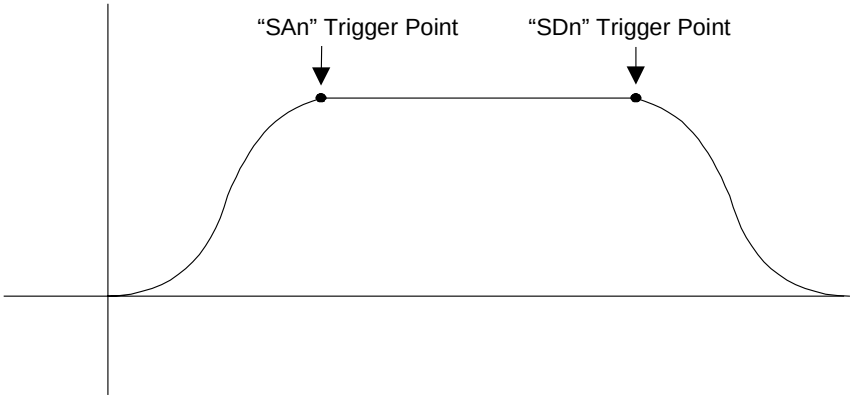


Figure 7-7. Sine-Type Ramp



7.8.56. LOADDLL

The LOADDLL command will keep a DLL loaded in memory during execution of a MMI program for subsequent uses of the CALLDLL command.

SYNTAX:

v# = **LOADDLL** "*dllname*"

where,

"dllname" is the name of the dll to load into memory

v# (v0 – v255) returns the handle to the dll

Related Commands:

CALLDLL , FREEDLL

7.8.57. LOOP



The LOOP command signals the beginning of a group of program statements to be repeated, and specifies the number of repeats.

SYNTAX:

LOOP *number*

LO *number*

number	Specifies the number of times to repeat the statements in the "command block." The command block consists of the program statements contained between the LOOP and NEXT commands.

EXAMPLE:

```
LOOP 5          ;Signifies the start of the command block
X10             ;⎫
  :             ;⎬ Command block
X-10            ;⎭
NEXT            ;End of command block.  Repeat 5 times
```

C library level.

Related Commands:

NEXT



7.8.58. LVDT

The LVDT command links an axis number (1-4) with an R/D channel number (9-16) of an RDP-PC board for very accurate positioning. When the LVDT command is executed, the specified axis moves in the specified direction (CW or CCW) until the RDP-PC board determines that the LVDT sensor is at its null position (i.e., a point that is halfway between the minimum and maximum position). The axis moves in the search direction at the rate set by the *Power-on home feedrate (machine steps/ms)* (axis parameter x04). When the LVDT reaches a specified point, the axis will ramp down and begin searching for the null position. The axis stops on the null position of the LVDT sensor, completing the cycle.

SYNTAX:

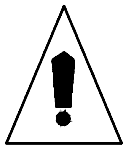
LVDT *axisnumber,channelnumber,direction* [[*speed1,speed2*]]

axis number	Specifies the axis number (1, 2, 3, or 4) for the LVDT reference.
channel number	Specifies the RDP channel number to be referenced by this command. The default for unused RDP channels is 16 bit mode. The LVDT command must specify an RDP channel (9-16). No other axis should reference this channel through axis parameters x38, x39, x40 or x41.
direction	Direction of search (CW or CCW).
speed 1	Speed axis moves in machine steps/ms until the ramp-down point.
speed 2	Speed axis moves, in machine steps/ms until the null is found.

EXAMPLE:

```
LVDT 1,9,CW        ;Axis #1 uses R/D channel #9 for feedback. Direction of
                   ;motion is CW
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.



The U500 RDP board is required to provide the feedback channel. Also, no other axes should reference the argument channel through axis parameters x38, x39, x40 or x41.

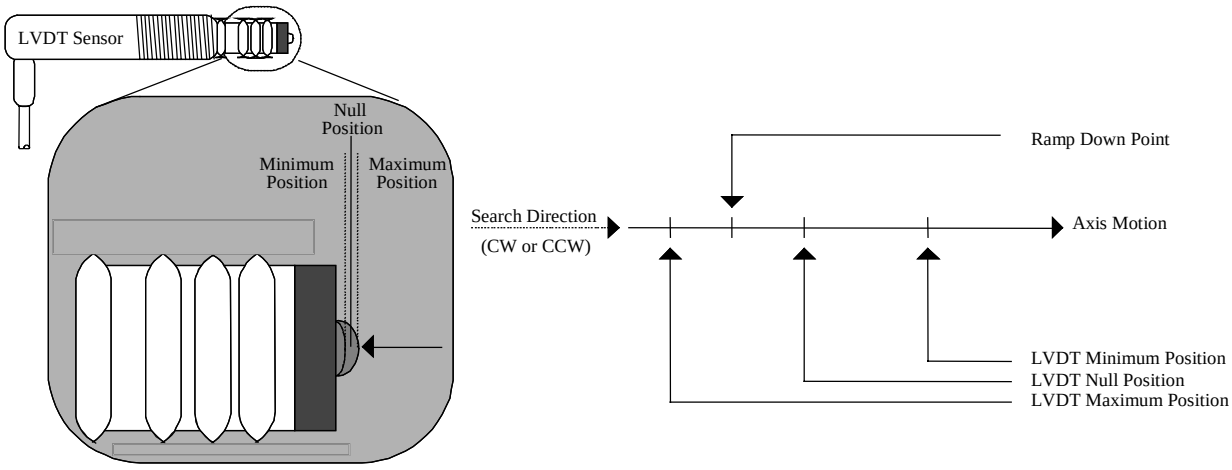


Figure 7-8. LVDT Sensor

7.8.59. M0 (“M Zero”)

The M0 ("M zero") command is used to initiate a pause in the program flow to wait for input from the CYCLE START button. Program execution continues when the CYCLE button is pushed.



SYNTAX:

M0

There are no arguments needed with the M0 command.

EXAMPLE:

```
M0                    ;Wait here until "step" key is pressed
```

C library level.

MC

7.8.60. MCOMM (Motor Commutation)

The MC (motor commutation) command is used to setup AC brushless motors for commutation with the UNIDEX 500. This command automatically disables position, velocity, and integral traps and outputs a current (torque) vector, which is 90 degrees advanced from the rotor vector. The command can also be used to output a constant torque. The motor should spin freely and smoothly in the direction specified. Refer to Section 4 and Appendix F of this manual for additional motor configuration information.

SYNTAX:

MC *axis, volts*

axis	Axis X, Y, Z, or U
volts	-10 to +10 volts of commutation where: < 0 = CCW rotation = 0 = Cancels the function > 0 = CW rotation

EXAMPLES:

```
MC X,2.5        ;Set drive #1 to 2.5 volts for commutation
MC X,0          ;Cancel previous command, disable axis

ENABLE Z        ;Enable third axis (Z)
MC Z,1          ;One volt peak output to axis 3
MC Z,0          ;Stop rotation and disable axis
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

MSET, parameters x38-x44



To avoid damage to the motor and related equipment, the motor must be disconnected from the load.

Refer to Chapter 3: Software Installation and Fundamentals and Appendix F for more information about motor commutation and setup.

7.8.61. MESSAGE



The MESSAGE command is used to send a message to the display, printer, serial port, or a file. Variable values may also be entered. The message may be text or the value of user or system variables.

SYNTAX:

MESSAGE *dest1+dest2+var_input "message"*

ME *dest1+dest2+var_input "message"*



dest# A message may be sent to one or more of the following destinations using the syntax shown below:

MESSAGE DISPLAY A message is sent to the MMI display.

MESSAGE FILE(*path\name.ext, w|a*)
A message is sent to the file *path\name.ext* where:
w = write to a new file (any existing file is overwritten)
a = append to the end of an existing file.
Use the "w" argument to create and write the first line of a new file. Then use the "a" argument to append messages to the new file.

var_input V0-255 data input.
VV0-255 data input (array).

Formats for displaying variable values are:

%	Real number with 6 digits after the decimal point
%e	Displays number in exponential format (i.e., 1.257e-7)
%.nf	Real number with n digits after the decimal point
%x	Hex format
%%	Display the % symbol

"message" Comprised of text or variable. Must be enclosed in double quotes (" ") or single quotes (' ').

EXAMPLES:

```
ME FILE(TEST.TXT,w) "This is a test"
                    ;Creates a new file and writes "This is a test" to the file
ME DI+V0 "V0 = %V0, ENTER NEW VALUE "
                    ;Displays current value of V0 in a dialog window, and ;prompts
                    for a new value
ME DI+FI(ERR.LOG,A) "Program Error Status %V0"
                    ;Display and append to file called ERR.LOG the value of
                    ;program variable V0 (the program error status)
```

C library level.



7.8.62. MR (Memory Read)

The MR command reads the value of a DSP memory location. This command is for special applications and is not intended for general use.

SYNTAX:

MR *mtype, addr*

mtype	X, Y, or L for X, Y, or L memory space.
addr	address to read within memory space

EXAMPLE:

```
V0=MR X,0xb     ;this reads the data at address 0xb from X memory space
                ;and places the value in v0 (this is the location of the 16
                ;inputs)
```



This command is for special applications and is not intended for general use.

Programming level: Status information executed in real time by host processor and UNIDEX 500.

Related Commands:
MW

7.8.63. MSET (Motor Setup)



The motor setup (MSET) command is used to set a fixed vector when setting up an AC brushless motor. This function outputs a fixed vector current command. The rotor will lock into the commanded position. This function can be used to setup motor phasing by checking the Hall effect states at each point. . Refer to Chapter 5, (Section 5.8.1. "Introduction to Motors and Feedback Configurations"), and Appendix E for additional information.

SYNTAX:

MSET *axis,volts,phase*

MS *axis,volts,phase*

axis	Defines the axis channel number (X, Y, Z, and U). Only one drive at a time may be configured. Entering a 0 after the drive number cancels this command.
volts	Output voltage (0 to +10 volts) (A value of 0 stops motion and disables the axis).
phase	Electrical phase (0 to 360 degrees). Specifies a 0 to 360 degree electrical offset for the torque vector.



To avoid damage to the motor and related equipment, the motor must be disconnected from the load. Also, make sure parameters x53–*Clamp current output (0-100%)*, x48–*RMS current trap (0-100%)*, and x49–*RMS current sample time (1-16,383 ms)* are properly set.

When using this command with a configured system, servo loop traps will occur. To eliminate this, temporarily set the following axis parameters to zero.

(x19) *Max position error (0-8,383,607)*
(x20) *Max integral error (0-8,383,607)*

Do not proceed unless you are sure of maximum motor current and amplifier scaling.

Make certain the *Max position error (0-8,383,607)* and the *Max integral error (0-8,383,607)* parameters are returned to their original values following phasing of the AC brushless motor.



For resolver feedback, the resolver position should be "0000" when the Phase Offset parameter is set to 0.

EXAMPLES:

```
MS X,2.5, 180        ;Sets X axis to 2.5 volts and the vector to 180°.
MS X,0               ;Cancels previous command, disables drive
```

To verify phasing, a small user program may also be written which steps through all states. The motor should rotate in the clockwise direction. If the motor steps in the opposite direction, the motor phasing is incorrect. Refer to Table 7-18.

```
EN U                 ;fourth (U) axis
:loop_label          ;loop back here
MS U,.5,0            ;1/2 volt output - phase 0
DW 500               ;1/2 second pause
MS U,.5,60           ;phase 60
DW 500               ;
MS U,.5,120          ;phase 120
DW 500               ;
MS U,.5,180          ;phase 180
DW 500               ;
MS U,.5,240          ;phase 240
DW 500               ;
MS U,.5,300          ;phase 300
DW 500               ;
GOTO :loop_label     ;continue
```

Table 7-18. Motor Phase Labels and Hall States

	Desired Hall State		
Phase Labels on the Motor	**C**	**A**	**B**
Commanded Vector	*MSB*		*LSB*
330-30 degrees	1	0	0
30-90 degrees	1	0	1
90-150 degrees	0	0	1
150-210 degrees	0	1	1
210-270 degrees	0	1	0
270-330 degrees	1	1	0

The MSET command can be used to initialize commutation. If the axis is enabled while an MSET command is active, commutation will start from the MSET angle (this angle should be 90 degrees). If the axis is enabled when in mset, the U500 will continue to track the commutation vectors when the axis is disabled. Subsequent mset functions are not necessary, but can still be used.

Warning: If the encoder power is lost and the motor shaft is turned, the commutation will not be preserved! This is also true if the maximum tracking rate of the feedback device is exceeded. A subsequent MSET command would be required to initialize the commutation.




```
wa al                          ; wait all
V3=PRM(155)                    ; save the error mask fault for axis 1 into
                                       variable
ER X,0,0                       ; turn off faults
V0=GAIN X KPOS                 ; save the Kpos gain
V1=GAIN X KI                   ; save the Ki  gain
V2=GAIN X KP                   ; save the Kp gain
ME DI "%.0fV0 %.0fV1 %.0fV2"
GA X KPOS=0 KI=0 KP=0          ; set the gains to 0
EN X                           ; enable the x axis
MS X,0,0                       ; clear commutation tracking
EN X                           ; enable the x axis
MS X,1,90                      ; set the initial commutation vector

DW 3000                        ; dwell 3 seconds
GA X KPOS=V0 KI=V1 KP=V2       ; restore the gains
EN X                           ; re-enable the axis
ER X,0,V3                      ; restore the fault mask
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

MC (Motor Commutation)



7.8.64. MW (Memory Write)

The MW command writes a data value to a DSP memory location. The mode argument indicates whether to overwrite the existing memory data, AND the new data with the previous data, or OR the new data with the previous data. This command is not queued in the U500 memory. It is executed immediately by the host processor and the U500. This command is for special applications and is not intended for general use.

SYNTAX:

MW *mtype, addr, data* [,*mode*]

mtype	X or Y for X or Y memory space.
addr	address to write within memory space.
data	data to be written
mode	AND if ANDing data; OR if ORing data; otherwise overwrite data.

EXAMPLES:

MW Y, 0x1a,6	;writes a 6 to memory location 0x1a in Y memory space (this is the location of the 8 outputs)
MW Y, 0x1a, 1, OR	;set bit 0 of the outputs and does not affect the other bits



This command is for special applications and is not intended for general use.

Programming level: Executed in real time by host processor and UNIDEX 500.

Related Commands:

MR

7.8.65. NEXT



The NEXT command is used to specify the endpoint of the group of program blocks comprising a loop. This command is used in connection with the LOOP command.

SYNTAX:

NEXT

EXAMPLE:

```
LOOP 10
X10
DWELL 100
X-10
NEXT                    ;Signals the end of the Repeat Loop.
```

Interpreter level. Executed by the host processor.

Related Commands:

LOOP

This command is only available with the MMI software interface.





7.8.66. OEn (Extended Output)

This command sets output bits on either the U500 card or the 4EN encoder card. The OEn command is executed immediately by the U500. This differs from the "OU" command that is loaded into the U500's queue buffer and executed synchronously with other programmed commands. It is useful for real time control of the output bus. The "n" in the OEn command must be filled with a number from 0 to 4 in order for the command to work.



General parameter number 099, bit number 0, should be set to 1 when using the 4EN option board's I/O.

SYNTAX:

OEn *val* where "n" is a number from 0 to 4.

OEn *bit, highlow, bit, highlow...*

n	0: U500 card (same as OU command except not queued) 1: 4EN option board - P7 (24 out) 3: 4EN option board - P9 (16 in, 8 out) 4: 4EN option board - P10 (12 in, 8out)
val	Value of output.
bit	Bit number to set.
highlow	0 sets bit low, 1 sets bit high.

EXAMPLES:

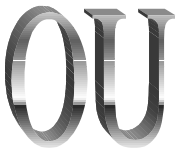
```
OE0 0X55          ; send real time output to U500
OE1 0,1,1,1,2,1   ; send real-time output to 4EN option board P7 connector
```

Programming level: Real-time command executed by host and UNIDEX 500.

Related Commands:

INn, $IN, OU

7.8.67. OUTPUT



The OUTPUT command is used to set or clear individual output bits or write an 8 bit value to the bus. The output value may be specified as either a decimal or hexadecimal number. The actual output polarity is the opposite of the programmed polarity. Programming an output bit as a "1" causes the output to be pulled low. Programming an output bit as a "0" causes the output to be in the high impedance state. The power-on default state of all outputs is "0" (high impedance). Refer to the digital I/O bus specifications in Chapter 12: Technical Details for additional information.

SYNTAX:

OUTPUT *Out_value*

OUTPUT *bitnumber, value*

Out_value	Output value from 0-255 decimal (0-0xFF for hex).
bitnumber	Specifies the bit number that is affected.
value	Specifies the bit polarity.

EXAMPLES:

```
OUTPUT 127          ;Output value specified in decimal
OUTPUT 0x55         ;Output value specified in hexadecimal
OU 0,1,3,0          ;Individual output bits (e.g., 0 and 3) affected. All others
                    ;remain unchanged
OU V0               ;Output condition is specified using a variable
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Output bits OUT0-OUT3 are present on the main P1 connector and P5 expansion connector. Outputs OUT4-OUT7 are available on the P5 connector only.



Related Commands:

OEn, $INP, $IN, INn, IOn



7.8.68. PARALLEL

The PARallel command should be used when in a Velocity Profiling (G8) sequence with Contour Mode 1 . If the angle between contour moves (G1,G2,G3) is greater than the angle specified with this command, the U500 will temporarily switch to G9 mode (i.e., decelerate to a stop).

SYNTAX:

PARallel = angle

angle = max. angle between contour moves before switching to G9 mode, specified in degrees

EXAMPLE:

```
PAR=1        ; set stop condition to 1° between vectors
```

Related Commands:

Velocity, Contour Mode

7.8.69. PRM (PARAMETER READ)



This command is used to read the value of a parameter from the current parameter file. This command will only work for numeric parameters and yes/no parameters. If a parameter value = yes, the variable will = 1, for no, the variable = 0.

SYNTAX:

v## = **PRM** (xxx)

= a variable number

xxx = parameter number

EXAMPLE:

```
v0=PRM(100)          ; reads parameter #100 value into v0
```



7.8.70. PAUSE

The PAUSE ON command is used to map an input bit to the pause function. If the input bit value equals the bitstate specified in the command, the pause function is called. This command is identical to the F3 "PAUSE" button on the bottom of the UNIDEX 500 main window (see Figure 7-9). PAUSE OFF stops the scanning of the input bit. These commands also can be used with the SBX-IO48 board.

SYNTAX:

PAUSE ON, *inputbit,bitstate*

inputbit	Input bit number (0 to 15) or valid SBX-IO48 input bit ($000 to $127).
bitstate	Bit value to send pause, either 1 or 0.

PAUSE OFF

EXAMPLE:

```
PAUSE ON, 3,1        ;Checks input bit 3 for a logical 1. If the value is present,
                     ;calls pause function
```

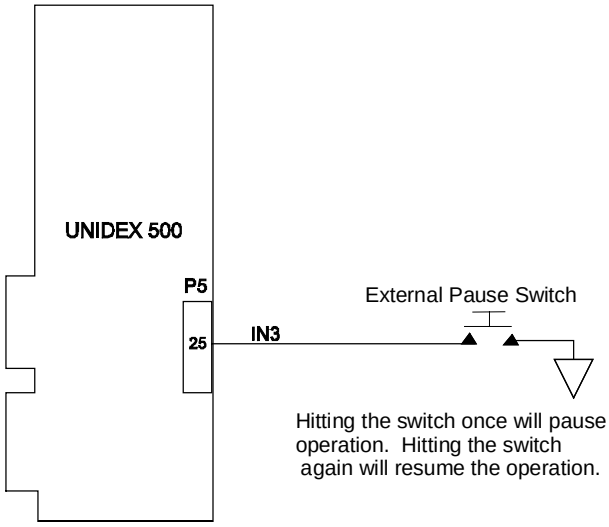
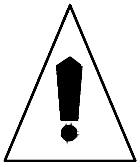


Figure 7-9. PAUSE Function



This functionality is only available when a program is executing.

7.8.71. PLANE



The PLANE command specifies which execution queue commands are being sent to the queue. A queue is a buffer that holds commands. Commands are then executed in the order that they were received. The U500 can be configured for 1, 2 or 4 execution planes. Each plane executes asynchronously and has its own set of execution parameters, such as ramp time, feedrate, units, etc.

The number of planes is set by the general parameter #0.

SYNTAX:

PLANE *number*

PL *number*

number ; Selects one plane at a time selection remains in effect
; until the next plane command is issued.

EXAMPLES:

```
PL 1 G1 X100    ; Move in plane 1, linear move of the X axis
PL 2 G1 Y100    ; Move in plane 2, linear move of the Y axis. Because the
                ; X and Y axes are in different planes, Y will begin motion
                ; at the same time as the X axis
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

MAP, WAIT, START, HALT, TRIGGER, G1/G2/G3, SD, SA, LINEAR, QUEUE



7.8.72. PLC

A PLC is a scanning (i.e., endless looping) program, resident in the UNIDEX 500 memory, which is used primarily for monitoring and I/O control.

SYNTAX:

PLC *arg ...*

arg — The PLC command takes many forms, depending on the desired action:

ON/**OFF** — Converts a UNIDEX 500 task (or plane buffer) to operate a PLC-type program. This first step designates on-board memory to hold the PLC program. These memory areas are typically used as coordinate planes.

A U500 *task* is one of four sets of instructions that are executed individually and sequentially at such a high rate of speed that each task appears to have sole ownership of all of the processor's time. Each UNIDEX 500 PLC program is assigned to a task.

A U500 *plane* is a task that has one or more axes assigned to it.

OPEN *plc_prog_num*

CLOSE — This second step segments the task in as many as 32 separate spaces for storing the PLC-type program. It also facilitates the downloading of program command lines from a host PC to the UNIDEX 500. Because the same command queue memory in the UNIDEX 500 is now reallocated to hold the PLC-type program, the length of sum of all allocated PLC-type programs for the task is limited to (on average) 400-500 commands for all 32 PLC programs in the task. (Only one PLC-type program per task needs to be specified.)

The *plc_prog_num* argument specifies the PLC program number that is about to receive commands. This argument can range from 0-31.

To download a PLC-type program, the PLC ON command must have been executed.

All of the *cmds* commands that follow the PLC OPEN statement are assumed to be the PLC program lines for PLC program number *plc_prog_num*. These commands are downloaded to the designated task and PLC program area. The UNIDEX 500 will continue to accept PLC program commands until: (1) its memory is full or (2) a PLC CLOSE command is received.

FREE *plc_prog_num* This step (if needed) frees previously allocated on-board memory from holding a PLC-type program and prepares it to accept new PLC program data. The *plc_prog_num* argument specifies the program number (0-31) to be erased.

ENABLE *plc_prog_num*

DISABLE *plc_prog_num*

Starts or stops the execution of the board-resident PLC-type program (only enabled PLC-type programs will execute). Once a PLC program is resident in the U500 task, it must either be enabled or disabled. Not all task-resident PLC programs need to execute (i.e., be enabled). In fact, many programs may be memory resident, but only a subset of those need to execute. This command is task- (or plane-) oriented. That is, the command PLC ENABLE 1 is only effective if the UNIDEX 500 is directed to the plane which holds PLC program 1. Note that it is possible for two different tasks to each hold a different PLC program 1.

The *plc_prog_num* argument specifies the program number (0-31) to be enabled or disabled.

The following statements and operators are valid in PLC programs:

IF (*condition*) ... If *condition* (a conditional statement) is true, then execute the block of commands that follow up to the ENDIF statement.

ENDIF ... Marks the end of a block of commands, the start of which is denoted by the previous IF/ELSE command. ENDIF is required every time an IF statement is used.

ELSE ... Follows an IF statement. When the initial IF statement is false, the commands following the ELSE statement are executed. ELSE signals the end of the block of commands started by the previous IF command.

WHILE (*cond*)... While condition *cond* is true, the block of commands that follow are executed.

ENDWHILE... The endpoint for the block of commands started by the WHILE statement.

=, <, > Equals, less than and greater than operators.

AND/OR Boolean operators used to link expressions. Each separate condition must be enclosed in parentheses. These conditions include the following:

I0-I15... 16 input lines from the U500
P0-P23... 24 PLC bits (generic variables - use in PLC only)
PO0-PO3... Axis positions (1-4) in machine steps
$nn (byte input) or $nnb (bit input) where:
nn = SBX-IO48 address 00-0f: iSBX bank 0, address 0-f.
10-1f: iSBX bank 1, address 0-f
b = specifies bit number (0-7) for test

Fx00-Fx15	Fault conditions 0-15 (see below) for axes 1-4 (specified by x)		
	0=position error	5=plus soft limit	11=emergency stop
	1=rms current level exceeded	6=minus soft limit	12=axis 1 any fault
		7=amplifier fault	13=axis 2 any fault
	2=integral error	8=feedback fault	14=axis 3 any fault
	3=plus hard limit	9=feedrate fault	15=axis 4 any fault
	4=minus hard limit	10=velocity error	

EXAMPLES:

```
PLC ON                  ;PLC programming ON

PLC OPEN 1              ;start of program 1
WHILE (I0=1)            ;while input bit0=1
  OU 0x55               ;set even output bits high
  P0,1,2,0              ;set PLC bit 0 to a 1 and bit 2 to 0
ENDWHILE                ;end block of commands
PLC CLOSE               ;end of program 1

PLC OPEN 2              start of program 2
IF ((I0=0) AND (I1=0) OR (I2=0))
                        ;If I0=0 and I1=0, or I2=0
  OU 0x00               ; Set all outputs low
ELSE
  OU 0xFF               ;otherwise, set all outputs high
ENDIF

IF (PO1<1000)           ;If the position of axis 2<1000 machine steps
  OU 255                ;set all outputs high
  P0,1                  ;set PLC bit 0 to a 1
ENDIF
WHILE (F203=1)          ;While axis 2 is in positive hardware limit
  OU 1                  ;set output bit 0 to a 1
ENDWHILE
PLC CLOSE               ;end of program 2

PLC OPEN 3              ;start of program 3
IF ($137=0)             ;check bit #7 = 0, bank 1, address 3
OUTPUT 4,1              ;set output bit 4 to 1
ENDIF

IF ($00=0x1f)           ;check byte = 0x1f (bits #7-5=0, #4-0=1), bank 0, addr. 0
OUTPUT 5,0              ;set output bit 5 to 0
ENDIF

IF ($00=31)             ;same as above except in decimal
OUTPUT 2,1              ;set output bit 2 to 1
ENDIF
PLC CLOSE               ;end of program 3

PLC ENABLE ALL          ;start executing all PLC programs
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

7.8.73. POSITION COMMAND (IN POSITION)



An output bit can be set when a one or more axes are within a specified error band of their final positions. The axes must meet the following criteria to be considered in position:

1) axis must be enabled
2) commanded velocity must be zero
3) actual feedback velocity must be less than 1 count / millisecond
4) position error must be less than axis parameter #0x35 (In position dead-band).

The "POSITION" command performs a logical AND of the specified axes "in position" bits. If any specified axis does not meet the above requirements, the output signal will be false. The output signal can be delayed to allow for any mechanical settling or ringing to damp out.

Note: Individual "in-position" bits are displayed in the diagnostic window.

SYNTAX:

"**POSITION XYZU DWELL** nmsec [**OUTPUT** *bitnum*] [**IO***n,bitnumber*] [**OFF**]"

Table 7-19. Arguments for In Position Command Syntax

Argument	Syntax			
XYZU	Specifies axes to track, one to four axes may be specified here.			
DWELL nmsec	Specifies the number of milliseconds that all specified axes must remain in position before the output bit is set.			
OUTPUT *bitnum*	in-position signal to P5 (16IN/8OUT) connector. Outputs are active low, open collector, and reset to high Z state. Bit#0 = P5-47… Bit #7 = P5-33.			
OFF	optional argument used to turn off the in position monitor.			
IO*n,bitnumber*	generates in position signal on P4 (8X3 IO) port or P6 connector. U500 PCI Only.			
	n	***bitnumber***	**Active Polarity**	**Reset State**
	0	Bit #0 = P4-47… Bit #7 = P4-33	low	high Z
	1	Bit #0 = P4-31… Bit #7 = P4-17	low	high Z
	2	Bit #0 = P4-15… Bit #7 = P4-1	low	high Z
	3	Bit #0 = P6-17… Bit #1 = P6-19	high	GND

EXAMPLES:

```
POS XY DWELL 10 OUTPUT 0      ; monitor X and Y axes. When both axes are
                              within their "in position dead-band"
                              parameters for more than 10 msec, output bit
                              #0 will be set low.

POS OFF                       ;turn off previous function
```



7.8.74. PROGRAM

The PROGRAM (or PR) statement sets the current programming units and mode. The default programming environment is incremental units and units/minute. The default programming system (ENglish/MEtric) may be specified per plane by using general parameters 20, 38, 56, and 74.

SYNTAX:

PRogram *eng_or_met* {EN|ME}
abs_or_incr {AB|IN}
unit_or_step {UN|ST}
unit/min_or_unit/sec_or_step/min_or_step/sec
{UNit/MInute|UNit/SEcond|…}

EN (English)	Motion distance units are specified in English units (e.g., inches, feet, yards, etc.).
ME (Metric)	Motion distance units are specified in metric units (millimeters, centimeters, meters, etc.).



The above subcommands override, but do not change the setting of general parameters 020, 038, 056, or 074 (*Plane n metric system [y/n]*).

The following arguments may be specified in any order. If one of each of the groups is not specified, the one in bold print will be in effect.

abs_or_incr	ABsolute	Motion distance is referenced to the software home position.
	INcremental	Motion distance is an offset referenced from the current position.
unit_or_step	**UNit**	Motion distance is in user units (e.g., inches/millimeters, etc.).
	STep	Motion distance is in program steps.
unit/min_or_unit/sec_ or_step/min_ or_step/sec		
	UNit/MInute _	Motion feedrate is in (inch or mm) per minute.
	UNit/SEcond	Motion feedrate is in (inch or mm) per second.
	STep/MInute	Motion feedrate is in program steps per minute.
	STep/SEcond	Motion feedrate is in program steps per second.

EXAMPLE:

PROGRAM ME Incremental Unit Unit/Minute

```
;The program motion is in the metric mode, each move
;distance is an offset from the current position, the value is
;in millimeters, and feedrate is in millimeters per minute
```

C library level.

Related Commands:

PLANE

A "programming step" is the smallest programming increment. Programming units and steps are related by the *Metric* and *English mode number of decimal digits* parameters 029, 030, 047, 048, 065, 066, 083, and 084).





7.8.75. QUEUE

The QUEUE command provides board level flow control for the execution queue. The entire buffer can be re-executed with the QUEUE AGAIN command, or the buffer can be made to wait for a specified input condition.

When the queue buffer is full, the U500 will not accept additional commands until space is available.

SYNTAX:

QUEUE AGAIN

QUEUE CANCEL

QUEUE INPUT *num,val...*

QU IN,*$nn,value*

QU IN,*$nnb,value,$nnb,value,...*

QUEUE AGAIN	Directs program flow back to the top of the queue buffer, and repeats the entire command set.
QUEUE CANCEL	Cancels the AGAIN case, decodes the next command.
QUEUE INPUT *num,val*	Causes the system to wait for a specific signal on an input line before processing the next command.
num	Designates the input number (0-F).
val	Specifies the input value (0/1).
QU IN,*$nn,value*	Accesses I/O on iSBX expansion card and evaluates value of entire port.
$nn	Address of SBX-IO48.
value	Value of bit pattern to wait until.
QU IN,*$nnb,value,$nnb,value,...*	
	Accesses I/O on iSBX expansion card and evaluates individual bits of iSBX port.
$nn	Address of SBX-IO48. Must be the same for all arguments within command.
b	Bit number to check.
value	Value of bit number (0 or 1).



The Abort function is used to clear the queue buffer.

EXAMPLES:

1.

```
EN X
HOME X
WAIT ON
ABORT                ;Clear queue
X100 F1000
DW 1000
X-100
DW 1000
QU AG                ;Proceed to the top of the queue buffer, then repeat the
                     ;entire process
```

2.

```
QU IN,0,1,1,1        ;Waits until input bits 0 and 1 are "1" before execution
                     ;continues with the next command in the queue
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

ABORT, HALT, PLANE, TRIGGER, SD, SA, LINEAR



7.8.76. RAMP

The RAMP time command is the time that it takes each axis to change from the current velocity to the new velocity. RAMP time is used only for contour motion (X, G1/G2/G3, LINEAR, CW, CCW). This command sets the ramp time for the current contour plane only.

This command overrides, but does not change general parameters 019, 037, 055, and 073 (*Plane n contour ramping time*).

SYNTAX:

RAMP *time*

RA *time*

time	Time in milliseconds, ranging from 1 to 32,000 ms.

EXAMPLES:

RAMP 300	;Sets ramp time to 300 msec
RAMP V10	;Sets ramp time to value of variable 10

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

TRAJECTORY, LINEAR, CW_CIRCLE, CCW_CIRCLE, G2, G3



To ensure smooth motion make certain that (steady velocity)/(ramp time) <= (max. accel/ decel rate).

7.8.77. REFERENCE



This command moves the specified axes from the current postion to the marker position. It is similar to a home cycle, except, a limit/home switch is not used. Once the marker (or resolver null) is found, the home offset move is executed and the hardware/software positions are cleared.

SYNTAX:

REF *axis…*

axis =X[YZU].

EXAMPLE:

REF XYZU

Related Commands:

HOME, SOFTWARE HOME, G92

7.8.78. RETURN



The RETURN command is used to signal the end of a subroutine and to direct the program flow back to the program block that follows the block calling the subroutine.

SYNTAX:

REturn

RE

Arguments are not needed with the RE command.

EXAMPLE:

```
:SUB1
  :
  :
RETURN              ;Program flow will return to the caller.
```

C library level.

Related Commands:

SUBROUTINE



7.8.79. ROTATE (Part Rotation)

Part rotation reproduces a parts program at a specified angle. The command can be used to create a circular array. Part rotation begins when the "ROT" command is given with a non-zero rotation angle. All moves are rotated with respect to the point when rotation was turned on. Rotation continues until the ROT command is given with a zero rotation angle.

SYNTAX:

ROT *axis1,axis2,angle*

axis1	First axis to rotate.
axis2	Second axis to rotate.
angle	Angle in degrees.

EXAMPLES:

```
ROT X,Y,45      ; Start part rotation - rotate XY plane by 45 degrees
ROT X,Y,0       ; Turn off part rotation
```



See Chapter 9 for a programming example of the ROT command.

7.8.80. ROUNDING

RO

When performing a contour motion, this command affects the behavior of deceleration. Under normal conditions, when this command option is OFF-G24 (default state), each contour path decelerates to its target position before the next block of motion begins. Refer to Figure 7-10.

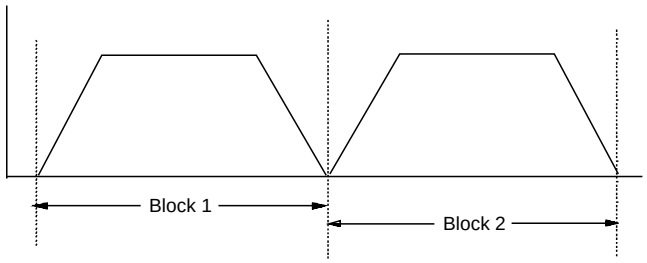


Figure 7-10. Illustration of No Corner Rounding (G24)

When this command option is ON-G23, the next block of motion will begin before the previous path is complete, creating a "rounded corner" at the end of the path. Refer to Figure 7-11.

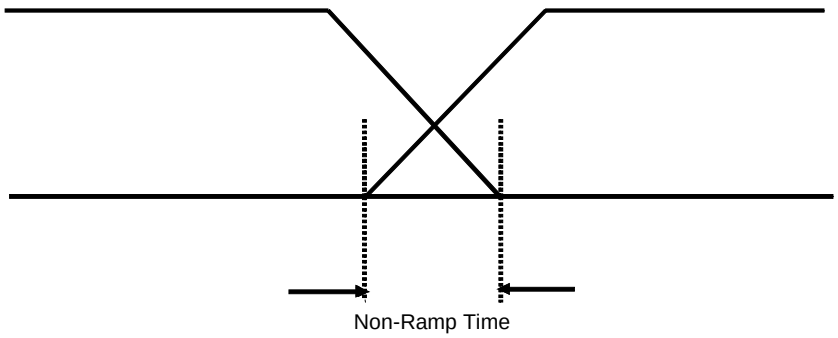


Figure 7-11. Illustration of Corner Rounding (G23)

The time between the path stop and the start of the next block is defined as "non-ramp time" and may also be programmed through this command option. Programming a non-ramp time overrides but does not change the setting of General Parameters 028, 046, 064, or 082 (*Corner rounding non-ramp time*).

Enhanced Corner Rounding

The G25, G26, and G27 corner rounding commands offer an enhanced method of blending linear moves. This mode places a programmed circle between two linear moves (G1/Linear) if they are more than ± 9 degrees out of parallel. The radius of the circle placed between the moves is defined by the G25 command. The G25 command is also used to specify the axis pair and needs to be specified only once in the program. The U500 automatically calculates the circle parameters and adjusts the linear move segments to preserve the programmed path. This operation is totally transparent to the user. If a square is programmed for example, the effect will be to produce a square with rounded corners, with correct dimensions.

The linear move segments must be greater than two times the radius specified with the G25 command, otherwise the U500 will not insert a rounded corner. Velocity profiling mode (G8) can be used with G26 mode. If the linear segments are too small, the U500 will temporarily turn velocity profiling off (G9). It will also do this if a move is programmed with an axis other than those specified with the G25 command.

The contour acceleration command can be used to specify the maximum vector acceleration for G1/G2/G3 moves. If used in conjunction with corner rounding mode, the U500 will limit the acceleration into the corners. This means that the U500 will automatically reduce the vector speed before the corner circle, then speed up after it. The contour acceleration command syntax is "AC PL = value", where value is the acceleration in units/sec/sec or steps/sec/sec. Setting value to 0 turns acceleration mode off.. The U500 then calculates acceleration based on the programmed feedrate and ramp time.

In order to implement this feature, the U500 must know what the "next' motion command is. This is called "look ahead". The U500 will look ahead up to 10 program lines to find the next linear move. The look ahead feature will stop if a program flow statement such as "goto" or "loop" is found, and will not generate a corner rounding move.

SYNTAX:

ROUNDING {*on*|*off*}

ROUNDING *time*

G23

G24

G25 axis_pair **R**raduis

G26

G27

on	G23, activates corner rounding for the specified plane.
	G26, activates enhanced corner rounding for the specified plane.
off	G24, deactivates corner rounding for the specified plane.
	G27, deactivates enhanced corner rounding for the specified plane.
time	Designates the non-ramp time in milliseconds.
axis_pair	axis pair for rounding
radius	radius of the circle placed between linear moves

Make certain the non-ramp time is less than or equal to the ramp time.



EXAMPLES:

EXAMPLE 1: Standard Corner Rounding

```
ROUNDING 100        ;Sets non-ramp time to 100 msec
ROUNDING ON         ;or
G23                 ;Activates standard rounding option
ROUNDING OFF        ;or
G24                 ;Deactivates standard rounding option
```

EXAMPLE 2: Enhanced Corner Rounding

```
CM 1          ; Set contouring mode 1
AC PL=500     ; Set maximum acceleration at 500mm/sec2
EN X Y        ; Enable axes
WA ON         ;
G1 F2400      ; Define feedrate of 2400mm/min
G25 X Y R1    ; Define rounding axes XY with radius of 1mm
G1 Y-5        ; lead in move
G26 G8 X5     ; bottom of square
Y10  ; right side
X-10 ; top
Y-10 ; left
G9 G27 X5     ; bottom, velocity profiling and rounding off
Y5    ; move off of part
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

LINEAR, CW_CIRCLE, CCW_CIRCLE, RAMP



7.8.81. SCOPE

The Scope command tells the scope window to collect one set of data. This is identical to selecting “Collect one set of data” from the Trigger menu of the Axis Scope screen. This command should be put into a user program at the point where data acquisition is to begin. The SC command is downloaded to the U500 and executed synchronously. The number of points to collect, the sample time, etc. should be setup from the tuning screen before running the program.

The scope window must be open for this command to work. Care must be used when programming multiple SC commands or SC commands in a loop. If an SC command is executed while a previous SC command is collecting data, the collection will be restarted and the initial data will be lost.

SYNTAX:

SCOPE

SC

There are no arguments needed with the SC command.

Programming level: Board level.

Related Commands: (none)



This command is only available with the MMI software interface.

7.8.82. SCF (Overriding Scale Factor)



The SCF command can enlarge or reduce a part without a rewrite of the user's program by scaling motions/moves. The programmed distance is essentially multiplied by the overriding scale factor.

The scale factor affects G0, G1, G2, and G3 commands. The commanded positions for all motion commands will be enlarged or reduced by the scale factor. Setting scaling to 1 will effectively disable the scaling command. This command will work in both absolute and incremental moves. The relative and absolute position registers will contain positions as though scaling is turned off. Only the feedback register will be changed to reflect the true (scaled) position of the motor. When using the G2 or G3 commands, the scale value for each of the two axes must be identical, however, the signs may differ.

SYNTAX:

SCF X*xScaleFactor* **Y***yScaleFactor* **Z***zScaleFactor* **U***uScaleFactor*

xScaleFactor yScaleFactor...
Overriding scale factor for the X axis, Y axis etc.

EXAMPLES:

SCF X2 Y.5	;sets scaling of x to twice programmed distance, and y to ;half of programmed distance
SCF X1 Y1	;turns off scaling for x and y axes
SCF Z-1 U1	;produces mirror image

See Chapter 9 for a scale factor programming example.





7.8.83. SKEY (Soft Keys)

The SKey command is used to reprogram the function keys located across the bottom of the MMI screen.

SYNTAX:

SKey SEt *fKey, type, label,' text'*

fKey	Function key to reassign (2 - 9).
type	1 = Goto label, no abort 2 = Subroutine label, no abort 3 = Parts program, no abort 4 = Goto label, abort motion 5 = Subroutine label, abort motion 6 = Parts program, abort motion
label	Label to jump to when function key hit.
text	Text to place on function key should be in quotes.

SKey GEt Waits for a valid function key to be pressed.

SKey ENable *fKeyA(,fKeyB,fKeyC,...)*
Enables a function key.

fKeyA	Function key to enable (2 - 9).
fKeyB,...	Optional function keys to enable.

SKey DIsable *fKeyA(,fKeyB,fKeyC,...)*
Disables a function key.

Arguments same as Skey ENable.

SKey UNdef *fKeyA(,fKeyB, fKeyC,...)*
Returns the function key to the original function as set in the MMI software.

Arguments same as Skey ENable.

EXAMPLES:

SKey SEt *3,1,:cut, 'Cut Part'*
;set key F3 to jump to :cut when hit, button text = ;"Cut Part"

SKey UN *3* ;undefined F3, returns F3 back to original function

C library level.



This command is only available with the MMI software interface.

7.8.84. SLAVE



The slave command is used to link an axis (slave axis) to a feedback channel (master axis). This command is similar to the GEAR command but provides the ability to start tracking based on the UINT_N input. This input can be connected to an external trigger to define when the tracking starts. When the trigger signal occurs, the U500 will sample the encoder feedback channels (within 2 usec). This sampled position will be used as the starting position for the slave axis. The slave axis will then ramp up using a sinusoidal profile to gain synchronization with the master. This is done within the *ramp time* specified in milliseconds.

The slave axis can be scaled to move more or less counts than the master. This is specified by the *slave distance* and *master distance*. These define the slave distance moved as a function of master distance moved. This ratio must not be greater than 127.99. The minimum value is approximately $1x10^{-12}$.

The slave function can be disabled by pressing the abort key or by issuing the SLAVE command with a master axis setting of 0. This will decelerate the axis at the maximum accel / decel rate.

The UINT_N input is an opto-isolated input which generates a hardware interrupt to the U500. The anode of the isolator is OPTOA, the cathode is UINT_N. When the diode is forward biased, a negative edge triggered interrupt will be generated and the gear reference position will be taken. This is true if bit #0 is set in the *flags* argument. If bit #0 is not set, the slave function will engage when the command is given.

The *max slave travel* argument will ramp the slave axis to a stop at the max ac/de rate when the travel has been reached. This argument is relative to the location that the gear function was triggered.

SYNTAX:

"SLAVE *flags, slave axis, master axis, slave distance, master distance, ramp time, [[max slave travel, sync delay, output bit]]***"**

Arguments:

Flags The flags argument allows the user to adjust the operation of the slave command. Each bit has a different function. For most cases the flags parameter can be set to 0. Values can be entered in either hex format or in decimal format.

Note that multiple bits can be set by adding the values from the hex or decimal columns.

Table 7-20. Setting for the 'flags' argument of the Slave Command

bit #	Hex	Decimal	Description (when bit is set)
0	0x0001	1	start slave function when UINT is forward biased
1	0x0002	2	Reverse direction of slave axis motion so that it is opposite from the masters.
2	0x0004	4	slave distance and master distance are in machine steps not units
3	0x0008	8	"queue" buffer for the current plane will execute the next command in the queue, if present, when the slave axis is synchronized with the master. Otherwise execution will not continue until the slave has moved the "max slave distance" , if specified, or abort command is given. The slave command, with master axis set to 0, can be issued in a different plane to stop the axis from moving. This will also allow execution to continue in the original plane.
4..23			reserved, set to 0.

slave axis: Axis to be moved based on the master axis position. Specified as XYZ or U.

master axis: This axis generates the command to the slave axis. Specified as XYZU. 0 can also be specified to un-link the axes and ramp the slave axis to a stop.

slave distance, master distance: Defines how far the slave axis will move for every "master distance" increment. These numbers determine the effective ratio of slave motion to master motion. These arguments are interpreted in the current programming mode ; program steps or units. If bit #2 is set, these numbers are interpreted as machine steps.

ramp time: This is the time in milliseconds that it takes the slave axis to achieve synchronization with the master.

max slave travel: Defines the distance that the slave axis will move with respect to the location in which the slave function was first enabled. When this distance is reached, the axis will ramp to a stop at the max ac/de rate. This parameter may be omitted or set to 0 if the user wishes to stop the slave axis by other means. This number is in current programming units.

sync delay: The slave is considered synchronized with the master after the ramp time and after the sync delay (in milliseconds). This allows any slave position error caused by the acceleration to settle out. This argument can be set to 0 for no delay.

output bit: Output bit to set after the ramp and sync delay. This argument should be set to 0 or 1-8. "0" will cause no change

in the outputs. 1-8 will set the output bit to a logic 1 after the sync delay and logic 0 when the max slave travel has been met (if specified).

Note: Programming a logic 1 causes the output bit to be pulled to ground (active low). Programming a 0 forces a high impedance state on the output bit .

EXAMPLE:

```
"SLAVE 1,X,Y,1000,1000,100"      ; start slaving on UINT
                                 ; input
"SLAVE 0,X,0"                    ; ramp axis to stop
```

Example 1:

Link X axis (slave) to Y axis (master) position. Scaling is 1:1. The X axis is ramp up to meet the speed of the Y axis within 100ms. The slave function will not begin until the UINT_N input is activated. The UINT_N signal could be connected to the Y axis's marker signal for example to start synchronization. When the X axis has traveled 10 units (mm,in,etc), the gearing will disengage and execution will continue. The next command will cause the Y axis to stop its freerun.

```
ENABLE X Y
FREERUN Y1200
SLAVE 1,X,Y,1,1,100,10
FREERUN Y0
```

Example 2:

```
ENABLE X Y
PLANE 1
SLAVE 8,X,Y,999,1000,1
OUT 0,1
PLANE 2
G1 Y100 F1200
SLAVE 0,X,0
```

Programming Notes:

1) Using an effective scale factor which is much greater than 1 may result in audible noise on the slave axis. This is because the master axis will typically have at least +/- 1 count of velocity error as it is moving. The slave's velocity feedback gets multiplied by the effective scale factor and sent to the slave as a velocity command. If the effective scale factor is 10 for example, this +/- 1 count of error will result in +/- 10 counts of velocity command to the slave axis.

2) The slave's position may be adjusted real time by sending a move (G1, etc) while the gearing is enabled. This motion command will add (or subtract) from the slaves position. This must be done in a separate plane if bit #3 of the flags argument is not set.

3) The software position registers will not be correct after the completion of a slaving operation. It is necessary to use the SOFTWARE POSITION command to update the position registers in the PC buffers.

4) This feature requires software version 5.10 or higher and "U500CN.JWP" firmware file.

Using the velocity command filter with the slave command.

When using the filter for velocity command smoothing, the slave's position will lag the commanded position by the following amount…

$$lag = vel / (4 * (1 - e^{-3/(4t)}))$$

where:

"lag" number of encoder counts that the slave axis is behind the command position

"vel" speed of the slave axis in encoder counts / millisecond

"t" filter time parameter in milliseconds.

To compensate for this, an extra delay can be added (sync delay) before setting the output. The additional time needed is calculated as follows…

time delay (msec) = lag (encoder counts) / steady state velocity (encoder counts / msec)

Also the filter must settle to steady state velocity. This takes approximately twice the filter time value.

To calculate the total distance traveled before the U500 will set the output bit…

1)	Acceleration		slave's steady state velocity * ramp time *or* (master axis velocity * slave to master ratio) * ramp time
2)	Sync delay	+	slave's steady state velocity * sync delay time
3)	Filter lag	-	"lag" value from above

EXAMPLE:

It is desired to make the slave axis achieve synchronization with the master within 1 inch. For a stage with .04um resolution, this gives 25000*25.4=635000 encoder counts per inch. The master axis has 4096 counts / revolution and is running at 3850 RPM. It is desired to move the slave axis 960um for every rev of the master axis; 24,000 slave encoder counts for every 4,096 master encoder counts. It has been determined that the slave axis needs 150msec to ramp to the desired velocity.

The steady state slave axis velocity = ((3850/60)*4096) * (24000/4096) = 1540000 steps/sec = 1540 steps/msec

Assuming a filter value of 50msec (time constant), we need at least 150+50*2 = 250 milliseconds for the ramp up and filter settling time. The lag introduced by the filter = 1540 / (4 * $(1-e^{-3/4*50})$) = 25860 encoder counts.

To ramp within 635000 slave counts requires (635000+25860)/1540 = 430 msec. Therefore the sync delay should be set to 430-150 = 280 msec. This is larger than the filter settling time so no additional delay is needed.

SL

7.8.85. SLEW

The SLEW command is used in conjunction with the joystick option to provide immediate axis control. While in the joystick slew mode, the following functions are available:

Joystick A Button	Toggles pairs of axes/drives between *h1 v1* and *h2 v2* as specified.
Joystick B Button	Selects between high velocity, low velocity, and absolute positioning mode. Refer to axis parameters x50 (*Joystick: High speed*), x51 (*Joystick: Low speed*) and x52 (*Absolute mode scale*).
Joystick C Button	Cancels previous joystick command. Same as the SLEW C command.

If more than one contour plane is enabled when the SLEW command is issued, the UNIDEX 500 processes them one at a time. Refer to PLANE for information concerning contour planes.

When slew mode is canceled, the absolute and relative position registers will not reflect the axis positions. They can be updated with the SOFTWARE POSITION command. Refer to the SOFTWARE command description for additional information.

Following use of the joystick to slew the axes to the desired position, the slew mode may be terminated by depressing the joystick's C button (Refer to Figure 7-12). The UNIDEX 500 will resume processing with the next command in the queue buffer.

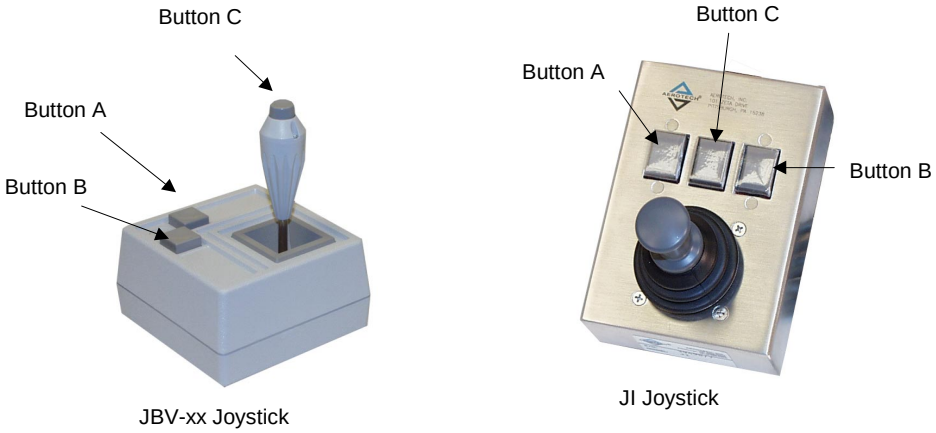


Figure 7-12. Optional UNIDEX 500 Joysticks (JBV Model Left, JI Model Right)

SYNTAX:

SLEW *h1 v1 h2 v2*

SLEW C

h1 v1 h2 v2	Defines 2 horizontal and vertical axis/plane pairs (X, Y, Z, and U) for the joystick.
C	Cancels slew type motion (same as pressing the C button on the joystick).

EXAMPLES:

```
SLEW                    ;No arguments.  Slews all axes as defined below:
                        ;X and Y axes are assigned to plane 1 which is enabled
                        ;Z and U axes assigned to plane 2, which also is enabled
                        ;X = joystick plane 1, horizontal joystick movement
                        ;Y = joystick plane 1, vertical joystick movement
                        ;Z = joystick plane 2, horizontal joystick movement
                        ;U = joystick plane 2, vertical joystick movement
SLEW X Y Z 0            ;Slew the selected axes of the enabled contour plane
                        ;The sequence of the axes (X, Y, Z, U, or 0 for blank)
                        ;determines the joystick's directional relationship:
                          ;X = joystick plane 1, horizontal joystick movement
                          ;Y = joystick plane 1, vertical joystick movement
                          ;Z = joystick plane 2, horizontal joystick movement
                          ;U = joystick plane 2, axis not affected
SLEW 1 2 3 4            ;Slew the selected drives (not axes). The selected drives do
                        ;not need to be within the enabled contour plane.  The
                        ;sequence of the drives (1, 2, 3, 4, or blank) determines the
                        ;joystick's directional relationship
    :
SLEW X Y                ;Enable axis joystick
WAIT ON                 ;Wait for previous command to finish
SOFTWARE POSITION X Y
    :                   ;Update absolute and relative position registers with
    :                   ;adjusted machine position, then resume normal operation
```

Make certain a drive assigned to joystick slew motion has no other motion assigned to it.



UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

WAIT, SOFTWARE POSITION X Y Z U



7.8.86. SOFTWARE

The SOFTWARE setting command is used to establish a variety of software functions. An individual command block must request each of these functions.

SYNTAX:

SOFTWARE HOME *axis_position…*

G92 *axis_position…*

SOFTWARE LIMIT *axis_CCWdistance,Cwdistance …*

SOFTWARE POSITION *axis…*

The functions available with this command are as follows:

SOFTWARE HOME *axis_position* or

G92 *axis_positio*n — The command with no axis designation sets the position register of all active axes to zero. The command, with an axis designation, sets the position registers of the designated axis to zero. An axis designation with a following number sets the position register to that number.

SOFTWARE LIMIT *axis_CCWdistance,CWdistance*



Sets the counterclockwise (CCW) and clockwise (CW) travel limit distance (in program units such as inches, mm, etc.) for the specified axis as referenced from the hardware home position. Must HOME axes before software limits take effect.

SOFTWARE POSITION *axis*

Updates the software position(s) for each of the specified axes from the current hardware position. This position is useful after a freerun or when using either a joystick or handwheel option, so that the new software position is updated to match the current hardware position.

EXAMPLES:

```
G92                 ;Sets position register of all of the axes in the current plane
                    ;to 0
G92 X0              ;Sets the position register of the X axis to 0.  All other axes
                    ;are unaffected
SOFTWARE HOME X1.2 Y3.4 Z5.6 U7.8
                    ;Sets the position registers to the specified values
SOFTWARE LIMIT X-10,5
```

```
                    ;Sets the X axis counterclockwise limit at -10 program
                    ;units from the hardware home
                    ;Sets the X axis clockwise limit at 5 program units from
                    ;the hardware home
SOFTWARE POSITION X Y Z
                    ;Updates the X, Y, and Z axes software position from
                    ;the current hardware position
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Before using the SOFTWARE HOME command, the SOFTWARE POSITION command must be used first.





7.8.87. SPLINE

The UNIDEX 500's SPLINE function refers to the controller's ability to perform cubic spline fitting of multiple successive target positions. The result is a smoother path, with minimal positional disturbances and jerking between points. The command is well suited for non-Cartesian geometric motion.

The cubic splining function is in terms of position versus time for up to four axes at once. The target positions are specified in command lines that follow the SPLINE ON command. These target position specifications look very much like INDEX commands. The UNIDEX 500 looks ahead to two positions to assure path smoothing, so at minimum, two target position specifications are needed to begin proper splining motion.

The target positions contain the axis designators intended to spline X, Y, Z, and/or U, and their position values attached to each axis designator. The path time (T) specifies the execution time for an axes in the block and has a minimum value of .001 (1 msec).

When splining motions are enabled, the controller will not process any other types of motion commands such as INDEX, contoured, or freerun commands.

SYNTAX:

SPLINE {*ON*|*OFF*}

SP {*ON*|*OFF*}

ON	Enables the splining function. Subsequent motions will curve fit.
axis distance	T = segment execution time
	OU = output value at end of move
	L = specify linear interpolation for this block only
OFF	Disables the splining function.

EXAMPLE:

The following example illustrates a three point spline. Incremental mode is assumed.

```
SPLINE ON              ;Spline is turned ON
X3.32 Y4.321 T0.123    ;Incremental motion in 123 ms
X0.332 Y0.555          ;Incremental motion in 123 ms
X1.099 Y0.987 OU0x55   ;Incremental motion in 123 ms. At the end of the motion,
                       ;the output command of 0x55 is executed, turning ON
                       ;output bits 0,2,4, and 6
SPLINE OFF             ;Splining is turned OFF
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

INDEX, CW_CIRCLE, CCW_CIRCLE, FREERUN, G90, G91, G70, G71, PROGRAM

The SPLINE command is available only with the UNIDEX 500 Ultra model.



7.8.88. START

The START command is used to activate planes that are currently under the HALT command.



SYNTAX:

START *plane*

START *wait,plane, ...*

plane	Identifies the planes (1, 2, 3, and 4) that require a start.
wait	Wait until designated plane(s) go into the halt state, then start them.

The START command can be used only to activate planes other than its own.



EXAMPLE:

```
START 1,2          ;Activates planes 1 and 2 if they are on halt
START WAIT,1,2     ;Wait until planes 1 and 2 go into halt state, then start both
                   ;of them.  If either or both planes 1 and 2 are not yet in the
                   ;halt state, WAIT will continue indefinitely
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

WAIT, HALT, TRIGGER, SD, SA, G1/LINEAR



7.8.89. SUBROUTINE

The SUBROUTINE (or SU) command is used to direct program flow to a previously defined label or another program. Variable labels are accepted for branching using the "**:%v**###" syntax. The designated program will be processed and then program flow will return to the program block that follows the SUBROUTINE command. The RETURN command must be included at the end of the subroutine.

SYNTAX:

SUBROUTINE *:label*	Jump to label
SUBROUTINE *program*	Jump to program
SU :%v###	Jump to label specified by variable ###.
:label	Program flow will go to the specified label.
program	Program flow will go to another program. The called program must be identified using the *filename.ext* format. The full path may be specified if the file is located in a directory other than the current directory.
###	Is a U500 variable 0 through 255

EXAMPLE:

```
SU :SUB1            ;Program flow will go to the location of the label :SUB1
                    ;and begin processing the command blocks until a
                    ;RETURN command is encountered
SU C:\PROG\PROG1.PRG
                    ;Program flow will go to file PROG1.PRG and begin
                    ;processing the command blocks until a RETURN is
                    ;encountered
v25 = 700
SU :%v25            ;program execution will jump to label :700
...
:700
ME DI "program jumped to here"
RETURN
```

C library level.

Related Commands:

Return



7.8.90. SYNC

The SYNC command causes queue execution to pause until all corner rounding/velocity profile moves have completed.

7.8.91. Target Tracking Commands (TE, TD, TP)



Target tracking is a real time form of motion profile generation. Real time target positions are sent to the U500 from the PC. The axes then attempt to move to the desired position at the velocity specified. At any time, a new target position can be sent to the U500. The filter parameter is used to implement a first order exponential digital filter. This smoothes starts, stops, and transitions in velocity and direction. Target tracking functions are usually used from the C library interface. The functions are as follows:

SYNTAX:

TE *axis*	Tracking Enable–enable target tracking on single axis (1, 2, 3, or 4). Functions will not enable if axis is in fault condition. Repeating the command can enable multiple axes.
TD *axis*	Tracking Disable–disable target tracking mode on single axis 1-4 and return to normal.
TP *axis,pos,vel,filter*	Target Position–set tracking position for single axis.
axis	Axis number (1, 2, 3, or 4).
pos(machine steps)	Target position for the specified axis. This position is with respect to the hardware home position and is in machine steps.
vel (machine steps/sec)	Maximum speed at which the axis will move to get to the target position.
filter (0-1)	The exponential ramping filter. A value of 1 will produce no ramping effect. Values close to 0 will produce long ramp times. Typical values are .01 to .001.

EXAMPLE:

This example program enables target tracking for the X axis and moves to target position. The program assumes the X axis is mapped to drive 1.

```
ENABLE X            ;
HOME X              ;
WAIT ON             ; Wait for home to finish
TE 1                ; Enable target tracking drive 1
TP 1,1000,1000,.01  ; Set target position–axis will move here
  :                 ;
  :                 ; Add real time desired position commands here
  :                 ;
TD 1                ; Disable target tracking for drive 1
```



7.8.92. TRAJECTORY

The TRAJECTORY command is used when doing contour type motion, to specify whether the acceleration and deceleration ramp type will be linear or sine.

SYNTAX:

TRAJECTORY *type*

TR *type*

type	Where:
LInear	Identifies the acceleration/deceleration ramp as linear type for the current plane.
SIne	Identifies the acceleration/deceleration ramp as inverse-sine type for the current plane.

EXAMPLE:

```
TRAJECTORY LINEAR  ;Accel/decel ramp trajectory type is linear
TRAJECTORY SINE    ;Accel/decel ramp trajectory type is inverse-sine
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

Related Commands:

RAMP, G1/G2/G3, LINEAR, CW, CCW

7.8.93. TRIGGER



This command starts planes 1, 2, 3, or 4 (or any combination of planes) that are currently halted. This is a real time, now queued command.

SYNTAX:

TRIGGER *plane, plane,…*

plane	Plane 1-4.

EXAMPLE:

This example loads commands into planes 1 and 2 of the U500's queue. Then it triggers them simultaneously.

```
PLANE 1
HALT
G1 X1 F1000
G1 X.2 Y.2
…                   ; More commands for plane 1
PLANE 2
HALT
G1 Z1 F1000
G1 Z.3
G1 Z.1 U.2
…                   ; More commands for plane 2
TRIGGER 1,2         ; Start planes 1 and 2
```

Related Commands:

WAIT, HALT

7.8.94. UMFO (Manual Feed Override)



The UMFO command option is used to override the MFO potentiometer setting.

SYNTAX:

UMFO {*OFF*|*ON,feed_rate*}

OFF	Enables the MFO potentiometer.
ON,0-199	Disables the MFO potentiometer and specifies a feedrate from 0-199% of the programmed feedrate.

EXAMPLES:

```
UMFO ON,100     ;Disables the MFO potentiometer and sets feedrate at
                ;100% of programmed value
UMFO OF         ;Enables the MFO potentiometer
```

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.



7.8.95. VAR (Read/Write Variables)

The VAR command is used to read and write user variables to files.

SYNTAX:

VAR OPEN *"filename"*	Opens file for storing and retrieving variables.
VAR READ *#,#,#,...*	Read a variable from the open file.
VAR READ ALL	Reads all 256 variables.
VAR WRITE *#,#,...*	Writes current value of the variable to the open file.
VAR WRITE ALL	Writes the values of all 256 variables to file.
VAR CLOSE	Closes open variable file.
filename	Name of file for storing or retrieving variables.
#	Name of variable where # = 0 to 255.

EXAMPLE:

```
VAR OPEN "c:\u500\example.var"
                    ;opens file c:\u500\example.var
VAR WRITE 23,45     ;writes values of v23 and v45 to var.txt
VAR READ 45         ;reads value of v45 from var.txt
                    ;and sets v45 to this value on the card
VAR CLOSE           ;closes var.txt
```



This command is only available with the MMI software interface.

7.8.96. VELOCITY



The VELOCITY command is used when performing contour type motion to blend consecutive motions into one continuous path. It is a modal command and as such will remain in effect until turned OFF.

If consecutive motion blocks are not geometrically tangent to each other the UNIDEX 500 will modify the path at the junction portion to maintain smooth velocity transitions.

SYNTAX:

VELOCITY *on*|*off*

G8

G9

on (G8)	Enables Velocity Profiling.
off (G9)	Disables Velocity Profiling.

EXAMPLES:

```
G8 G1 X100 F100          ;Enables velocity profiling
G2 X0 Y20 C0,10 F200     ;Blends linear motion to produce this circular motion at
                         ;specified feedrate
G9 G1 X-100 F100         ;Blends linear motion, disables velocity profiling at the end
                         ;of this move


VELOCITY ON              ;Does same thing as above
G1 X100 F100             ;
G2 X0 Y20 C0,10 F200     ;
VELOCITY OFF             ;
G1 X-100 F100            ;
```

The last contour move in velocity profiling must include a VELOCITY OFF (G9) command.



7.8.96.1. Correct Usage and Limitations of the Velocity Profiling Algorithm in Contour Mode 0 (CM0)

The following two plots show the results of running a program first without velocity profiling, then with profiling. The three moves of this program are: G1 X10 F960, G1 X5, and G1 X10.

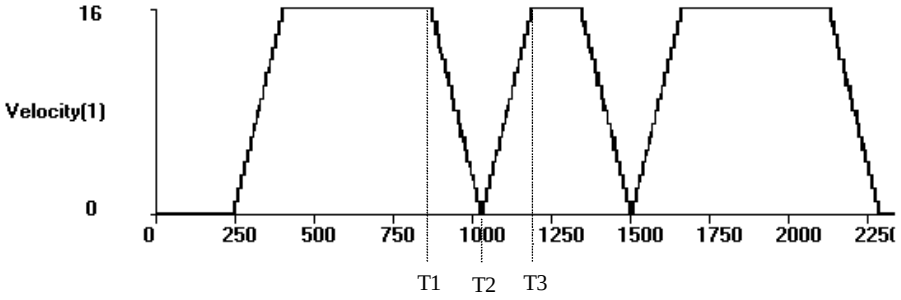


Figure 7-13. Plot of Velocity Without Velocity Profiling

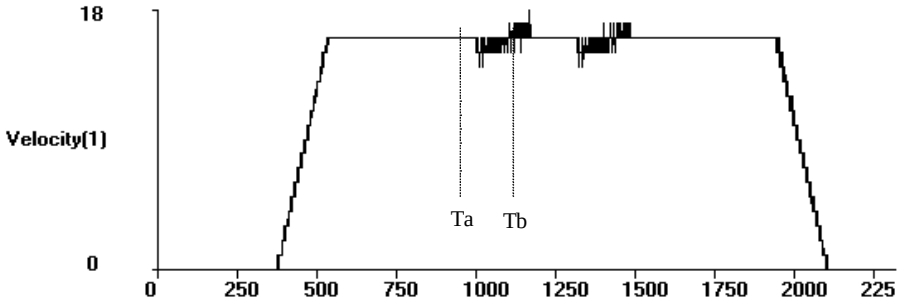


Figure 7-14. Plot of Velocity With Velocity Profiling

The second plot shows how the three moves are blended into one smooth motion. Overlapping the two motion commands by beginning the next move when the current move begins to decelerate generates this velocity. The velocity during this transition between moves (time Ta to Tb) is achieved by adding the velocity of the deceleration of the current move (time T1 to T2) to the acceleration of the next move (time T2 to T3). With ramp times staying constant and distances long enough so that the motion will ramp up to the desired velocity, velocity profiling will produce the desired results.

The following mathematical analysis shows how to determine the shortest move that will still allow for proper velocity profiling. The time of this shortest move is twice the ramp time, and doing the calculations assuming linear acceleration, the acceleration equals the feedrate divided by the ramp time:

$$a = v_F / t_R$$

The distance is the sum of the distances traveled during acceleration and deceleration.

$$x_t = \frac{1}{2} a t_{Ra}^2 + \frac{1}{2} a t_{Rd}^2$$

Substituting in for acceleration yields the simple formula:

$$x_t = v_F t_R$$

For the above example, where v = 960 mm/min = 16 mm/s and with ramp time set at 160 ms, the shortest programmable move for proper velocity profiling is 2.56 mm.

Changing the second move in the above example from `G1 X5` to `G1 X1`, demonstrates the problem that can occur. See the following plots. The first plot shows the motion without profiling, the second plot shows what happens when the moves are blended together.

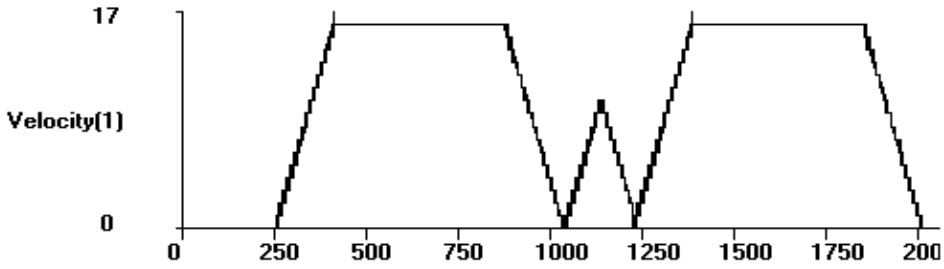


Figure 7-15. Short Middle Move With No Velocity Profiling

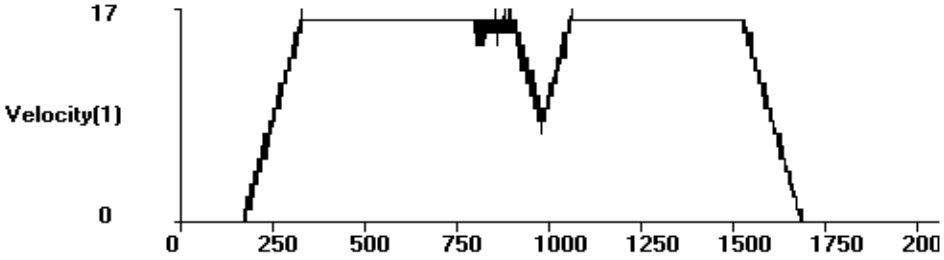


Figure 7-16. Short Middle Move With Velocity Profiling

Because the second move is not able to ramp up to full speed, the profiled velocity is not smooth. One way to remove the dip in the profile is to use contour mode 1, as shown in Figure 7-17. Another way to reduce this dip in the profile is to lower the ramping time of the entire set of G8 moves. This 1 mm move would take about 60 ms to ramp up to the desired velocity. The next plot shows the same motion with the ramping time set to 50 ms.

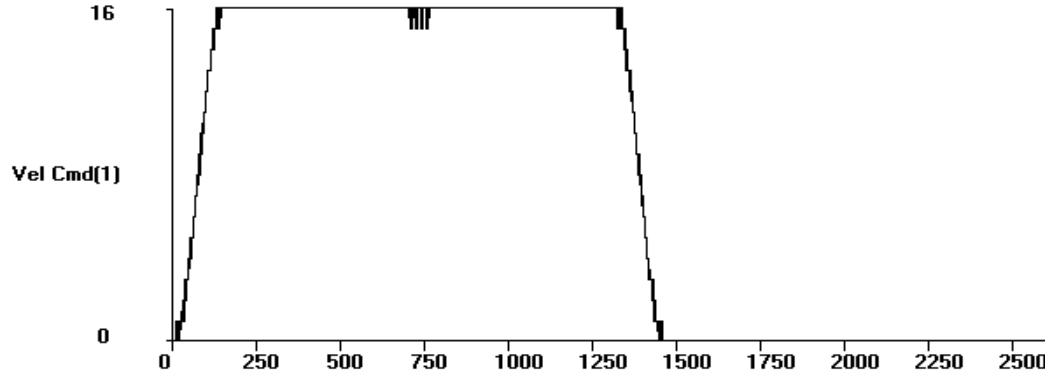


Figure 7-17. Plot Of Velocity With Velocity Profiling In Contour Mode 1.

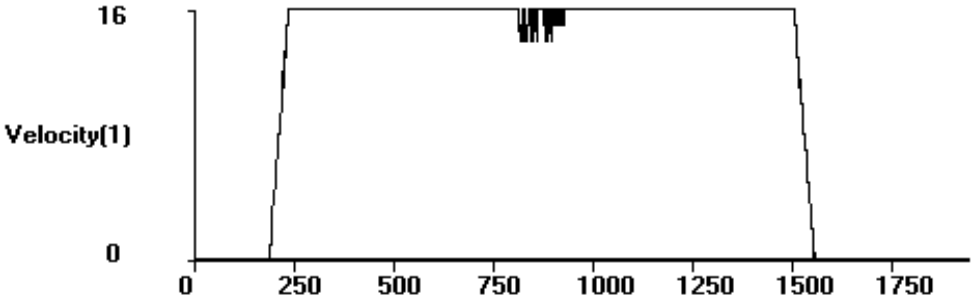


Figure 7-18. Same Motion With Ramping Time Reduced

With two axes in motion, the same effects are present. The next example is velocity profiling a circle using linear moves. The circle has a 5 mm radius and is divided into 500 linear moves. The ramp time is again the parameter to change in order to get the appropriate response. The following two plots show the velocity of the axes, the first plot has a 150 ms ramp time, the second plot has a 5 ms ramp time. Notice that the profiles are similar, but the shorter ramp time allows the axes to achieve higher speeds.

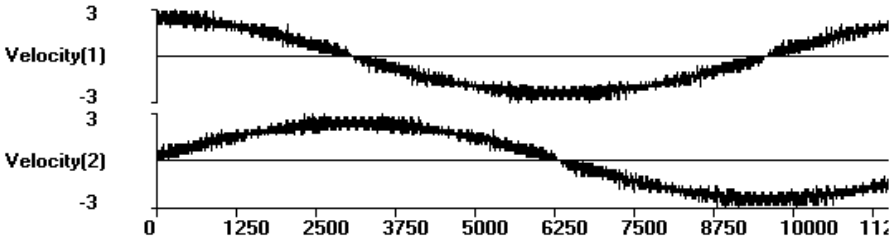


Figure 7-19. Circular Profiling With Long Ramp Time

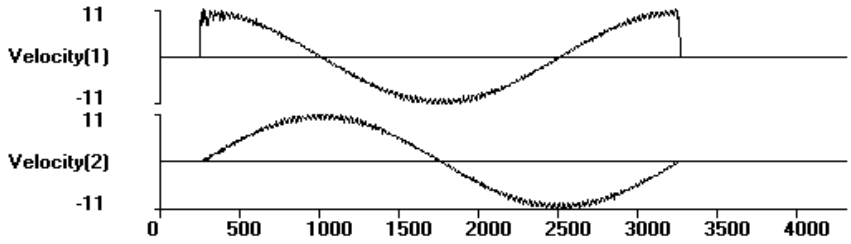


Figure 7-20. Circular Profiling With Short Ramp Time

There are advantages to generating the profile by adding the velocity of the deceleration of the current move to the acceleration of the next move. In some controllers, velocity profiling only works with smooth curves, the UNIDEX 500 allows for profiling of "corners" within the move. Consider the following three moves in G8 mode, `G1 X10, G1 X5 Y5, G1 X10`. The following plot shows the plot of the two axis velocities. The motion is completed with the surface speed staying constant.

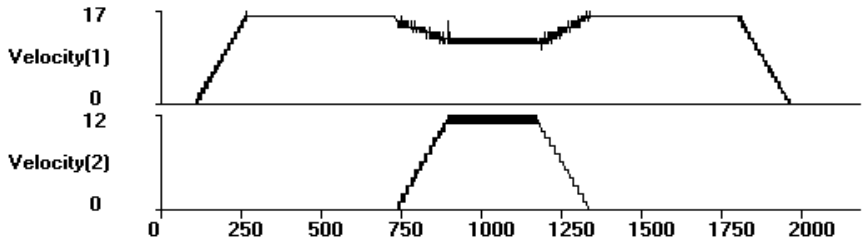


Figure 7-21. Two Axis Linear Move With Velocity Profiling

The constant surface speed is 16 mm/sec. The speed of each individual axis during the X5 Y5 move is determined by breaking the surface speed into its component vector speeds. The equation relating the three velocities together is determine by the Pythagorean Theorem:

$$v_s^2 = v_x^2 + v_y^2$$

In this example, since the moves are the same distance, $vx = vy = 11.3$ mm/sec.

UNIDEX 500 level. Executed by the UNIDEX 500 processor. Bundled and passed by library functions to the controller.

7.8.96.2. CM1 Contouring Mode

The contouring mode can be changed by executing the CM command. The normal mode (CM0) blends moves together by combining deceleration of one move with the acceleration of the next move. The alternate mode (CM1) does not. It requires that the last move be preceded by a G9 (velocity profiling off) command if in G8 mode. The default contouring mode can also be set by general parameters 31, 49, 67, and 85. The previous profile examples assumed CM0 mode. In the following examples, CM1 is assumed. Also shown are the program codes for the motion generating the profiles.

Consider the following program:

```
ENABLE X Y
WAIT  ON
SC
CM 1                    ; SET CONTOUR MODE 1
G8                      ; VELOCITY PROFILING ON
G1 X10 Y1 F10000
X9 Y2
X8 Y3
X7 Y4
X6 Y5
X5 Y6
X4 Y7
X3 Y8
X2 Y9
G9 X1 Y10
```

Velocity Command plots for each axis of this program are shown in Figure 7-22. The plots in Figure 7-22 show how the U500 generates the velocity profile for non-tangential vectors. Note that there is no ramping of individual axis velocities between vectors.

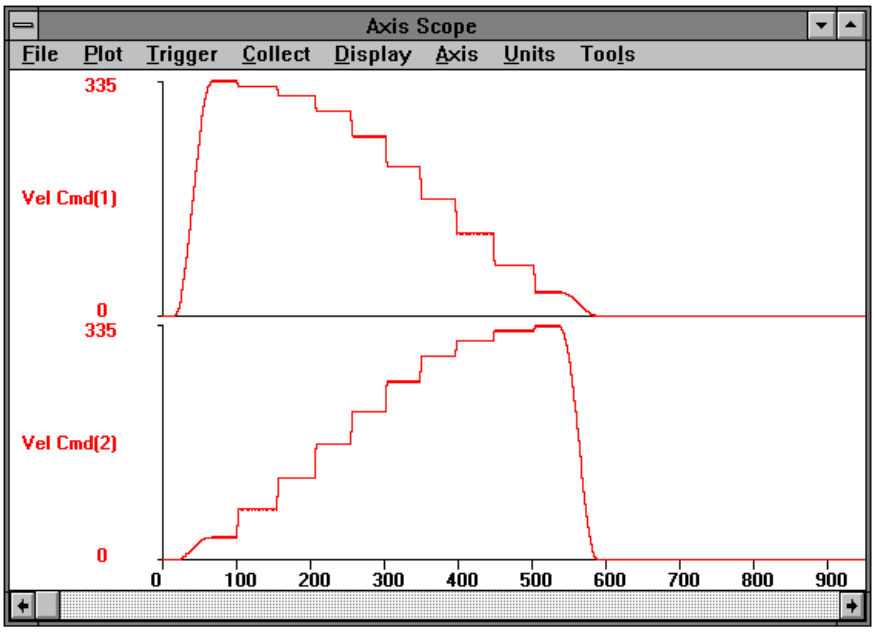


Figure 7-22. Velocity Profile for Non-tangential Vectors

Consider the same program but with a digital filter added through the FL command. A plot for this situation is shown in Figure 7-23.

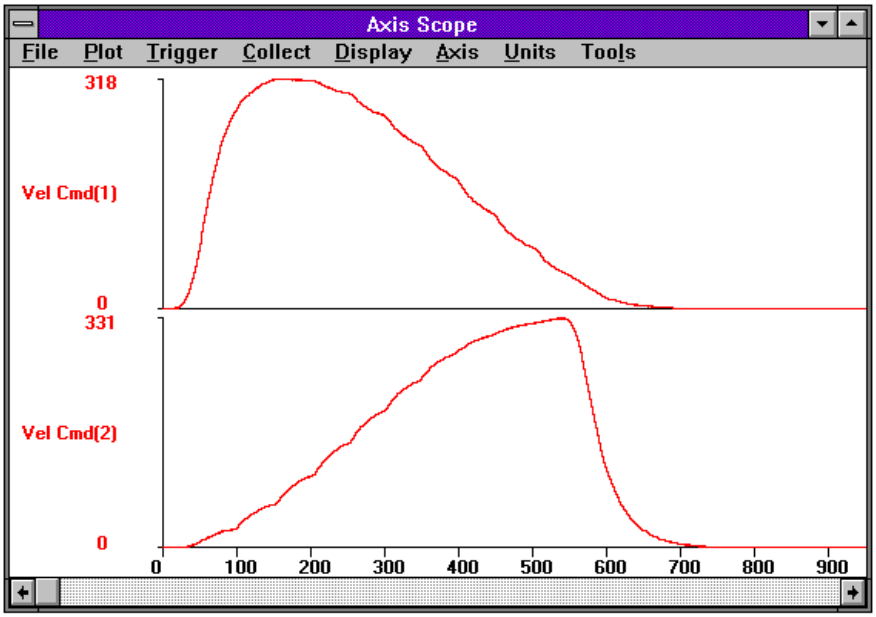


Figure 7-23. Velocity Profile With Digital Filter

This plot shows the same move profile as the preceding plot except that a digital filter has been added with a time constant of 100 ms (see the FL command for more details). This filter smoothes the edges between non-tangential vectors. Although the smoothing occurs it results in some skewing or distortion of the part.

The next program has the G9 command commented out to show what will happen if the last move in a sequence of G8 moves is not a G9 move.

```
ENABLE X Y
WAIT  ON
SC
CM 1                    ; Set contour mode 1
G8                      ; Velocity profiling on
G1 X50 F10000           ; Last move
;; G9 X10                ; Commented-out to show effects of CM1 velocity
                        ; profiling
```

A plot of the velocity command is shown in Figure 7-24.

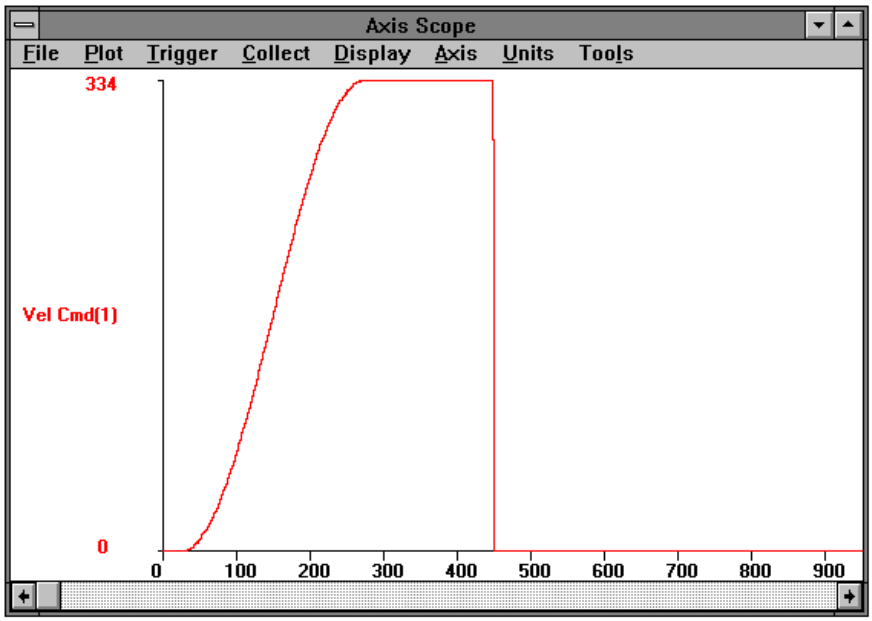


Figure 7-24. Velocity Profile Without a G9 Command at the End of the Sequence

The plot shows that the Velocity Command will immediately ramp to a stop. This will almost always cause an axis fault of some sort and is not considered proper use of velocity profiling mode.

This type of transition can be smoothed by the use of the Filter command (see FL) but the best way is to terminate a sequence of G8 moves with a G9 move.



Related Commands:

INDEX, LINEAR, CW_CIRCLE, CCW_CIRCLE



7.8.97. WAIT

The WAIT command is used to halt the flow of commands to the U500 card until all previous commands in the current contour plane's queue buffer are completed before beginning to receive the next command.

SYNTAX:

WA {ON|OFF|ALL} [[mask]]

WAIT

ON	[[mask]]	Enables the WAIT command. All previous commands are processed before taking next command.
OFF		Disables the WAIT command.
ALL	[[mask]]	Automatically inserts a WAIT ON command at the end of each command block.
mask		Specifies a value that allows you to check specific planes or combinations of planes to see if they are currently busy. 0x11100=wait for plane 1 0x22200=wait for plane 2 0x44400=wait for plane 3 0x88800=wait for plane 4 The mask may be specified to check other status information, such as ENABLED and POSITION. This mask is used internally with the READ_STATUS (S) data. If mask is omitted, it is equivalent to specifying a mask of 0xfff00.

EXAMPLES:

WA ON	;Waits until completion of all previous commands
WA OF	;Disables the WAIT command
WA AL	;Waits at the completion of every command block

Host level. Executed by the host processor.

Related Commands:

MAP, HALT, START

7.8.98. WHILE/ENDWHILE



The WHILE command evaluates the expression and if true, executes to the ENDWHILE statement. The ENDWHILE statement returns to the WHILE statement and the loop is executed until the expression becomes false.

SYNTAX:

WHile (*expression*)

WH (*expression*)

ENDWhile

ENDW

expression Any valid U500 expression.

EXAMPLE:

```
V0=0              ;Assign zero to variable
WHILE (V0<10)     ;Do while variable less than 10
G1 X10 F1000      ;Move axis
V0=V0+1           ;Increment variable by 1
ENDWHILE          ;Return to WHILE as long as V0 less than 10. Otherwise
                  ;quit
```

Related Commands:

LOOP, NEXT, IF, GOTO

∇ ∇ ∇

CHAPTER 8: U500 PCI PULSE OUTPUT

In This Section:

- Hardware Configuration 8-1
- Pulse Output Programming..... 8-2
- U500PCI/PSO-PC Functionality 8-14

8.1. Hardware Configuration

The pulse output can be viewed on TP22 with an oscilliscope. Here, the signal is active low and uses 3.3V levels. The output polarity can be changed by changing jumper JP12. The default polarity (JP12 1-2) is active low for all outputs. The U500PCI pulse output can be configured as follows.

1. TTL output (P9-44): This is a +5V level output capable of sourcing / sinking 24ma.
2. Open Collector output (P9-43)

   This is an open collector output capable of interfacing to 24V levels and sinking 30ma. The M58 opto coupler should be removed when using this output.
3. Opto-coupled output (P9-47,48,49)

   Several opto coupler options can be provided by Aerotech. These are listed below, in Table 8-1.

Table 8-1. Opto Coupler Options

Option	Device	Max Voltage	Current Sink	Speed
-1	HCPL2601	5V	15ma	10Mhz
-2	6N136	15V	5ma	1Mhz
-3	4N33	30V	50ma	30Khz

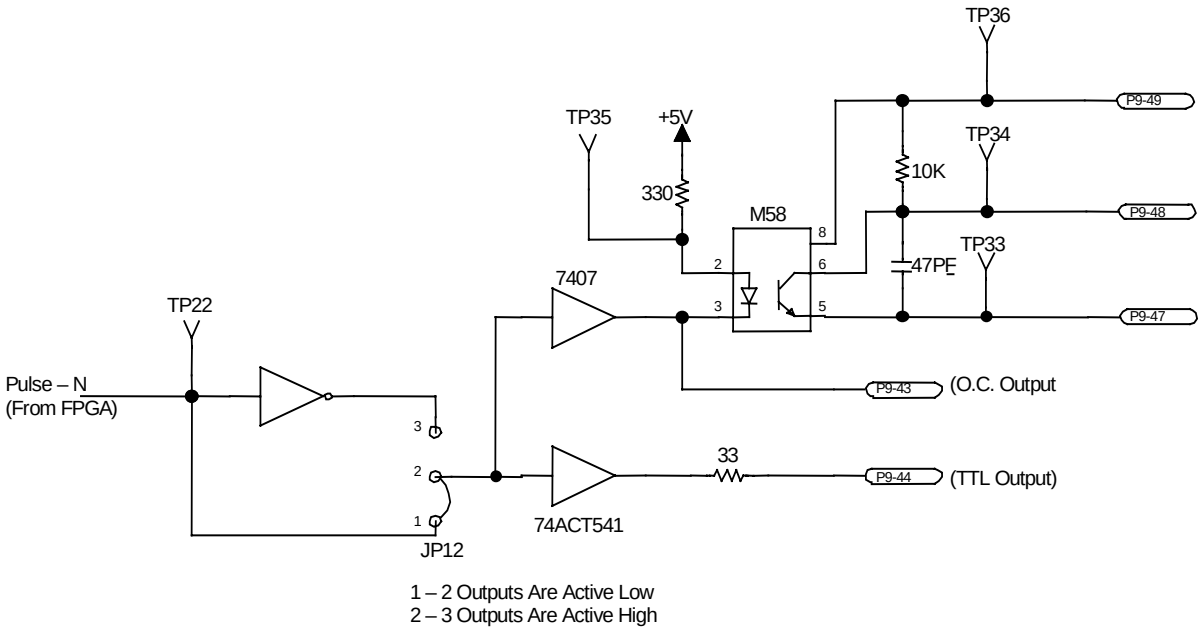


Figure 8-1. Pulse Output Circuit Diagram

8.2. Pulse Output Programming

The pulse output can operate in one of two modes, depending on the FPGA configuration loaded after reset. The single axis firing mode is selected from the "FPGA Setup" tab in the systems options menu of the MMI interface. FPGA#2 should be set to "1 – Single PSO" for the single axis firing mode. It should be set to "4 – Dual PSO" for the dual axis firing mode.

8.2.1. Single PSO Mode

The U500 contains a set of hardware counters which can track the position of a single axis encoder channel. It can then generate an output pulse at a specified number of encoder counts. The output pulse width is user programmable. The U500 also contains two "window" counters and compare registers which can be used to qualify the output pulse. The window counters can be setup to track any axis, independent of the main tracking axis.

The maximum tracking rate (encoder data rate) in single axis firing mode is 20Mhz. The maximum pulse output frequency is 10Mhz. To use the single axis firing mode, FPGA #2 must be set to "single PSO" mode in the system options menu.

The syntax and arguments (Table 8-2) are shown below.

SYNTAX:

"FIRE STEPS XYZU *distance* **PULSE** *width* **ON/OFF ABS/INC WINDOW D2 TRIGGER** *mode* **ARRAY** *index,numpoints* **SET** *mode* **ABORT"**

Table 8-2. Arguments for the FIRE Command

Arguments	Description
X, Y, Z, or U	Select one axis for tracking.
distance	Sets the incremental distance between output pulses (firing distance) in current programming units (English, metric, program steps) or in encoder counts if the STEPS argument is given. The *distance* argument is optional and must follow the axis letter when used. The *distance* argument, when specified, forces an immediate load of the tracking counter. Output pulses will be generated incrementally at positive (and negative) intervals of the distance specified. The firing distance has a maximum value of 8,388,607 encoder counts and should not be specified when using the array mode
PULSE *width*	Output pulse width in usec. The maximum output pulse width is 400000 usec (400 millisec = .4 sec). The minimum output pulse width is .1 usec. Note: 1000 usec = 1 millisec.
ON/OFF	Turns pulse output on or off. The tracking axis, firing distance, and output pulse width should be setup before turning the pulse output on. The OFF command will stop the output pulse after the current pulse completes (if present).
STEPS	Specifies that the distance argument should be interpreted in encoder counts, i.e. scale factor is not used. This argument must be specified before the distance argument. If not specified, the distance argument is interpreted in the current programming mode.
ABS	When firing is turned off with the command FIRE OFF, the fire counter will continue to track the specified axis's position. When firing is re-enabled with the FIRE ON command, the pulse output will remain synchronized to the starting position. Specifying a new distance, however, will cause the counter to load and the reference position will be lost. This command is modal. Once specified it will remain in effect until the FIRE INC command is given. This is the default mode after reset. This command is not the same as the "PROGRAM ABSOLUTE" command.
INC	When firing is turned off with the command "FIRE OFF", the fire counter will reload with the previously specified firing distance. When firing is re-enabled with the FIRE ON command, the fire axis counter will begin counting and pulse outputs will occur at the incremental firing distance. This command is modal. Once specified it will remain in effect until the FIRE ABS command is given or the U500 is reset. This command is not the same as the "PROGRAM INCREMENTAL" command.
WINDOW	Specifies that the main output will be the window function and not the pulse function. The ON/OFF arguments are used to enable/disable the output. Since pulses are not generated incrementally, all other arguments are not used. See the WINDOW command.
D2	Specifies that the encoder will be decoded in X2 mode, not X4 mode. The fire distance will be divided by 2 to preserve the specified distance. This mode is useful if the maximum encoder data rate exceeds the 20Mhz limit of the tracking hardware.

Table 8-2. Arguments for the Fire Command (continued)

Arguments	Description		
TRIGGER *mode*	Allows the firing to be synchronized to the marker output from the encoder.		
	Mode	**Decimal**	**Description**
bit #1..0	00	0	counter is not affected by marker signal
	01	1	counter held at load value until first marker edge. pulse output does not occur until first edge. marker latch should be armed to enable the marker edge
	10	2	counter re-loaded on every marker edge. this is useful for rotary applications
	11	3	reserved
bit #2	0	0	marker latch is not affected
	1	4	arm marker latch (can be combined / added to above cases 1 and 3)
bit #3			reserved
ARRAY *index,numpoints* *	Option argument to tell U500 to use the previously downloaded array data as incremental firing points. The array will be read starting at *index*. The U500 will generate "*numpoints*" output pulses if non-zero. If *numpoints* is set to 0, the u500 will continue through the table until the end. Note that the array data is interpreted in encoder counts, no scaling is performed. If this command is not specified, a fixed distance must be specified.		
SET mode *	The optional SET argument is used to control the loading of the counter register and buffer register. If mode is set to 0, the U500 will load the counter with the first specified array element and the second array element to the buffer register. If mode is set to 1, only the buffer register is updated, the counter is not affected. Note that the count register is buffered so that the next firing distance is immediately available.		
TOGGLE *	This mode generates a continuous pulse output at specified pulse width. The FIRE OFF/ABORT commands turn toggle mode off.		
ABORT	This will terminate the current output pulse immediately and turn off firing.		

* : command only available with software version 5.12 or higher.

EXAMPLES:

```
FIRE STEPS X1000 PULSE 1000 ABS  ; setup fire axis counter to track the X axis
                                 ; 1 pulse generated every 1000 encoder counts
                                 ; pulse output width will be 1msec
                                 ; absolute mode is specified here so that the
                                 ; fire counter is not  reset when firing is
                                 ; manually turned on/off

FIRE PULSE 1000 TOGGLE ON        ; fire continuous pulse train
                                 ; pulse output width will be 1msec
```

```
;*************************************************************************
; Simple single axis firing example.
; Generate incremental output pulse, software enabled
; For use with single axis firing FPGA configuration.
;*************************************************************************
ENABLE X

; turn off window functions - ( this is the default mode )
WINDOW 1 DISABLE
WINDOW 2 DISABLE

; track in X axis
; fire pulse every 1 mm
; output pulse width is 1 msec
; fire counter will not reset when firing is turned off
; specify "INC" if you desire to reset fire counter on FIRE OFF command
FIRE X1 PULSE 1000 ABS
; pulses will occur at exactly the same encoder positions of the x axis
:LOOP
FIRE ON                 ; turn firing on during positive move
G1 X10 F10000           ; will generate 10 pulses
FIRE OFF                ; turn firing off during negative move
G1 X-10
DW 5000
WAIT ON
GOTO :LOOP
```

8.2.1.1. Array Download *

The U500PCI card can store array data for use with the FIRE command. The data resides on the U500 card rather than the PC . This allows fast access to the array for time critical applications. Each array element is a 24 bit integer with a range of +/- 8388607. The array data is loaded through the use of the user variables V0-V255. Only the integer portion of the variable is loaded to the array. The on board U500 array has 65536 locations and is referenced at index 0-65535. . The U500 on-board array is much larger than the number of user variables. It is possible to load the entire on-board array using the “C” functions or by loading 256 values at a time.

SYNTAX:

“ARRAY *uservar,numvar,u500arrayidx* **INC”**

Table 8-3. Arguments for Array Command

Arguments	Description
uservar	specifies the starting index of the first user variable (Vn) to load to the on board array
numvar	specifies the number of variables to download to the U500 on board array
u500arrayidx	specifies the destination index of the U500 on board array
INC	specifies that the data is absolute and should be converted to incremental. It does this by subtracting the Nth point from the N-1th point. The first point is not subtracted. The FIRE command requires that the internal array elements are incremental distances between desired pulse outputs, specified in encoder counts. Therefore if the user creates variables with absolute pulse positions, he should use the INC command to convert to data to incremental form. If the variable array is already in incremental format, the INC command should not be used.

* : command only available with software version 5.12 or higher.

EXAMPLES:

```
ARRAY 0,10,0          ; load data contained in V0..V9 to u500 array starting at
                      ; index 0.
ARRAY 100,50,1000 INC ; load data contained in V100..V149 to u500 array
                      ; starting at index 1000. The data is converted into
                      ; incremental format.
```



The maximum incremental firing distance using the FIRE command is 8,388,607. Negative numbers will result in improper operation.

```
;*************************************************************************
;  Array firing example.
;  For use with single axis firing FPGA configuration.
;*************************************************************************
ENABLE X

;; load variables with incremental firing distance data
V200=0
V201=100
LOOP 100
  V(V200)=V201    ; setup firing distances ( machine steps )
  V200=V200+1     ; increment array index
  V201=V201+10    ; increment firing distance
NEXT

;; load 5000 points to U500PCI on-board array
V201=0            ; on board array index variable
LOOP 50
    ARRAY 0,100,V201    ; load 100 elements at a time
    V201=V201+100       ; increase U500 onboard array index
NEXT

FIRE X PULSE 500                ; define pulse width
FIRE ARRAY 0,4000 SET 0 ON      ; fire 4000 pulses then stop
FR X1000                        ; command motion...
```

8.2.1.2. Window Comparators

The U500 contains two window counters that can be linked to separate axis encoders. These are specified as window number 1 and window number 2. The window logic consist of a 32 bit up down counter (load-able) and two 32 bit compare registers, one for the lower window limit and one for the upper window limit. These are signed numbers and have the same direction polarity as the axis positions displayed in the diagnostic window. When the counter value is greater than the lower limit and less than the upper limit, the counter is "inside" the window. This can be used to enable the tracking of the firing axis. A second identical window function is incorporated to be used in conjunction with the first window to create a two dimensional window.

Both window functions are disabled by default.

SYNTAX:

"WINDOW *window_number* **STEPS AXIS***counter_load* **LOW***min* **HIGH***max* **ENABLE/DISABLE D2 TRIGGER** *mode***"**

Table 8-4. Arguments for Window Command

Argument	Description
window_number	1 or 2
AXIS *counter_load*	XYZU – specifies axis for window tracking and the value to load the window counter with. This value is in current programming units, i.e. English, metric, units, program steps, unless the STEPS argument is present. This argument is a signed 32 bit number with limits of –2,147,483,648 to 2,147,483,647 encoder counts.
LOW *min*	Window low value. When the specified axis's encoder position is less than this value, the axis is considered outside the window. This value is in current programming units, i.e. English, metric, units, program steps, unless the STEPS argument is present. This argument is a signed 32 bit number with limits of –2,147,483,648 to 2,147,483,647 encoder counts.
HIGH *max*	Window high value. When the specified axis's encoder position is greater than this value, the axis is considered outside the window. This value is in current programming units, i.e. English, metric, units, program steps, unless the STEPS argument is present. This argument is a signed 32 bit number with limits of –2,147,483,648 to 2,147,483,647 encoder counts.
ENABLE/DISABLE	Enables or disables the specified window. When the axis is outside the specified window, firing will stop.
STEPS	Optional parameter used to indicate that all following arguments are in encoder counts, not current programming units.
D2	Specifies that the encoder will be decoded in X2 mode, not X4 mode. The counter load value will be divided by 2 to preserve the specified distance. This mode is useful if the maximum encoder data rate exceeds the 20Mhz limit of the tracking hardware.
ABS	When firing is off because the window axis is outside the window limits, the fire counter will continue to track the specified axis's position. When the window axis is inside the window limits, the pulse output will remain synchronized to the starting position. This command has the same effect as the FIRE ABS command This command is modal. Once specified it will remain in effect until the WINDOW INC command is given or the U500 is reset. This is the default mode after reset.

Table 8-4. Arguments for Window Command (continued)

Argument	Description		
ABS (continued)	This command is not the same as the "PROGRAM ABSOLUTE" command.		
INC	When firing is off because the window axis is outside the window limits, the fire counter resets. When the window axis is inside the window limits, the fire axis counter will begin counting and pulse outputs will occur at the incremental firing distance This command is modal. Once specified it will remain in effect until the WINDOW ABS command is given or the U500 is reset. This command is not the same as the "PROGRAM INCREMENTAL" command.		
TRIGGER *mode*	Allows the window to be synchronized to the marker output from the encoder. The marker signal forces the window counter to reset (0) when active.		
	Mode	Decimal	Description
bit #1..0	00	0	counter is not affected by marker signal
	01	1	counter held in reset until first marker edge. pulse output does not occur until first edge. marker latch should be armed to enable the marker edge
	10	2	counter reset on every marker edge
	11	3	reserved
bit #2	0	0	marker latch is not affected
	1	4	arm marker latch (can be combined / added to above cases 1 and 3)
bit #3			reserved



This command has the same effect as the FIRE INC command.

EXAMPLES:

```
WINDOW 1 X0 LOW 10 HIGH 20 ENABLE ; define window 1 between 10mm and
                                  ; 20 mm. set current location to "0"
```

```
;*************************************************************************
;  Window firing example.
;  For use with single axis firing FPGA configuration.
;*************************************************************************
ENABLE X
HOME X
WA ON

WINDOW 1 STEPS X0 LOW4000 HIGH6000 ENABLE ; set up the window
                                  ; between 4000 and 6000 encoder counts,
                                  ; set window counter to 0
FIRE STEPS X100 PULSE 100 ON      ; fire 100usec pulse every 100 steps
                                  ; inside windows
:LOOP
G1 X10 F10000                     ; firing enabled on positive move between
                                  ; 4000 and 6000 counts
X-10                              ; firing enabled on negative move between
                                  ; 6000 and 4000 counts
DWELL 2000
WA ON
GOTO :LOOP
```

```
;*************************************************************************
; This example shows how to use the pulse output of the U500 in window mode with a
; rotary axis.
;
; For use with single axis firing FPGA configuration.
; We want to generate some pulses, equally spaced, at a particular location every
; revolution of the motor. The window function will be used to "gate" on the pulse
; output. The window counter will reset on every marker pulse, once per rev.
; This example assumes that the rotary axis has 4000 counts per revolution.
; A pulse will be generated every 100 encoder counts over 1/4 turn of the motor.
;*************************************************************************
ENABLE X

; setup window to track on axis X axis.
; define window low as 1000 encoder counts, high as 2000 encoder counts
; reset window 1 counter on every marker ( once per revolution )
; fire counter will reset when outside window
WINDOW 1 STEPS X LOW 1000 HIGH 2000 ENABLE TRIG 6 INC

; generate 1msec pulse every 100 encoder counts
; FIRE OFF will not reset fire counter
FIRE STEPS X100 PULSE 1000 ABS ON
FR X1000                                  ; command motion
```

*See the U500 Help File for additional programming examples

8.2.2. Dual Axis PSO Mode

The U500 can track on any combination of two axes and will generate an output pulse at the vector distance specified. The two axes should have the same resolution (encoder counts / in-mm) for the vectored firing distance to be equal in both directions.

The maximum tracking rate of the encoders in dual axis firing mode is 10Mhz. To use the dual axis firing mode, FPGA #2 must be set to "Dual PSO" mode in the system options menu. The U500PCI must also be factory configured to support this option.

SYNTAX:

"DUAL STEPS XYZU *distance* **PULSE** *width* **ON/OFF D1/D2"**

Table 8-5. Arguments for Dual Command

Argument	Description
STEPS	Specifies that the distance argument is in encoder steps, not current units. This argument is optional and must be specified before any other argument if used.
XYZU *distance*	Specifies axes and fire increment in current programming units i.e. English, metric, units , program steps or encoder counts if STEPS argument is given. The distance argument is the incremental firing distance and must be less than 8,388,607encoder counts. Two axes must be specified.
PULSE *width*	Output pulse width in usec. The maximum output pulse width is 400000 usec (400millisec = .4 sec). The minimum output pulse width is .025 usec. Note: 1000usec = 1 millisec.
D2	Specifies that the encoder will be decoded in X2 mode, not X4 mode. The fire distance will be divided by 2 to preserve the specified distance. This mode is useful if the maximum encoder data rate exceeds the 10Mhz limit of the tracking hardware.
D1	Specifies that the encoder will be decoded in X1 mode, not X4 mode. The fire distance will be divided by 4 to preserve the specified distance. This mode is useful if the maximum encoder data rate exceeds the 10Mhz limit of the tracking hardware.
ON/OFF	Turns the pulse output generation on or off.

EXAMPLES:

```
DUAL STEPS X Y 10000 PULSE 1000 on  ; Set up the PSO to track the x and y axes
                                    ; firing a 1 msec pulse every 10000
                                    ; encoder counts
```

```
;*************************************************************************
;  Dual firing example.
;  For use with dual axis firing FPGA configuration.
;  This example tracks the x and y axes during 50 circular motions
;*************************************************************************

ENABLE X Y              ; enable the x and y axis
CM 1                    ; set contour mode 1
WA ON

; Set up the PSO to track the x and y axes firing a 1 msec pulse every 10000
; encoder counts
DUAL STEPS X Y 10000 PULSE 1000 on

; Motion commands - blend consecutive motions into one continuous path
LOOP 49
G8 G2 X0 Y0 I10 J10 F1000
NEXT
G9 G2 X0 Y0 I10 J10 F1000
```

Single PSO Mode

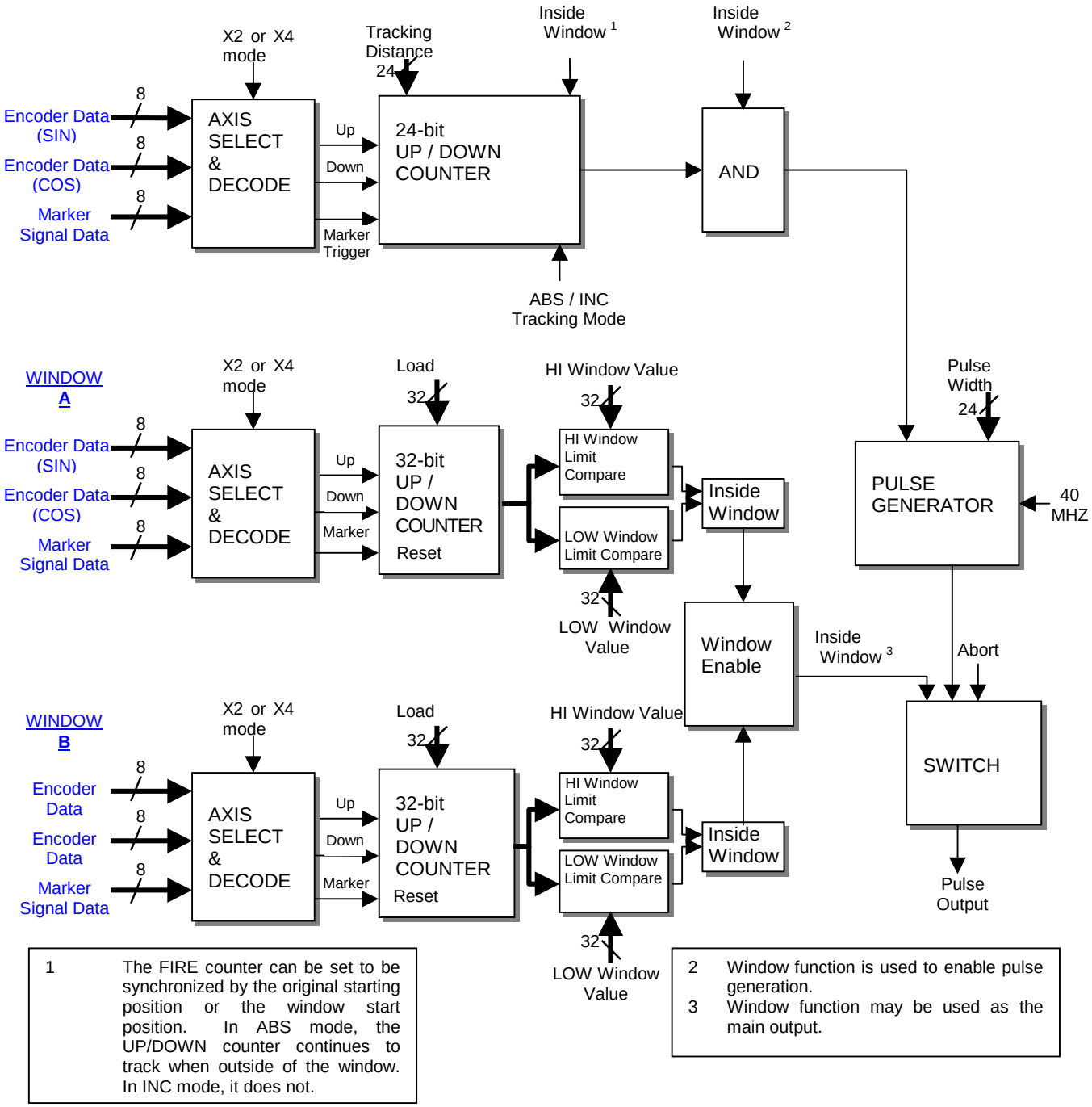


Figure 8-2. Single PSO Mode Block Diagram

Dual PSO Mode

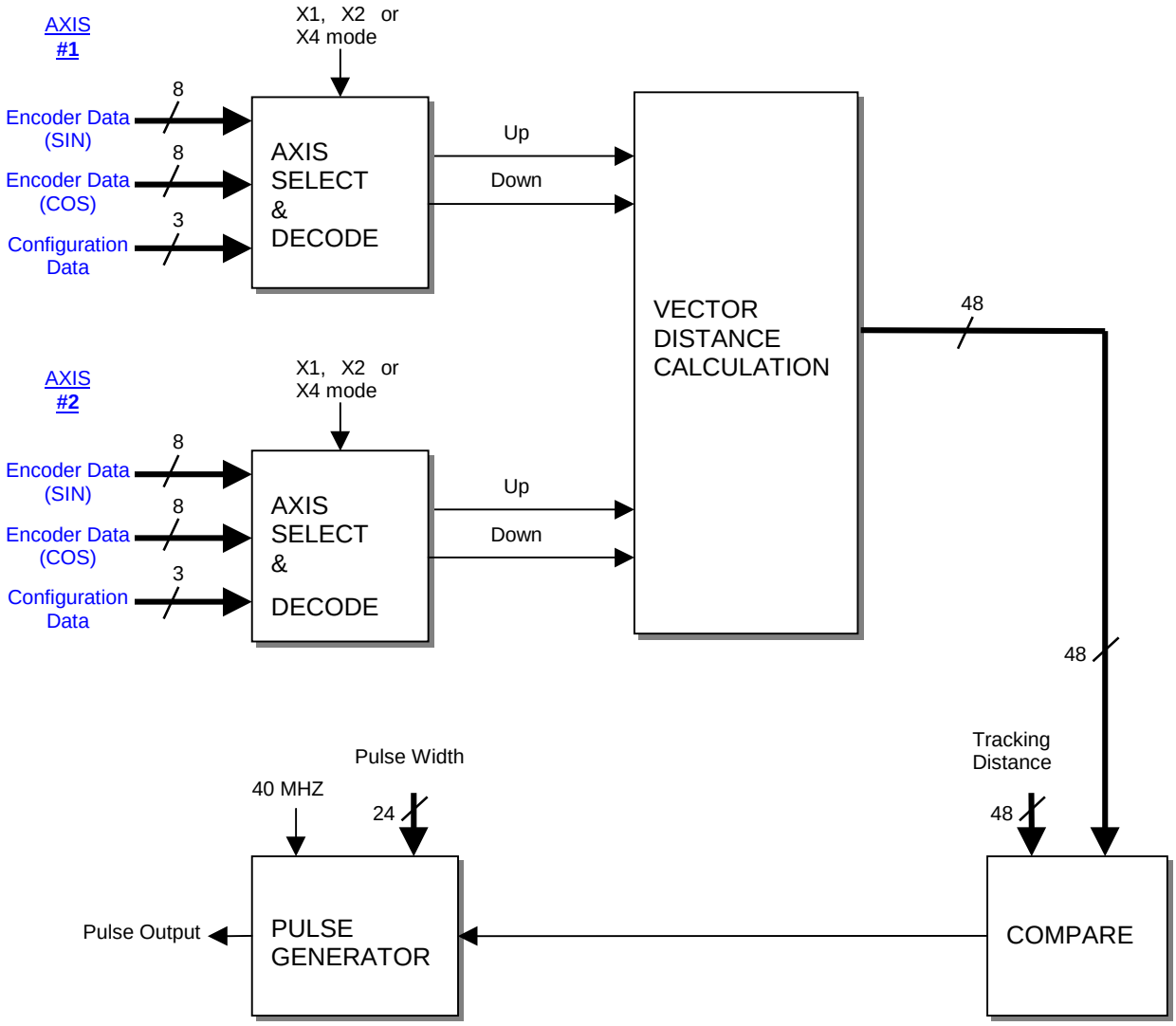


Figure 8-3. Dual PSO Mode Block Diagram

8.3. U500PCI/PSO-PC Functionality

The U500PCI card supports a sub set of the PSO-PC card's functionality. These are listed in the table below. The maximum tracking rate of the encoders in dual axis firing mode is 10Mhz. To use the dual axis firing mode, FPGA #2 must be set to "Dual PSO" mode in the system options menu. The U500PCI must also be factory configured to support this option.

See the "DUAL", "FIRE", and "WINDOW" commands for more information on pulse generation. See the "ANALOG" command for more information on the velocity tracking mode.

Table 8-6. PSO Commands

PSO Command	Description
PSOD,0,*distance*	incremental firing distance (encoder steps)
PSOF,0	firing off
PSOF,1	enable firing – continuous pulse train
PSOF,3,*axis1*	enable firing – single axis (X,Y, Z or U)
PSOF,3,*axis1*,*axis2*	enable firing – two axis (any two of X,Y,Z,U)
PSOP,0,*width*	pulse width in .1 millisecond increments (10 = 1 millisecond)
PSOP,4,*width*	pulse width in 1 microsecond increments (1000 = 1 millisecond)
PSOT,2,*dacnum*,*volts*	analog outputs to DAC9/10
PSOT,4,*dacnum*,*vmin*,*vmax*,*vel*	analog outputs to DAC9/10

Note: Although the command syntax in this table is no longer supported, the functionality is supported through the commands listed in Section 8.2

∇ ∇ ∇

CHAPTER 9: SAMPLE PROGRAMS

In This Section:

- Introduction 9-1
- Incremental (Relative) Motion with Velocity Profiling 9-2
- Absolute Motion with Velocity Profiling 9-5
- CNC Demonstration Using Velocity Profiling, Linear, and Circular Interpolation 9-7
- Corner Rounding 9-8
- GEAR Demonstration of a Master Axis with Two Slave Axes 9-10
- Interlocking Contour Planes 9-11
- Splining 9-12
- Programming Using Inputs 9-13
- Part Rotation 9-14
- Overriding Scale Factor 9-16
- COM Functions 9-19

9.1. Introduction

This chapter is intended to provide an overview of several UNIDEX 500 applications. It is assumed that you have unpacked and checked the U500 system, configured the hardware and software, installed the necessary components, and are otherwise ready to begin using the UNIDEX 500 system in a *real* application.

The application examples and associated programs in this section are intended to give the reader only a general overview of just some of the capabilities of the UNIDEX 500 system. These samples provide some basic fundamentals on which more advanced (and virtually unlimited) applications may be realized.

Additional example programs are located in the u500\mmi\example\prg directory after installation of the UNIDEX 500 MMI or TOOLKIT software.

9.2. Incremental (Relative) Motion with Velocity Profiling

In this application, the UNIDEX 500 is used to *outline* (that is, etch or cut the shape of) a part using two axes (X and Y). The part is outlined using a program consisting of some setup statements and 15 individual movements. The outline shape and the individual movements are illustrated in Figure 9-1. In addition, incremental (relative) coordinates (Δ X,ΔY) are given for the beginning and end points of each movement as well as the home position (0,0). The center points for circular motions are shown as **X**'s with their relative center point coordinates given as well.

Velocity profiling is a programming feature that, when enabled, ensures that the path velocity for the entire shape remains constant. Without velocity profiling, acceleration and deceleration would occur between the paths of individual movements (producing a very *segmented* motion). Velocity profiling is not recommended for motions that make extreme direction changes (for example, 90° turns). Such changes will often cause faults to occur. For this reason, velocity profiling can be temporarily enabled or disabled throughout a program. This is illustrated in the example that follows.

In Figure 9-1, notice that velocity profiling is disabled prior to starting move number 12, and then re-enabled at the end of this movement. This movement begins and ends with sharp (90°) angles (which could cause faults if velocity profiling is used), therefore velocity profiling is disabled during this portion of the part outline. Because velocity profiling is off during motion 12, the path decelerates as motion 11 is completed, and then accelerates at the beginning of motion 13.

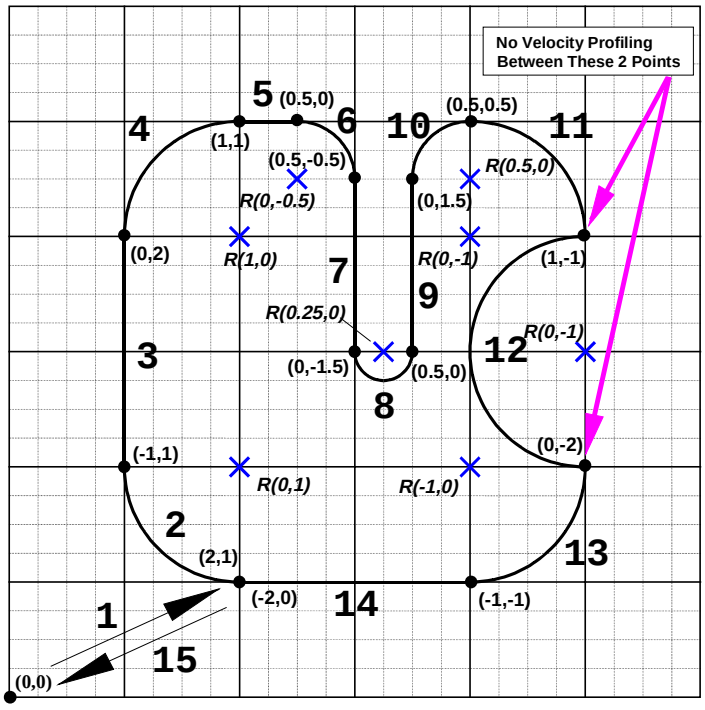


Figure 9-1. Sample Path for Incremental (Relative) Motion Demonstration Using Velocity Profiling

The program used to trace the path in Figure 9-1 (VELOCTY1.PRG) is listed below. Comments have been added for clarity. The program has been rewritten as VELOCTYG.PRG using G codes where possible.

```
;*****************************************************************************************
;       Title:          VELOCTY1.PRG
;       Description:    This program traces a part in incremental mode and
;                       shows velocity profiling.
;*****************************************************************************************
PROGRAM  EN  IN                     ;Use English and incremental modes
HOME X Y                            ;Send axes home
WAIT OFF                            ;Disable the WAIT command
ROUNDING OFF                        ;Disable corner rounding feature
VELOCITY ON                         ;Turn on velocity profiling
INDEX  X2 Y1                        ;1st move - Move from home

MESSAGE DISPLAY "WAITING 5 SECONDS!!!!"     ;Let operator know
DWELL 5000                                  ;we're pausing...

CW_CIRCLE X-1 Y1 C0,1 F100          ;2nd move - CW arc (circle)
LINEAR X0 Y2                        ;3rd move - Vertical (linear) move
CW_CIRCLE X1 Y1 C1,0                ;4th move - CW arc (circle)
LINEAR X.5 Y0                       ;5th move - Horizontal (linear) move
CW_CIRCLE X.5 Y-0.5 C0,-.5          ;6th move - CW arc (circle)
LINEAR X0 Y-1.5                     ;7th move - Vertical (linear) move
CCW_CIRCLE X.5 Y0 C0.25,0           ;8th move - CCW semicircle
LINEAR X0 Y1.5                      ;9th move - Vertical (linear) move
CW_CIRCLE X.5 Y.5 C.5,0             ;10th move - CW arc (circle)
VELOCITY OFF                        ;11th move - Shut off velocity profiling at
CW_CIRCLE X1 Y-1 C0,-1              ;            end of this move (CW arc)
CCW_CIRCLE X0 Y-2 C0,-1             ;12th move - CCW semicircle
VELOCITY ON                         ;13th move - Restore velocity profiling
CW_CIRCLE  X-1 Y-1 C-1,0            ;            then do CW arc (circle)
LINEAR X-2 Y0                       ;14th move - Horizontal (linear) move

MESSAGE DISPLAY "WAITING 5 SECONDS!!!!!!"   ;Let operator know
DWELL 5000                                  ;we're pausing again...
HOME X Y                            ;15th move - Return to home position.
EXIT                                ;End of program
;*****************************************************************************************
;       End of Program VELOCTY1.PRG
;*****************************************************************************************
```

```
;*****************************************************************************************
;       Title:          VELOCTYG.PRG
;       Description:    This program traces a part in incremental mode and
;                       shows velocity profiling using G codes.
;*****************************************************************************************
G70                                     ;Use English mode
G91                                     ;Use incremental mode
HOME X Y                                ;Send axes home
G8                                      ;Turn on velocity profiling
G0  X2  Y1                              ;1st move - Move from home

MESSAGE DISPLAY "WAITING 5 SECONDS!!!!"         ;Let operator know
G4  5000                                        ;we're pausing...

G2  X-1 Y1 C0,1  F100                   ;2nd move - CW arc (circle)
G1  X0  Y2                              ;3rd move - Vertical (linear) move
G2  X1 Y1 C1,0                          ;4th move - CW arc (circle)
G1  X.5 Y0                              ;5th move - Horizontal (linear) move
G2  X.5 Y-0.5 C0,-.5                    ;6th move - CW arc (circle)
G1  X0 Y-1.5                            ;7th move - Vertical (linear) move
G3  X.5 Y0 C0.25,0                      ;8th move - CCW semicircle
G1  X0 Y1.5                             ;9th move - Vertical (linear) move
G2  X.5 Y.5 C.5,0                       ;10th move - CW arc (circle)
G9                                      ;11th move - Shut off velocity profiling at
G2  X1 Y-1 C0,-1                        ;            end of this move (CW arc)
G3  X0 Y-2 C0,-1                        ;12th move - CCW semicircle
G8                                      ;13th move - Restore velocity profiling
G2  X-1 Y-1 C-1,0                       ;            then do CW arc (circle)
G1  X-2 Y0                              ;14th move - Horizontal (linear) move

MESSAGE DISPLAY "WAITING 5 SECONDS!!!!!!"       ;Let operator know
G4  5000                                        ;we're pausing again...

HOME X Y                                ;15th move - Return to home position.
M2                                      ;End of program
;*****************************************************************************************
;       End of Program VELOCTYG.PRG
;*****************************************************************************************
```

9.3. Absolute Motion with Velocity Profiling

In this application, the UNIDEX 500 is used to *outline* (that is, etch or cut the shape of) a part using two axes (X and Y). The part is outlined using a program consisting of some setup statements and 15 individual movements. The outline shape and the individual movements are illustrated in Figure 9-2. In addition, absolute coordinates (X,Y) are given for the beginning and end points of each movement as well as the home position (0,0). The center points for circular motions are shown as **X**'s with their absolute center point coordinates given as well. (Centers of circles are always incremental.) This application is the same as the previous application except that absolute motion is used instead of relative motion.

In Figure 9-2, notice that velocity profiling is disabled prior to starting move number 12, and then re-enabled at the end of this movement. This movement begins and ends with sharp (90°) angles (which could cause faults if velocity profiling is used), therefore velocity profiling is disabled during this portion of the part outline. Because velocity profiling is off during motion 12, the path decelerates as motion 11 is completed, and then accelerates at the beginning of motion 13.

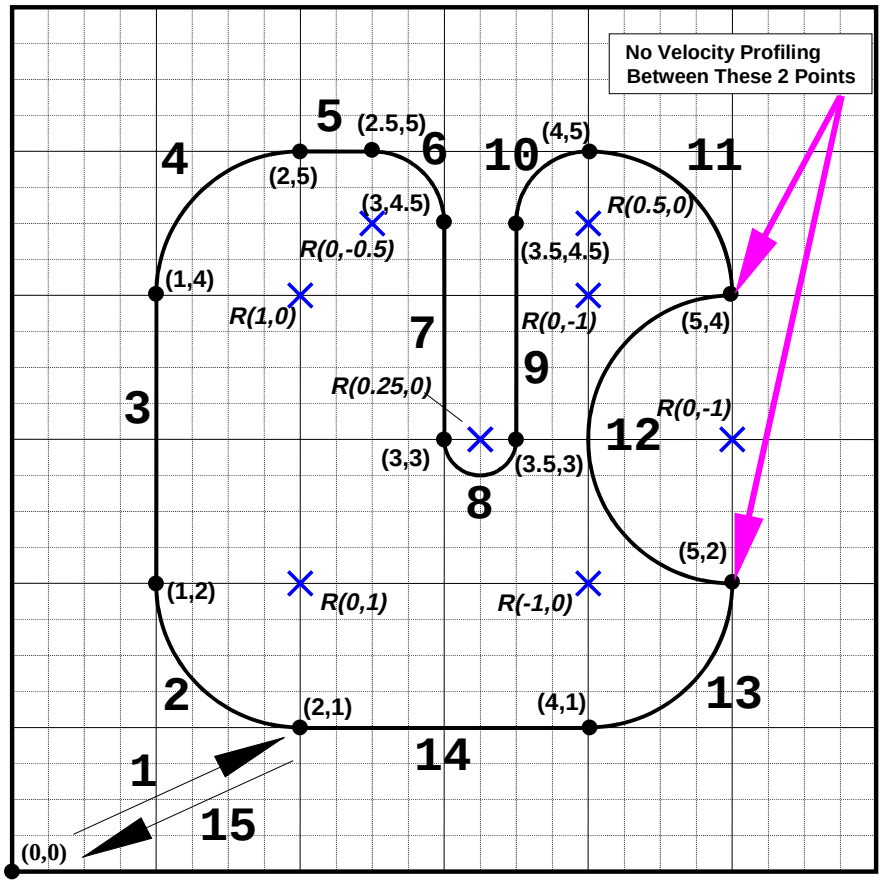


Figure 9-2. Sample Path for Absolute Motion Example Using Velocity Profiling

```
;*****************************************************************************************
;       Title:          VELOCTY2.PRG
;       Description:    This program traces a part in absolute mode and
;                       shows velocity profiling.
;*****************************************************************************************
HOME X Y                                ;Send axes home
WAIT OFF                                ;Disable the WAIT command
ROUNDING OFF                            ;Disable corner rounding feature
VELOCITY ON                             ;Turn on velocity profiling
INDEX  X2 Y1                            ;1st move - Move from home

MESSAGE DISPLAY "WAITING 5 SECONDS!!!!"         ;Let operator know
DWELL 5000                                      ;we're pausing...

CW_CIRCLE X1  Y2  C0,1  F50             ;2nd move - CW arc (circle)
LINEAR X1  Y4                           ;3rd move - Vertical (linear) move
CW_CIRCLE X2  Y5  C1,0                  ;4th move - CW arc (circle)
LINEAR X2.5  Y5                         ;5th move - Horizontal (linear) move
CW_CIRCLE X3  Y4.5  C0,-0.5             ;6th move - CW arc (circle)
LINEAR X3  Y3                           ;7th move - Vertical (linear) move
CCW_CIRCLE X3.5  Y3  C0.5,0             ;8th move - CCW semicircle
LINEAR X3.5  Y4.5                       ;9th move - Vertical (linear) move
CW_CIRCLE X4  Y5  C0.5,0                ;10th move - CW arc (circle)
VELOCITY OFF                            ;11th move - Shut off velocity profiling at
CW_CIRCLE X5  Y4  C0,-1                 ;            end of this move (CW arc)
CCW_CIRCLE X5  Y2  C0,-1                ;12th move - CCW semicircle
VELOCITY ON                             ;13th move - Restore velocity profiling
CW_CIRCLE  X4  Y1  C-1,0                ;            then do CW arc (circle)
LINEAR X2  Y1                           ;14th move - Horizontal (linear) move

MESSAGE DISPLAY "WAITING 5 SECONDS!!!!!!"       ;Let operator know
DWELL 5000                                      ;we're pausing again...

HOME X Y                                ;15th move - Return to home position.
EXIT                                    ;End of program
;*****************************************************************************************
;       End of Program VELOCTY2.PRG
;*****************************************************************************************
```

9.4. CNC Demonstration Using Velocity Profiling, Linear, and Circular Interpolation

```
;*************************************************************************************
;This program is a CNC demonstration.  It uses velocity profiling
;along with linear and circular interpolation.  The part is first cut
;using velocity profiling, then without velocity profiling.
;*************************************************************************************
PROGRAM ENGLISH INCREMENTAL   ;Program in English, incremental mode
ENABLE X Y                    ;Enable the axes
G0 X5                         :Shape is created using
G8 G1 X10 Y10 F100            ;velocity profiling
G2 X0 Y-10 I0 J-5 F200        ;Clockwise semicircle
G1 X-10 Y10 F100              ;Linear move
G9 G3 X0 Y-10 C0,-5 F300      ;Velocity profiling turned off
                              ;at the end of this move
DW 3000                       ;Dwell (wait) for 3 seconds

G1 X10 Y10 F100               ;Create same shape again
                              ;without velocity profiling
G2 X0 Y-10 I0 J-5 F200
G1 X-10 Y10 F100
G3 X0 Y-10 C0,-5 F300
M0                            ;Wait until step key is hit
AGAIN                         ;Repeat program
EXIT
```

9.5. Corner Rounding

In this application, the UNIDEX 500 is used to *outline* a square using two axes (X and Y). The part is outlined using a program consisting of some setup statements and 4 linear absolute movements. The outline shape and the individual movements are illustrated in Figure 9-3. In addition, absolute coordinates (X,Y) are given for the beginning and end points of each linear movement. Rather than incorporating circular motions at the corners of the square, the corner rounding function is enabled. This feature provides an easy way to smooth *sharp edges* or to create *fillets* without having to incorporate circular motions into the program.



For rounded edges that require precise circular contouring, it is recommended that the programmer use the circle commands (CW and CCW) rather than the corner rounding feature.

An illustration of the square outline using corner rounding is shown in Figure 9-3. The associated program listing follows.

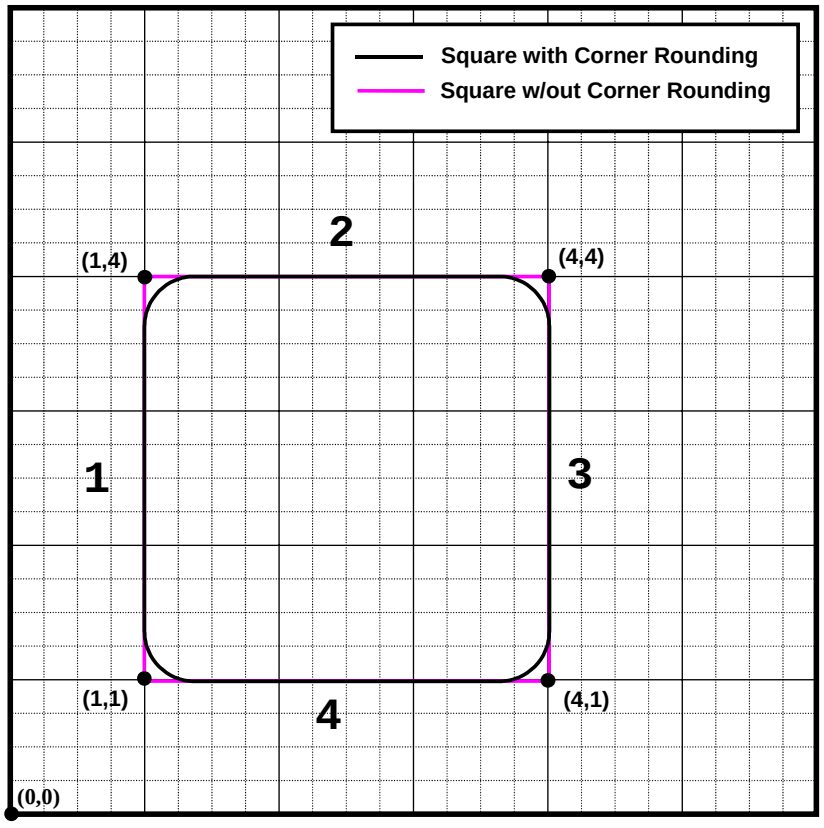


Figure 9-3. Sample Path of Square with and without the Rounding Feature

```
;*****************************************************************************************
;       Title:          CORNER.PRG
;       Description:    This program demonstrates the effects of using the
;                       corner rounding feature.
;*****************************************************************************************
PROGRAM EN  AB                      ;Use English and absolute modes
HOME X Y                            ;Send axes home
WAIT OFF                            ;Disable the WAIT command
INDEX  X1 Y1                        ;Move into position, then prompt the
                                    ;operator for a rounding time (msec).

MESSAGE DISPLAY+V0  "Enter Rounding Time (msec): "

IF  V0<>0  :DOROUND                 ;Do rounding for any non-zero time
ROUNDING OFF                        ;Else, shut off rounding
GOTO :CONT                          ;Continue (merge)

:DOROUND                            ;Perform the rounding
ROUNDING ON                         ;First turn rounding ON
ROUNDING V0                         ;Set the rounding time

:CONT                               ;Do the square
LINEAR  X4  Y1                      ;1st move
LINEAR  X4  Y4                      ;2nd move
LINEAR  X1  Y4                      ;3rd move
LINEAR  X1  Y1                      ;4th move

ROUNDING OFF                        ;Done.  Turn off the rounding feature.
DWELL 2000                          ;Pause
HOME  X  Y                          ;Return to home position.

EXIT                                ;End of program
;*****************************************************************************************
;       End of Program CORNER.PRG
;*****************************************************************************************
```

The non-ramp time may be included as part of the ROUNDING command or may be set through general parameters 028, 046, 064, and 082 (the Corner Rounding Non-ramp Time parameters). Programming a non-ramp time through the ROUNDING command overrides, but does not change the settings of general parameters 028, 046, 064, and 082.

The non-ramp time value that you specify (either in the program or through the appropriate general parameters) is proportional to the amount of corner rounding that will take place. For example, a small rounding time will yield parts with very slightly rounded corners. A large rounding time will yield parts with a more pronounced amount of rounding. Since other factors such as acceleration/deceleration times, feedrate, etc. determine the extent of rounding, it may be necessary to experiment several times before the desired amount of rounding is achieved.

9.6. GEAR Demonstration of a Master Axis with Two Slave Axes

This demonstration shows how the GEAR command is used. In this example, two slave axes are controlled from a single master axis. Gear ratios for the slave axes and the master axis are entered as variables (in machine counts). The machine count values for each axis can have a maximum value of 8,388,608 and can be either positive or negative. If the machine counts for a master/slave axis pair have different signs (for example, the master has a positive number of machine counts and the slave has a negative number of machine counts), then that master/slave pair will move in opposite directions (that is, a CW move of the master axis will produce a CCW move of the slave axis).

Next, the master axis is disabled, allowing it to be turned manually. As the master axis is moved, the slave axes respond based on the master's movement and the respective master to slave gear ratios. The master axis can be enabled and moved automatically through program control if desired. The gear program, GEAR.PRG, follows.

```
;*****************************************************************************
;       Title:          GEAR.PRG
;       Description:    This program demonstrates the use of the GEAR
;                       programming statement to create a single master
;                       axis that is followed by two slaves axes at user-
;                       definable rates (gear ratios).
;*****************************************************************************
PROGRAM  ME  IN                         ;Use Metric and incremental modes
MESSAGE DISPLAY "The X & Y axes will be slaves to the Z axis."
MESSAGE DISPLAY "You must supply the ratio of Z counts to the other axes."
MESSAGE DISPLAY " "
MESSAGE DISPLAY "Requesting Z to X ratio:  (Z counts to X counts)"
MESSAGE DISPLAY+V0 " # of Z counts"
MESSAGE DISPLAY+V1 " # of X counts"
MESSAGE DISPLAY " "
MESSAGE DISPLAY "Requesting Z to Y ratio:  (Z counts to Y counts)"
MESSAGE DISPLAY+V2 " # of  counts"
MESSAGE DISPLAY+V3 " # of Y counts"
MESSAGE DISPLAY " "
MESSAGE DISPLAY "The X to Z ratio is %.0FV0 to %.0FV1"
MESSAGE DISPLAY "The Z to Y ratio is %.0FV2 to %.0FV3"
MESSAGE DISPLAY " "
GEAR 1,3,V1,V0                          ;Link slave axis 1 (X) with master 3 (Z)
                                        ;using the specified gear ratios
GEAR 2,3,V3,V2                          ;Link slave axis 2 (Y) with master 3 (Z)
                                        ;using the specified gear ratios
ENABLE X Y                              ;Enable slave axes X and Y
DISABLE Z                               ;Disable master axis Z so it can be turned
                                        ;manually
MESSAGE DISPLAY "Z axis is disabled, so you can move it by hand."
MESSAGE DISPLAY " "
MESSAGE DISPLAY "  - Type 'GEAR 1,0,0' to disable X axis"
MESSAGE DISPLAY "  - Type 'GEAR 2,0,0' to disable Y axis"
MESSAGE DISPLAY " "
;*****************************************************************************
;       End of Program GEAR.PRG
;*****************************************************************************
```

9.7. Interlocking Contour Planes

```
;*****************************************************************************************
;   This program will begin motion on the X axis. At the start of deceleration,
;   Plane 2 will be triggered. An output bit will be set. Planes 3 and 4 will
;   then be triggered synchronously. Parameter #0 must be set to 4 to define
;   4 planes.
;*****************************************************************************************

ENABLE X Y Z U
WAIT ON                 ;Wait for enable to finish

PLANE 2                 ;Queue commands to plane 2
HALT                    ;Stop execution in this plane
OUT 0,1                 ;Set output bit
START 3                 ;Start plane 3 when output is done

PLANE 3                 ;Queue commands to plane 3
HALT                    ;Stop execution in this plane
G1 Y2 Z2 F1000          ;Do motion
START 4                 ;Start plane 4 when done with move

PLANE 4                 ;Queue commands to plane 4
HALT                    ;Stop execution in this plane
G1 U2 F1000             ;Do motion

PLANE 1                 ;
SCOPE                   ;Start data collection ( optional )
OUT 0,0                 ;Set output low
G1 X2 F1000 SD2         ;Do move, trigger plane 2 at start of decel
```

9.8. Splining

This program is an example of splining. Refer to Figure 9-4 for the output of the program.

```
;*************************************************************************************
;This is an example of splining.  The part will be cut at a constant velocity.
;*************************************************************************************
        PROGRAM ENGLISH INCREMENTAL
        ENABLE X Y
        OUTPUT 0X00                 ;Sets all outputs low, high z impedance
        SPLINE ON                   ;Turn on spline
        X1  Y3  L  F100             ;L denotes a linear motion, no splining for
                                    ;Motion to this position
        X0.06  Y0.1                 ;Position 2
        X0.14  Y0.2                 ;Position 3
        X0.2   Y0.07                ;Position 4
        X0.2   Y0.03                ;Position 5
        X5     Y-0.2    L           ;Linear move to this position
        X0.1   Y-0.02               ;Position 7
        X0.08  Y-0.08               ;Position 8
        X0.02  Y-0.1                ;Position 9
               Y-3.2    L           ;Linear move to this position
        X-0.02 Y-0.1                ;Position 11
        X-0.08 Y-0.08               ;Position 12
        X-0.1  Y-0.02               ;Position 13
        X-6.2           L           ;Linear move to this position
        X-0.1  Y0.01                ;Position 15
        X-0.13 Y0.04                ;Position 16
        X-0.1  Y0.05                ;Position 17
        X-0.09 Y0.2                 ;Position 18
        X0.02  Y0.1                 ;Position 19
        SPLINE OFF                  ;Turn off spline
        EXIT
```

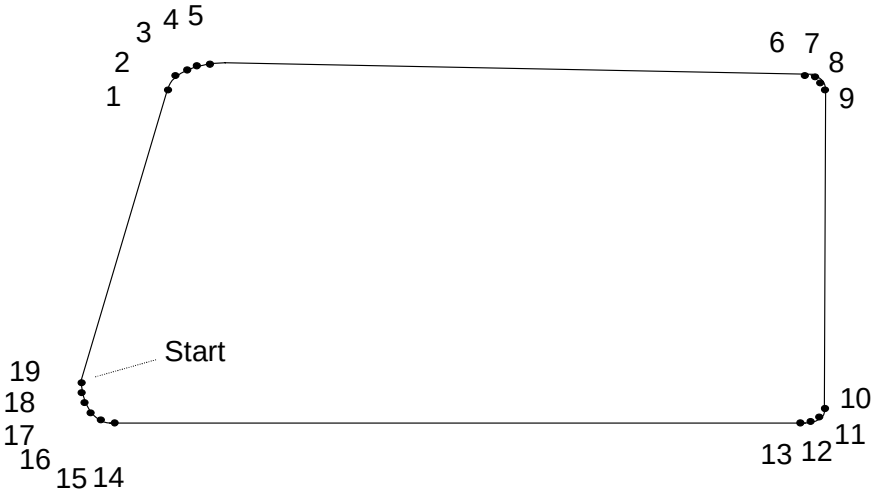


Figure 9-4. Output from Splining Example

9.9. Programming Using Inputs

```
;*************************************************************************************************
;This program is a CNC example.  Motion depends on input bit 0.  The program
;stays in a wait loop until input bit 0 is a 0.  When the input goes low, motion is
;completed and the program loops back and checks the input bit again.  The
;"start button" is assumed to pull input bit 0 low when the button is pressed.
;*************************************************************************************************
WAIT ALL                        ;Wait for each command to finish before
                                ;processing next
ENABLE X Y                      ;Enable the axes
HOME X Y                        ;Send x and y home
G0 X5 Y3 F500                   ;Move x and y to starting point
G92                             ;Set software home, reset axes to 0

:start
MESSAGE DISPLAY "Press start button to begin"

:wait
IF $IN0 = 1 :wait               ;Loops until bit 0 = 0
G2 X-2 Y-2 C0,-2 F100           ;If bit 0=0, complete motion
G0 X2
G0 Y2
MESSAGE DISPLAY "Motion is finished"
DWELL 3000                      ;Wait 3 seconds
GOTO :start                     ;Loop back, wait until bit 0 = 0 to run again
```

9.10. Part Rotation

This program demonstrates the proper use of the ROTATE or "ROT" command. Refer to Figure 9-5 for the output of the program.

```
;*****************************************************************************************
;       Title:          BOAT2.PRG
;       Description     Demonstrates parts rotation by drawing a
;                       boat every 30 degrees.
;                       Program in Millimeters.
;*****************************************************************************************
PROGRAM ABSOLUTE METRIC UNITS UNITS/MIN
ROT X,Y,0                               ; ROTATION CMD (0 for rotation off)
ENABLE X Y
HOME X Y                                ; Home X, Y
G0 X70 Y70 XF1500 YF1000                ; Move to center
G92                                     ; Set all positions to 0
V0=0
LOOP 12

; ############# Boat #################
ROT X,Y,V0                              ; ROTATION CMD (0 for rotation off)
G91                                     ; Incremental mode
G1 X50 F750                             ; Move away from center
G92                                     ; Set all positions to 0

G90                                     ; Absolute mode
G1 X25 F750                             ; Boat deck, default feedrate
                                        ;(1500mm/min)
G1 X20 Y-7.5                            ; Boat end
G3 X17.5 Y-7.5 C-1.25,0                 ; Wave
G2 X15 Y-7.5 C-1.25,0                   ; Wave
G3 X12.5 Y-7.5 C-1.25,0                 ; Wave
G2 X10 Y-7.5 C-1.25,0                   ; Wave
G3 X7.5 Y-7.5 C-1.25,0                  ; Wave
G1 X0 Y0                                ; Boat front
G2 X12.5 Y22.5 C26.5,0                  ; Sail
G3 X12.5 Y0 C25,-11.25                  ; Sail
G3 X0 Y0 C-6.25,-17.5                   ; Sail
G91                                     ; Incremental mode
G1 X12.5                                ; GOTO mast location
G1 Y25                                  ; Mast
G1 X5 Y-1.25                            ; Flag
G1 X-5 Y-1.25                           ; Flag
G90                                     ; Absolute mode
G2 X25 Y0 C-14,-22.5                    ; Sail
G3 X12.5 Y0 C-6.25,-17.5                ; Sail
G3 X12.5 Y22.5 C-25,11.25               ; Sail
```

```
G1 X0 Y0                                ; Return to 0,0
G91                                     ; Incremental mode
G1 X-50 F750                            ; Move back to center
G92                                     ; Set all positions to 0
; ############# Done Boat #################

V0=V0+30
NEXT
EXIT                                    ; End of program
;*****************************************************************************************
;
;                               End of BOAT2.PRG
;*****************************************************************************************
```

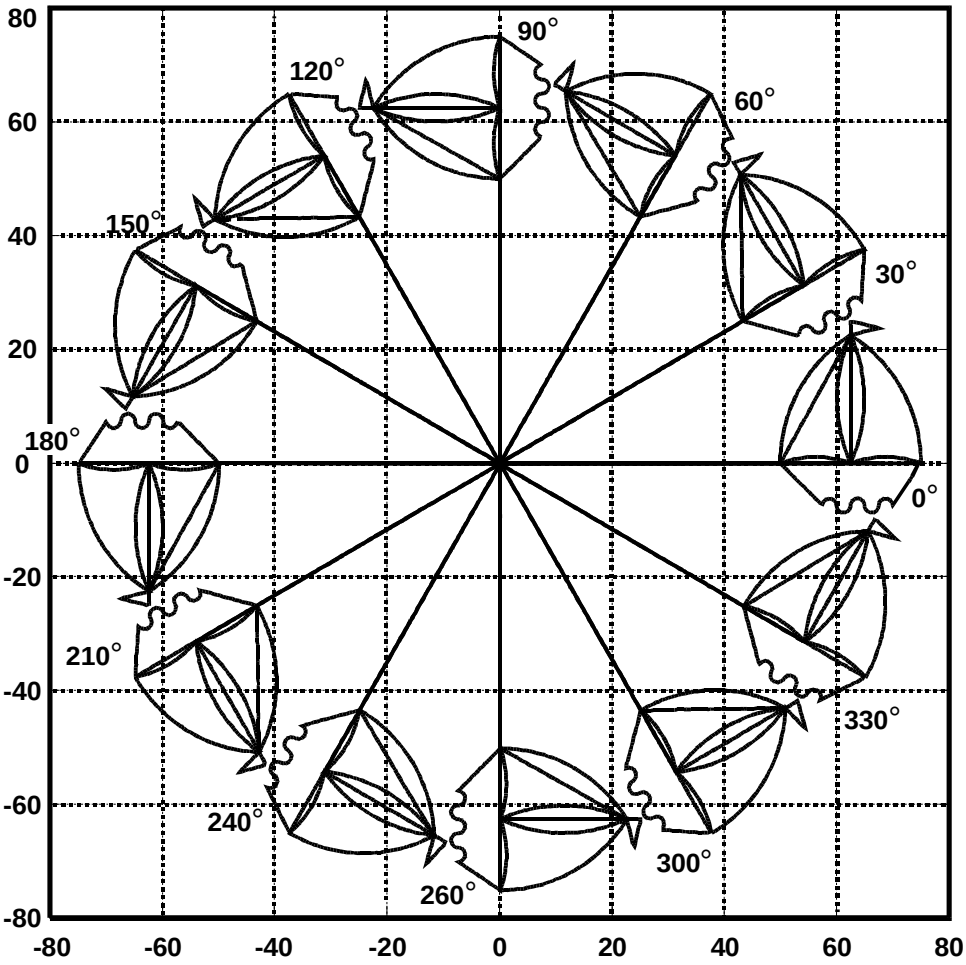


Figure 9-5. Output from Parts Rotation Example Program

9.11. Overriding Scale Factor

This program is an example of the proper use of the SCALE FACTOR or "SCF" command. Refer to Figure 9-6 for a representation of program output.

```
;*****************************************************************************************
;       Title:          BOAT_SCF.PRG
;       Description:    This program demonstrates use of the overriding scale
;                       factor (SCF) command.
;                       Units are Millimeters (Metric).
;*****************************************************************************************
;
PROGRAM METRIC INCREMENTAL UNITS UNITS/MIN
ENABLE X Y
HOME X Y                                ; Home X, Y
G0 X70 Y10 XF1500 YF1000                ; Move to center
G92                                     ; Set all positions to 0

; ######## Make a boat at 1/3 the size ###########
SCF X1/3 Y1/3                           ; Scale Factor Override to 1/3 for X & Y
SUBROUTINE :BOAT

; ######## Make a boat at 64.5% of original size ###########
SCF X0.645 Y0.645                       ; Scale Factor Override to 0.64 for X & Y
SUBROUTINE :BOAT

; ######## Make a boat at original size ###########
SCF X1 Y1                               ; Scale Factor Override to 1 for X & Y
SUBROUTINE :BOAT

; ######## Make a boat at 132.5% of original size ###########
SCF X1.325 Y1.325                       ; Scale Factor Override to 1.3 for X & Y
SUBROUTINE :BOAT

; ######## Traverse to new position ###########
SCF X1 Y1                               ; Turn Scale Factor Override OFF
V0=(1+1.325)*40
G1 Y-V0
G92

; ######## Make a boat at 215.3% of original size ###########
; ######## Boat is a MIRROR Image folded across Y Axis ###########
SCF X-2.153 Y2.153                      ; Scale Factor Override to 2.153 for X & Y
                                        ; (Mirror Image)
SUBROUTINE :BOAT

SCF X1 Y1                               ; Turn Scale Factor Override OFF for x & y
                                        ; axes
DISABLE X Y
EXIT
```

```
; ############# Boat Subroutine #################
:BOAT
G90                                 ; Absolute mode
G1 X25 F500                         ; Boat deck, default feedrate (500mm/min)
G1 X20 Y-7.5                        ; Boat end
G3 X17.5 Y-7.5 C-1.25,0             ; Wave
G2 X15 Y-7.5 C-1.25,0               ; Wave
G3 X12.5 Y-7.5 C-1.25,0             ; Wave
G2 X10 Y-7.5 C-1.25,0               ; Wave
G3 X7.5 Y-7.5 C-1.25,0              ; Wave
G1 X0 Y0                            ; Boat front
G2 X12.5 Y22.5 C26.5,0              ; Sail
G3 X12.5 Y0 C25,-11.25              ; Sail
G3 X0 Y0 C-6.25,-17.5               ; Sail
G91                                 ; Incremental mode
G1 X12.5                            ; GOTO mast location
G1 Y25                              ; Mast
G1 X5 Y-1.25                        ; Flag
G1 X-5 Y-1.25                       ; Flag
G90                                 ; Absolute mode
G2 X25 Y0 C-14,-22.5                ; Sail
G3 X12.5 Y0 C-6.25,-17.5            ; Sail
G3 X12.5 Y22.5 C-25,11.25           ; Sail
G1 X0 Y0                            ;Return to 0,0
G1 Y40                              ; Get ready for next boat
G92
RETURN
;*****************************************************************************************
;                   End of program BOAT_SCF.PRG
;*****************************************************************************************
```

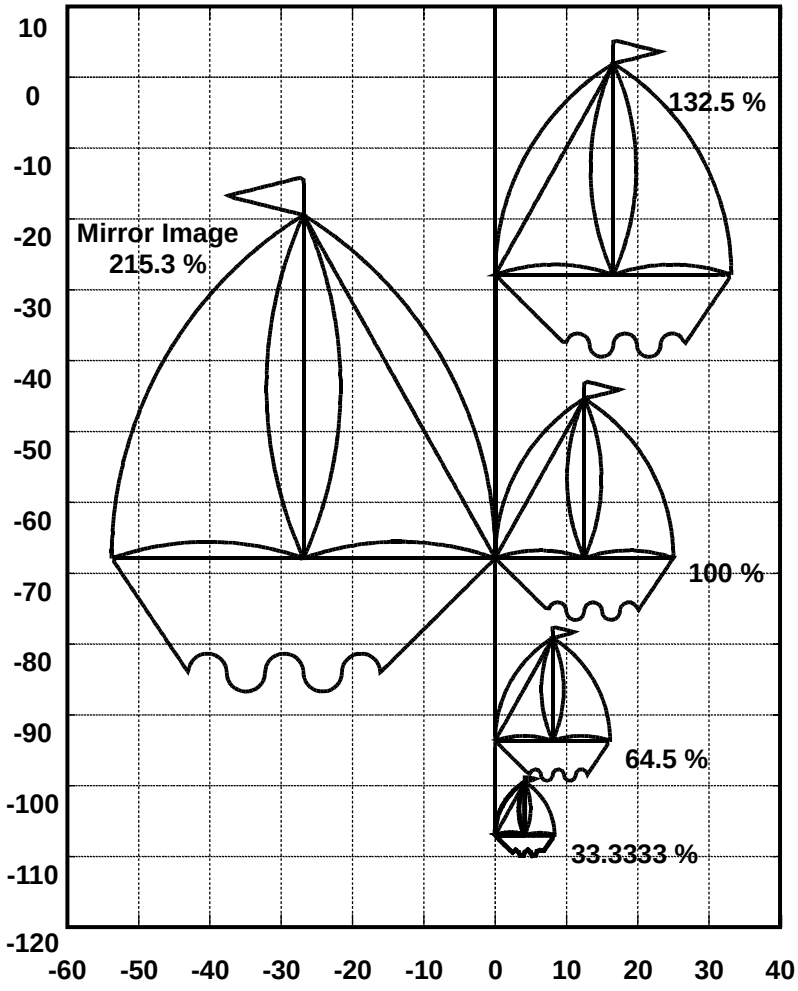


Figure 9-6. Output from Overriding Scale Factor Example Program

9.12. COM Functions

The following is an example parts program segment that uses COM functions.

```
com set _status_ v200              ;Set variable 200 to hold COM status
                                   ;information
com set _terminate_ 13,10          ;Set termination string to carriage return and
                                   ;line feed
com set _timeout_ 5,300            ;COM port time-out is 5 retries with 300 ms
                                   ;between
com init 1,8,1,N,9600              ;Initialize COM port 1, 8 data bits, 1 stop bit,
                                   ;no parity and 9600 baud
if v200 <> 0 :comerr               ;Check for error
com send “ASK POSITION/t”          ;Send the string and the termination seq.
if v200 <>0 :comerr                ;Check for error
com get v100, v101                 ;Wait for two values from COM port 1.
                                   ;Each value must be terminated with the
                                   ; _terminate_ sequence.
if v200 <>0 :comerr                ;Check for error
me di “Positions (%v100, %v102)”   ;Display the results
exit                               ;Done

:comerr                            ;Error routine
me di “COM error V200”             ;Display the error number that was
                                   ;encountered
exit                               ;Done
```

The following is an example of a sample CNC program illustrating how the command "PX1,10.6,5.5" was received through the serial port.

```
;%%%% Serial Port Test Program %%%%
clrscr
com_set _status_ v1
message display "com status = %v1"

com init 2, 8, 1, N, 9600
message display "com init status = %v1"

com set _terminate_ 13, 10
message display "com set _terminate status = %v1"

com send "#Aerotech#/t"
message display "com send status = %v1"

;#### RECEIVING DATA IN FORMAT:  PX1,1000,100  ####
com set _terminate_ 80, 88, 49, 44       ; _terminate_ = "PX1,"
com get v100
com set _terminate_ 44                   ; _terminate_ = ","
com get v101
com set _terminate_ 13, 10               ; _terminate_ = <RETURN> <LF>
com get v102
message display "receive data= PX1,%v101,%v102"
me di "com get status %v1"

com free
```

∇ ∇ ∇

CHAPTER 10: PROGRAMMING TOOLS

In This Section:

- Introduction.....10-1
- Summary of the UNIDEX 500 Quick Library Functions (32bit).....10-2
- Summary of the UNIDEX 500 WAPI Functions (32 bit) 10-31

10.1. Introduction

The UNIDEX 500 software package contains the software programs (Toolkit or MMI) and the library files that will provide an interface between the operator and the UNIDEX 500 system. Toolkit is primarily a startup and diagnostic tool for the UNIDEX 500 system. MMI is a more advanced interface that, among other items, contains all Toolkit features, an extended programming set, and extra features such as joystick digitizing and password options. The operator can also customize the interface to suit the needs of the system by using the library functions.

These library files are provided for C, Delphi and Visual BASIC programs. Different versions of library files are provided for different compiler manufacturers and different compiler versions. Be sure to use the appropriate library file for the desired application. In addition to these programming libraries, appropriate static and dynamic link libraries are also included with the software package. These libraries are listed in Table 10-1.

Table 10-1. Software Development Platforms and Libraries

Platform	Programming Language	Include File	Libraries
Windows 95, NT	Microsoft C	WIN50032.LIB QLB50032.LIB	WIN50032.DLL and/or QLB50032.DLL
	Borland C	WIN32BC5.LIB QLB32BC5.LIB	WIN50032.DLL and/or QLB50032.DLL
	Visual BASIC	WINAER.INC	WIN50032.DLL and QLB50032.DLL
	Delphi		WIN50032.DLL and QLB50032.DLL

The UNIDEX 500 Quick Library functions, presented in section 10.2., provide a low-level programming interface to the UNIDEX 500.

The UNIDEX 500 Windows API functions provide an easy interface that is useful for high level programming in the Windows environment. These functions are presented in section 10.3.

10.2. Summary of the UNIDEX 500 Quick Library Functions (32bit)

10.2.1. Overview

The UNIDEX 500 quick library functions provide a low-level programming interface to the UNIDEX 500. The functions interact directly with the device driver that handles the communication between the operating system and the UNIDEX 500. All of the functions are incorporated in the dynamic link library qlb50032.dll. The functions are ideal for developing faster applications with little overhead using the C programming language. When the quick library functions are used in a C environment, the library file "qlb50032.lib" must be included in the project.

10.2.2. Device Driver Information

In the 32-bit software, the UNIDEX 500 communications are handled using a device driver. In a custom application, the first step is to open the device driver using the aerq_open function. Before exiting the application, the aerq_close function must be called to close the device driver. The name and path of the UNIDEX 500 device driver is dependent on the operating system and the bus type. The device driver is installed during installation of the U500 32-bit MMI/Toolkit software.

U500 ISA DEVICE DRIVER INFORMATION

OPERATING SYSTEM	DEVICE DRIVER	PATH
Windows 95/98	U500ISA.VXD	C:\WINDOWS\SYSTEM
Windows NT	U500ISA.SYS	C:\WINNT\SYSTEM32\DRIVERS

U500 PCI DEVICE DRIVER INFORMATION

OPERATING SYSTEM	DEVICE DRIVER	PATH
Windows 95	U500PCI.VXD	C:\WINDOWS\SYSTEM
Windows 98	U500PCI.SYS	C:\WINDOWS\SYSTEM
Windows NT	U500PCI.SYS	C:\WINNT\SYSTEM32\DRIVERS

10.2.3. Registry Information

The 32bit libraries for the UNIDEX 500 require an entry in the WINDOWS registry file for the ISA and PCI boards. The registry entries, as stated below, are created by the MMI installation software. The user may not need to recreate them.

WINDOWS 95/98:

The WINDOWS 95 registry editor is called REGEDIT.EXE and is located in the C:\WINDOWS directory. A registry entry is made in:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD.

WINDOWS NT

The WINDOWS NT registry editor is called REGEDT32.EXE and is located in the C:\WINNT\SYSTEM32 directory. A registry entry is made in:

HKEY_LOCAL_MACHINE\System\ CurrentControlSet\Services.

The system registry holds time-out and base address information for the UNIDEX 500 ISA card. In the 16-bit software, this information was stored in the configuration file “u500.cfg”.



Refer to the Compiler/Language Requirements section in the QLIB programming help file for more information.



10.2.4. Quick Library Functions

Table 10-2 provides a brief description of the Quick Library functions. For more information, syntax, and examples, refer to:

⇒ Quick Library Programming Help File located in the U500 MMI program group. The file name is /u500/mmi/help/qlib.hlp.

⇒ Example C and Visual Basic code located in the /u500/mmi/example directory

Note that while most functions are supported by the U500 PCI and U500 ISA card, some functions are specific to the card type.

Table 10-2. Quick Library Functions

Function Name	Description	ISA	PCI
aerq_abort	Clears the queue buffer and stops all motion of all active boards. All enabled axes will ramp to a stop using the max accel/decel parameter x16 (Axis Max. Accel/Decel [in machine steps/ms/ms]). The software position registers will then be updated with the new position. This function is similar to pressing the abort key.	√	√
aerq_abs_inc	Set absolute/incremental operation G90, G91. In absolute mode, G90, the motion distance is referenced to the software home position. In incremental mode, G91, the motion distance is an offset referenced from the current position.	√	√
aerq_acceleration	Changes the maximum accel/decel rate for an axis. The function overrides but does not change the setting of parameter x16 (Axis Max. Accel/Decel [in machine steps/ms/ms]).	√	√
aerq_adccontrol	Enables/disables the UNIDEX 500 for a data fetch of A/D values. This function must be called before the user can use the aerq_readadc function to read the A/D values.	√	√
aerq_analog_vector	This function is used to generate an output voltage which is proportional to axis position or velocity.		√
aerq_autotune	This function is called automatically by the U500 MMI as an excitation signal for the autotuning function. This function, however can be also be invoked like any other command to generate a sinusoidal excitation to a specified axis. This can be used for simple frequency response calculations.	√	√
aerq_axis_to_drive	Converts the axis bit number (AXIS1, AXIS2, AXIS3, AXIS4) to an axis drive number 0-3.	√	√
aerq_axis_to_drivebit	Returns the bit value of the axis.	√	√

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_axiserrortoascii	This function converts the axis status value to a text message which can be displayed to the user.	√	√
aerq_axisfilter	Used to set the axis first order filter time constant. This command is used in conjunction with the contouring mode 1.	√	√
aerq_brake	Manually engages or disengages the UNIDEX 500 brake output.	√	√
aerq_checkstatus *	This function updates the status information that is output by the aerq_read_status and the aerq_read_position functions. This function retrieves system status bytes from U500 and fills in following appropriate locations of the board structure.	√	√
aerq_close	Closes the device driver when the user is finished with the UNIDEX 500 board. It also frees any memory that was allocated by the Open Device Driver function.	√	√
aerq_close_param	Frees the memory space where the parameters are stored. If the user wishes to save any parameters loaded in memory, the aerq_save_param function should be called prior to calling this function.	√	√
aerq_contour	Allows the user to perform any of the following contoured move combinations: • 1-4 axis linear • 1 circle • 1 circle + 1 axis linear (helicoil) • 1 circle + 2 axis linear • 2 circles	√	√
aerq_contour_feed	Set vector feedrate in program steps/sec. This feedrate is used during subsequent linear or circular contour (G1/G2/G3) moves that can be called using the aerq_contour function.	√	√
aerq_contour_mode	Sets the current contouring mode. The original contouring mode (CM 0) blends moves together using by combining the deceleration of one move with the acceleration of the next move. The new mode (CM 1) does not.	√	√
aerq_dac	Sets the output of the specified unused DAC channel to a voltage level between –10 and 10 volts.	√	√
aerq_device_driver_msg	Allows a user to retrieve and display a device driver message. Refer to the aerq_set_driver_debug function.	√	√
aerq_disableaxes	Disables servo control of 1,2,3, or 4 axes.	√	√

* See Table 10-3 after the function descriptions for the return values updated by the aerq_checkstatus function that are read using the aerq_read_status function.

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_drive_to_axis	Converts the drive number 0-3 to a qlib axis bit AXIS1, AXIS2, AXIS3, AXIS4.	√	√
aerq_dsplservo	Displays servo loop signals on DAC channels.	√	√
aerq_dual_setup	This function allows the U500 PCI to track on any combination of two axes and will generate an output pulse at the vector distance specified. The U500PCI must also factory configured to support this option.		√
aerq_dwell	Sets a time delay in milliseconds on the board.	√	√
aerq_edit_flash	This function is used to copy the contents of the flash memory to an edit buffer. This function should be called before the using the following functions: aerq_read_fpga_setup_code aerq_write_fpga_setup_code aerq_read_option_code		√
aerq_enableaxes	Enables servo control of the axes.	√	√
aerq_errormask	Changes the system fault masks. The function overrides but does not save the fault mask to the parameter file.	√	√
aerq_errortoascii	Converts the AERERR_CODE error codes to a text message which can be displayed to the user.	√	√
aerq_faultack	Acknowledges fault conditions. If an axis is in a limit, the fault acknowledge moves the axis out of the limit.	√	√
aerq_find_marker	Finds the marker for a specified axis. The move is similar to the move when the axis is searching for the marker after moving out of the home limit during the home cycle. Once the marker is found, the hardware and software positions are cleared.	√	√
aerq_freerun	Executes single axis motion which is unsynchronized to the contouring and indexing functions. The feedrate must always be specified. A feedrate of 0, stops the freerun motion.	√	√
aerq_g92_offset	Executes a software home (g92) command to set the axis position registers to a specified value. The aerq_software_position function must be called before this function to establish a software position for each of the specified axis that is referenced from a current hardware position.	√	√
aerq_gain	Change the servo loop related values of an axis. These values override but do not change the corresponding axis parameter values.	√	√
aerq_gear	Configures an axis to be moved based on the feedback of the master axis. The slave axis follows the motion of the master axis from the time the command is given until the gearing is released.	√	√
aerq_get_diag	Acquires the information displayed in the diagnostics screen.	√	√

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_get_drv_version	Returns the version number of the device driver that is in use.	√	√
aerq_get_firmware_info	Returns the device driver firmware information.	√	√
aerq_get_qlib_version	Returns the version number of the quick library that is in use.	√	√
aerq_halt	This function is used to stop all activity on the current contour plane. When the halt function is called, the UNIDEX 500 retains all commands in an internal queue buffer. These commands are not processed until: 1. the plane is triggered by the LINEARMOVE structure member trig_accel or trig_decel associated with an aerq_contour function call. 2. the plane is activated by the aerq_start function. 3. the plane is activated by the aerq_trigger function. The halt function affects the current plane only.	√	√
aerq_homeaxes	Moves the specified axes to the hardware home position.	√	√
aerq_immed_output	Sets the output bits on either the U500 card or the 4EN encoder card. This function is executed immediately by the UNIDEX 500. It is useful for real time control of the output bus.	√	√
aerq_in_position	This function is used to set a specified output when one or more axes are within a specified error band of their final positions. The axes must meet the following criteria to be considered in position: • axis must be enabled • commanded velocity must be zero • actual feedback velocity must be less than 1 count / millisecond position error must be less than axis parameter x35 (in position dead band)		√
aerq_index	Executes a point to point (G0 / INDEX) move on up to 4 axes. The INDEX command is used to specify point-to-point non-synchronized motion of any or all axes.	√	√
aerq_index_feed	Loads the feedrate on the U500 board for an index move. 1,2,3 or 4 feedrates may be loaded and each should be specified in units of program steps / second. A feedrate of 0 will not change the previous feedrate. This function must be called with at least one non-zero feedrate.	√	√
aerq_init	Initiates a hard reset of the UNIDEX 500 and loads the system firmware and the system parameter file into the PCs memory. Next, this function downloads the parameters to the memory on the UNIDEX 500 board.	√	√
aerq_init_pso	Initializes the PC-PSO board and downloads the PSO firmware file to the board.	√	
aerq_load_array	This function stores array data on the U500 PCI Card for use with the PSO aerq_single_setup function. This allows fast access to the array for time critical applications.		√

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_loadcalfile	Downloads calibration data from the Calibration File (*.CAL) to the UNIDEX 500 board. Axis calibration will be active when: 1. the axis calibration parameter (x15 and/or x71) is set to "yes" 2. the ASCII calibration file is present and has been loaded 3. the axis has been homed (see aerq_homeaxes function) Subsequent axis positioning is then interpolated based on the .CAL file data. A maximum of 2,047 data points of correction are available. The calibration files may also contain orthogonality correction data.	√	√
aerq_lvdt	Links an axis number (1-4) with an R/D channel number (9-16) of the RDP-PC board for very accurate positioning.	√	√
aerq_map	Assigns a drive (0 - 3) to any of the four contour planes, and assigns an axis name (X, Y, Z, or U) to the drive.	√	√
aerq_mem_read	Read a memory location from the DSP's memory.	√	√
aerq_mem_write	Writes "data" to a specified DSP address. The function can: • Overwrite the existing memory data • AND the new data with the previous data • OR the new data with the previous data	√	√
aerq_motor_commu	Sets up AC or brush motors for commutation. This function automatically disables position, velocity, and integral traps and outputs a current (torque) vector that is 90 degrees advanced from rotor vector. This function can also be used to output a constant torque. The motor should spin freely and smoothly in the direction specified.	√	√
aerq_motor_setup	Sets a fixed vector when setting up an AC servo motor. This function outputs a fixed vector current command. The rotor will lock into the commanded position. This function can be used to setup motor phasing by checking the Hall effect states at each point.	√	√
aerq_open	Opens the device driver and allocates memory so that the UNIDEX 500 software can communicate with the UNIDEX 500 board.	√	√
aerq_open_param	Allocates a location in memory and transfers parameter values from a file to that location in memory. After this function is called successfully, all parameter changes are only stored in memory until the aerq_save_param function is called to save the parameters to a file.	√	√
aerq_outmain	Sets or clears individual output bits or writes an 8 bit value to the digital I/O bus. The actual output polarity is the opposite of the programmed polarity.	√	√

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_part_rotate	Configures and executes the part rotation sequence. Part rotation begins with a function call that has a non-zero rotation angle parameter. All moves are rotated with respect to the point when rotation was turned on. Rotation continues until this function is called with a zero rotation angle.	√	√
aerq_pause	Toggles the motion pause feature. The first call to the pause function will stop all motion of all active boards. The next call of the pause function will resume motion.	√	√
aerq_posgrab_disable	This function is used to disable the position capture interrupt.		√
aerq_posgrab_hardware_enable	This function is used to enable position capture interrupt. This should be used when it is known that the position capture will be generated faster than the servo loop update rate. The default servo loop update rate is 4Khz. The position capture interrupt will take precedence over the servo loop update interrupt and servo loop beating with the user- interrupt (UINT) input may occur.		√
aerq_posgrab_read_array_size	This function is used to read the current size of the position capture array.		√
aerq_posgrab_read_index	This function is used to return the current index of the position capture array.		√
aerq_posgrab_read_position	This function is used to read the position stored in an index of the position capture array.		√
aerq_posgrab_reset_index	This function is used to set the index of the position capture array to 0.		√
aerq_posgrab_scan_enable	This function is used to enable the position capture interrupt. This function should be used when it is known that the position capture will not be generated faster than the servo loop update rate. The default servo loop update rate is 4Khz. This method is preferred since the servo loop will not be interrupted. This eliminates any potential for servo loop beating with the user-interrupt (UINT) input.		√
aerq_pso_get_diag	Acquires the PC-PSO diagnostics information such as the position register values, inputs, outputs and A/D value.	√	

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_psoc	Issues the PC-PSO board conditional firing command. The five possible cases are: Case 0: Position tracking is enabled unconditionally. The input signals are ignored. Case 1: Position tracking is enabled when a specified input bit number is in a specified state. In this mode counter data is retained when the position counter is disabled. Case 2: This mode is the same as Case 1 except that in this mode, counter data is reset to 0 when the position counter is disabled. Case 3: Position tracking is enabled when input bits 0–7 are configured as specified in the in_map argument. If the inputs are not configured as specified in the in_map argument, then the 16 outputs are set according to the out_map argument. Case 4: Position tracking is enabled when the specified axis is within a defined window. The window is defined by a "low" and "high" position.	√	
aerq_psod	Issues the PSO-PC board firing distance setup command. The three possible cases are: Case 0: Indicates that the pulse will be fired at a fixed incremental distances.. Case 1: Indicates that the pulse output will occur at incremental distances as defined in a firing array. Case 2: Indicates that the pulse output will occur at absolute distances as defined in a firing array.	√	

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_psof	Enables/disables synchronized output firing for the PC-PSO board. This command activates or deactivates the pulse train output on LOUT1 and tracking features. The six possible cases are: Case 0: The output firing pulse train and tracking features are disabled. This is the default case. Case 1: The output firing pulse train is activated. No position tracking occurs in this mode. Case 2: The output firing pulse train is activated for a specified number of times. No position tracking occurs in this mode. Case 3: The output firing pulse train is activated . The position counter locks on to the motions of the specified axes (X,Y,Z,U). Up to three axes may be locked on simultaneously. Output firing occurs at distances established by the aerq_psod function (case 1 or 2). Case 4: This case activates the output firing pulse train and locks the position counter on the specified axes. Up to three axes may be locked on simultaneously. The output firing pattern is determined by bit mapping as established by the aerq_psom function. The bit values serve the following functions: 0 Causes the output to go/remain low 1 Causes the output to go/remain high Case 5: This case activates the output firing pulse and locks the position counters onto the specified axis. Up to three axes may be locked simultaneously. The output firing pattern is determined by bit mapping as established by the aerq_psom function. The bit values serve the following functions: 0 Causes no output 1 Causes the output to be a single pulse train defined by the aerq_psop function.	√	
aerq_psom	Downloads a bit map array to the PC-PSO card. The bit patterns are used to specify when the laser is to be “On”, or when to fire a pulse output. If a bit is a 1, the laser output is on. If a bit is a 0, the laser output is off.	√	

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_psop	Defines the pulse train output of the PC-PSO card. The six possible cases are: Case 0: This case sets the width of a single pulse output in tenths of milliseconds. Case 1: This case sets a single pulse with definable pulse lead, pulse width, and pulse trail characteristics. The pulse lead, width and trail values should all be specified in tenths of milliseconds. Case 2: This case sets a pulse train with definable pulse lead, pulse width, pulse trail, ramp up/down, and pulse gap interval characteristics. The units used by the arguments are tenths of milliseconds. Case 3: This case sets a pulse train by defining a series of pulse widths and gap widths (in tenths of milliseconds) in an array. Case 4: This case sets up a simple one-shot pulse. The width of the single pulse output is in microseconds (with a minimum value of 1 microsecond). Case 5: This case is used to set the output toggle mode.	√	
aerq_psor	Provides various configurations of the PC-PSO's position counter. The psocase argument defines one of three possible configurations (0, 1, or 3) for the position counters: Case 0: This mode is used to clear all previous real time control from the counter. Case 1: This mode is used to stop the position counter from recording new data and retains current data under operator command. Case 3: This mode is used to stop the position counter from recording new data and returns the counter to zero.	√	

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_psot	Sets/clears digital output lines (OUT0-OUT15) and/or set the desired voltage (-10.0V to 10.0V) of the analog outputs (AOUT1 and AOUT2) on the PC-PSO. The argument psocase defines one of five possible configurations. The five possible cases are: Case 0: This case is used to set individual digital output lines to either a high signal (state = 1) or a low signal (state = 0). Case 1: This case is used to set a group of digital output lines high or low using a hexadecimal or decimal number specified by the states argument. Case 2: This case is used to set the output voltage of DAC0 (dac#=0) which is AOUT1 or DAC1 (dac#=1) which is AOUT2 to a constant value. Case 4: This case is used to set the output voltage of DAC0 (dac#=0) which is AOUT1 or DAC1 (dac#=1) which is AOUT2 to a value that is proportional to the velocity (velocity ramping). Case 6: This case is used to set the output voltage of DAC0 (dac#=0) which is AOUT1 or DAC1 (dac#=1) which is AOUT2 to a value that is proportional to the position (position ramping).	√	
aerq_pso_mem_read	Reads a memory location from the memory of the PSO's DSP.	√	
aerq_pso_mem_write	Writes "data" to a specified DSP address. The function can overwrite the existing memory data, AND the new data with the previous data, or OR the new data with the previous data.	√	
aerq_queue	Affects processing of commands that are waiting in the queue. The options are: 0 cancel all commands in queue 1 direct program flow to top of queue buffer and repeat entire command set 2 wait for signal on input line before processing next command	√	√
aerq_queue_pos_compare	This function is used to halt the command queue until the specified axis position is greater than or less than a specified position. The UNIDEX 500 compares the "position" argument to the actual real-time feedback position of the specified axis. This is the same position that is displayed in the U500 Diagnostic Window. The "position" argument is specified in program steps.	√	√

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_ramp	This function sets the ramp time. The ramp time is the time that it takes each axis to change from the current velocity to the new velocity and is used only for contour motion.	√	√
aerq_read_fpga_setup_code	This function is used to indicate which FPGA image was downloaded during the last reset. These setup codes can be changed from the U500 MMI or Toolkit software under System Options, FPGA Setup.		√
aerq_read_inputs	Reads inputs directly from the U500 or 4EN board.	√	√
aerq_read_option_code	This function is used to return the option code for Dual PSO and other factory enabled options.		√
aerq_read_param	Reads a parameter value from memory.	√	√
aerq_read_plane_feedrate	This function is used to read the current feedrate of the selected plane.	√	√
aerq_read_position	Reads the relative, absolute, and real time position of the axes. The position returned is in program steps. For the definition of a program step, refer to the English & Metric Conversion parameters in the U500 manual.	√	√
aerq_read_status *	Reads the status information from the U500 memory buffer. For up to date status information call the aerq_checkstatus function immediately before calling this function.	√	√
aerq_readadc	Reads the A/D values from one of the four A/Ds on the UNIDEX 500 ISA board or from one of eight A/Ds on the UNIDEX 500 PCI board.	√	√
aerq_readisbx	Reads data from the iSBX port.	√	
aerq_resetintr	Resets the interrupt lines on the ISA bus. Such interrupts may have been generated by the aerq_cint function or through the fault mask.	√	√
aerq_rounding	Turns corner rounding mode on/off. When performing a contour motion, this function affects the behavior of deceleration. When this function option is ON, the next block of motion will begin before the previous path is complete, creating a "rounded corner" at the end of the path. The time between the path stop and the start of the next block is defined as "non-ramp time" and may also be programmed through this function.	√	√
aerq_save_flash	This function is used to copy the contents of the edit buffer to flash memory. This function should be called after using the function in order for the changes to take effect during the next reset.		√

* See Table 10-3 for the arguments and return values for the aerq_read_status function

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_save_param	Saves the parameters loaded in memory, to the specified file. The parameters are loaded into memory with the aerq_open_param function. Once loaded, any changes made to parameters value are only stored in the memory space. This function is required to save the changes from memory to the permanent file location so that any parameter changes will be available after a reset.	√	√
aerq_scale	Sets modal operation G70/G71 (ENGLISH/METRIC mode). The UNIDEX 500 uses one of the these two conversion factors to convert programming units into machine steps.	√	√
aerq_scf	This function can enlarge or reduce a part by scaling motions/moves. The programmed distance is multiplied by the overriding scale factor.	√	√
aerq_scope_command	This function should first be used to prepare the U500 card to collect data, and then used to command the U500 card to collect the data. The following shows the order in which this function should usually be used: 1. Set time base. 2. Set number of samples. 3. Set auxiliary buffer type (optional). 4. Start tracking. 5. Set data type to be returned. This function can be called again after the aerq_scope_dump function to set up the return of other types of data. (i.e. with a SCMDCDMVEL argument).	√	√
aerq_scope_dump	Retrieves the scope data from card. This function should be called after the aerq_scope_command function has been used to set the data return type and find that the card is done collecting samples. This function then places these samples in an array of type ‘long’.	√	√
aerq_scope_trigger	Issues a queued command that triggers the U500 card to begin collecting data. This allows the U500 to start collecting data synchronously with commands in the queue buffer.	√	√
aerq_segment	Sets the amount of time used to "segment" each motion. The UNIDEX 500 divides each motion into "segments" and then cubic splines those segments into 0.25 millisecond velocity commands to the servo loop. The larger the segment size, the fewer number of steps are required for internal calculation.	√	√

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_set_crc	Sets cutter compensation parameters. Cutter compensation offsets programmed moves to compensate for the size of the cutting tool. The UNIDEX 500 implements cutter compensation with the following options: 1. G40 – cutter comp. off 2. G41 – cutter comp. on, LEFT 3. G42 – cutter comp. on, RIGHT 4. G43 – define cutter radius 5. G44 – defined compensated axes.	√	√
aerq_set_driver_debug	Enables/Disables debug mode. In debug mode, messages within the device driver can be retrieved by calling the aerq_device_driver_msg function. These messages can be useful to verify device driver functionality.	√	√
aerq_set_plane	Sets the current active plane, 1 – 4.	√	√
aerq_single_setup	This function allows the U500 PCI to track the position of a single axis encoder channel. It can then generate an output pulse at a specified number of encoder counts. The output pulse width is user programmable. There are also two "window" counters and compare registers which can be used to qualify the output pulse.		√
aerq_slew	Used in conjunction with the joystick option to provide immediate axis control. While in the joystick slew mode, the following functions are available: **Joystick A Button:** Toggles pairs of axes/drives between *h1 v1 a*nd *h2 v2* as specified. **Joystick B Button:** Selects between high velocity, low velocity and absolute positioning mode. Refer to axis parameters x50 (Joystick High Speed), x51 (Joystick Low Speed) and x52 (AbsoluteMode Scale). **Joystick C Button**: Cancels previous joystick command.	√	√
aerq_softlimit	Sets the counterclockwise and clockwise travel limit distance (in machine steps) for the specified axis as referenced from the hardware home position. The axes must be homed using the aerq_homeaxes function before the software limits take effect.	√	√
aerq_software_init	Loads the internal buffers with parameter information from the U500 memory and does a software reset. This function can be called in place of aerq_init if the board is already initialized.	√	√

Table 10-2. Quick Library Functions (Continued)

Function Name	Description	ISA	PCI
aerq_software_position	Establishes a position for the specified axis that is referenced from the current hardware position. This position is useful after a freerun or when using either a joystick or a handwheel option, so that the new software position is updated to match the current hardware position.	√	√
aerq_spline	Specifies the controller's ability to perform cubic spline fitting of multiple successive target positions. The result is a smoother path, with minimal positional disturbances and jerking between points. The function is well suited for non-Cartesian geometric motion. The cubic splining function is in terms of position versus time for up to four axes at once.	√	√
aerq_start	Issues a queued command that is used to activate planes that are currently under the halt command. This function must be called within a plane that is not under the halt command. See the aerq_halt function.	√	√
aerq_targetrack_disable	Disables the target tracking mode on a single axis (0-3) and returns the system to normal.	√	√
aerq_targetrack_enable	Enables target tracking on a single axis (0-3). It will not enable if the axis is in a fault condition. Multiple axes can be enabled by repeating the function call.	√	√
aerq_targetrack_position	This function sets the target position for single axis. The axis will move at specified velocity to an absolute position.	√	√
aerq_trajectory	Sets one of two types of accel/decel ramping trajectories: linear and inverse sine. Linear ramping uses a constant acceleration. Inverse sine ramping can be used to reduce "jerky" motion during accel /decel.	√	√
aerq_trigger	Issues a real-time command that is used to activate planes that are currently under the halt command. See the aerq_halt function.	√	√
aerq_umfo	Overrides the MFO potentiometer setting.	√	√
aerq_velocity	Turns velocity profiling on or off. This function is used when performing contour type motion to blend consecutive motions into one continuous path. It is a modal command and as such will remain in effect until turned OFF.	√	√
aerq_write_fpga_setup_code	This function is used to set which FPGA image will be downloaded during the next reset. This only writes the data to the flash memory. To save the changes, use the function.		√
aerq_write_param	Writes a parameter value to memory. This parameter value will only be valid until the system is reset, unless the parameters are saved to a parameter file using the aerq_save_param function.	√	√
aerq_writeisbx	Writes data to the iSBX port.	√	
get_drive_scalefactor	Returns the scale factor for the axis specified in the parameters.	√	√

10.2.5. Arguments and Return Values for the aerq_read_status Function

Table 10-3. Arguments and Return Values for the aerq_read_status Function

nStatus – function argument	BIT #	Return Value
0		16 input line condition
1 – 4 fault/trap/limit information for axes 1 – 4 respectively	0	1 = position error, 0 = no fault
	1	1 = RMS current error, 0 = no fault
	2	1 = integral error, 0 = no fault
	3	1 = hardware limit +, 0 = no fault
	4	1 = hardware limit -, 0 = no fault
	5	1 = software limit +, 0 = no fault
	6	1 = software limit -, 0 = no fault
	7	1 = driver fault, 0 = no fault
	8	1 = feedback device error, 0 = no fault
	9	1 = global abort active, 0 = global abort inactive
	10 –11	*unused*
	12	1 = feedrate > max setting error, 0 = no fault
	13	1 = velocity error, 0 = no fault
	14	1 = emergency stop, 0 = no fault
	15 - 30	*unused*
5 returns axis active/ in position/ plane information	0	1 = axis 1 enabled, 0 = disabled
	1	1 = axis 2 enabled, 0 = disabled
	2	1 = axis 3 enabled, 0 = disabled
	3	1 = axis 3 enabled, 0 = disabled
	4	1 = axis 1 not in position, 0 = in position
	5	1 = axis 2 not in position, 0 = in position
	6	1 = axis 3 not in position, 0 = in position
	7	1 = axis 4 not in position, 0 = in position
	8	1 = plane 1 comm. busy, 0 = comm. OK
	9	1 = plane 2 comm. busy, 0 = comm. OK
	10	1 = plane 3 comm. busy, 0 = comm. OK
	11	1 = plane 4 comm. busy, 0 = comm. OK
	12	1 = queue 1 buffer is not empty, 0 = empty *

Table 10-3. Arguments and Return Values for the aerq_read_status Function (Continued)

nStatus – function argument	BIT #	Return Value
	13	1 = queue 2 buffer is not empty, 0 = empty *
	14	1 = queue 3 buffer is not empty, 0 = empty *
5	15	1 = queue 4 buffer is not empty, 0 = empty *
returns axis active/in position/plane	16	1 = plane 1 halted, 0 = plane 1 "running"
information	17	1 = plane 2 halted, 0 = plane 2 "running"
(continued)	18	1 = plane 3 halted, 0 = plane 3 "running"
	19	1 = plane 4 halted, 0 = plane 4 "running"
	20	1 = global abort active
	21	1 = feedhold active
	22	1 = PC Bus Interrupt high
	23	1 = not ready for next command
	24 -31	*unused*
6		returns the current manual feedrate override % (MFO)
	0-1	00 = high velocity mode
		01 = low velocity mode
		1x = absolute positioning mode
	2-3	00 = plane 1 active
		01 = plane 2 active
		1x = block delete active (digitizing mode)
	4	1 = joystick interlock open (error)
		0 = joystick interlock closed (normal)
7	5-7	000 = no current horizontal axis defined
returns joystick status		001 = axis 1 active
		010 = axis 2 active
		011 = axis 3 active
		100 = axis 4 active

Table 10-3. Arguments and Return Values for the aerq_read_status Function (Continued)

nStatus – function argument	BIT #	Return Value
7 returns joystick status (continued)		current vertical axis (0-4)
	8-10	000 = no current vertical axis defined 001 = axis 1 is the active vertical axis 010 = axis 2 is the active vertical axis 011 = axis 3 is the active vertical axis 100 = axis 4 is the active vertical axis
	11	1 = received joystick cancel command
	12-13	*unused*
	14	0 = joystick is deactivated 1 = joystick is now active
8		returns the currently active board number (1-6)
9 – 12		returns the number of number of actions remaining in queue for planes 1-4. This data is meaningless when used with the PLC or QUEUE commands.

* An empty queue means that there are no unprocessed commands. If a plane's queue buffer is marked "not empty", it is processing commands.

10.2.6. Programming Example

```
/*****************************************************************************
      Example 1       link with Qlb50032.lib

      This example illustrates:
1:    Opening and closing the board pointer
2:    Initializing the U500
3:    Enabling and homing the axes
4:    Executing an index move, and setting the feedrate
5:    Executing contour moves (linear and circular) and setting
      the contour feedrate
6:    Waiting for moves to finish
7:    Checking axis errors using check status and read status fns
8:    Loading a calibration file
*****************************************************************************/

#include <stdio.h>
#include <conio.h>
#include "aerotype.h"
#include "boards.h"
#include "build.h"
#include "quick.h"
#include "qintface.h"
#include "qdrvw32.h"
#include "aerqcode.h"
#include "qparam.h"

#define       X_STEPS_PER_MM    1000
#define       Y_STEPS_PER_MM    1000

// error checking functions
void check_send_error( AERERR_CODE nRc);
int   U500_CheckAxisError( BOARD *bp, short axes);

// motion functions
AERERR_CODE     U500_MoveXYIndex(BOARD *bp, short nAxes,
                                        double lX, double lY, double
                                        xFeed, double yFeed);
AERERR_CODE     U500_MoveXYLinear(BOARD *bp, short nAxes,
                                  double lX, double lY, double
                                  cFeed);
AERERR_CODE U500_MoveXYCW(BOARD *bp, double e1, double e2,
                                  double c1, double c2, double
                                  cFeed);
AERERR_CODE     U500_MoveXYCCW(BOARD *bp, double e1,
                                        double e2, double c1,
                                        double c2, double cFeed);
AERERR_CODE  U500_WaitForMoveDone( BOARD *bp , short axes );
```

```
void main(void)
{
        BOARD *bp;
        AERERR_CODE nRc;

        /* firmware file for the REV C ISA board */
        char firm[] = {"c:\\u500\\mmi\\u500c.jwp"};

        /* generic parameter file, if there is a 5 digit parameter file
       created by the factory, use it in place if u500.prm  */
        char param[] = {"c:\\u500\\mmi\\u500.prm"};




        /* generic calibration file (optional) */
        /* char calfile[]  = {"c:\\u500\\mmi\\mycal.cal"}; */

        /* get handle to device and allocate board structure */
       if((bp = aerq_open(0)) == NULL)
        {                               // for board 1
                printf("Could not open board");
        }

        /* U500 initialization */
        if((nRc = aerq_init(bp, firm, param)) != 0)
        {
                check_send_error(nRc);
                exit(0);
        }

        /* enable the axes */
        if((nRc = aerq_enableaxes(bp, AXIS1|AXIS2)) != 0)
                check_send_error(nRc);
        U500_WaitForMoveDone(bp, AXIS1|AXIS2);
        if(U500_CheckAxisError(bp, AXIS1|AXIS2) )
                getch();

        if((nRc = aerq_homeaxes(bp, AXIS1|AXIS2)) != 0)
                check_send_error(nRc);

        U500_WaitForMoveDone(bp, AXIS1|AXIS2);
        if(U500_CheckAxisError(bp, AXIS1|AXIS2) )
                getch();

        /*(optional) -  load calibration file after axes have been homed
        if((aerq_loadcalfile(bp, calfile)) != 0)
                check_send_error(nRc);  */
```

```
    /* index move, G0 */
    U500_MoveXYIndex(bp, AXIS1|AXIS2, 20, 10, 50, 100000);
    U500_WaitForMoveDone(bp, AXIS1|AXIS2);
    if(U500_CheckAxisError(bp, AXIS1|AXIS2) )
            getch();

    /*  linear contour move, G1 */
    U500_MoveXYLinear(bp, AXIS1|AXIS2, 40, 40, 100000);
    U500_WaitForMoveDone(bp, AXIS1|AXIS2);
    if(U500_CheckAxisError(bp, AXIS1|AXIS2) )
            getch();




    /* clockwise circular move */
    U500_MoveXYCW(bp, 0, 0, 10, 10, 100000);
    U500_WaitForMoveDone(bp, AXIS1|AXIS2);
    if(U500_CheckAxisError(bp, AXIS1|AXIS2) )
            getch();

    /* counter_clockwise circular move */
    U500_MoveXYCCW(bp, 0, 0, 20, 20, 100000);
    U500_WaitForMoveDone(bp, AXIS1|AXIS2);
    if(U500_CheckAxisError(bp, AXIS1|AXIS2) )
            getch();

    U500_MoveXYLinear(bp, AXIS1|AXIS2, 25, 25, 100000);
    U500_MoveXYLinear(bp, AXIS1|AXIS2, 25, 25, 100000);
    U500_MoveXYLinear(bp, AXIS1|AXIS2, -50, -50, 100000);

    aerq_close(bp);
}
```

```
/*************************************************************************
This function moves two axes to the specified position using an
"index" move.  A feedrate is specified for each axis.

Note: This function can be extended to up to four axes.
*************************************************************************/

AERERR_CODE     U500_MoveXYIndex(BOARD *bp, short nAxes,   double lX,
                                        double lY, double xFeed, double yFeed)
{
      INDEXMOVE    Index;
      AERERR_CODE          eRc;

      memset(&Index, 0, sizeof(INDEXMOVE));

      if(xFeed != 0.0)
      {
            /*  x axis index feedrate in program steps/sec      */
            Index.ifeed[0] = xFeed;
      }
      f(yFeed != 0.0)
      {
            /*  y axis index feedrate in program steps/sec      */
            Index.ifeed[1] = yFeed;
      }

      /* set the feedrates */
      aerq_index_feed(bp, &Index);

      Index.dist[0] = (long) (lX * X_STEPS_PER_MM);
      Index.dist[1] = (long) (lY * Y_STEPS_PER_MM);
      Index.axes = nAxes;

      if(eRc = aerq_index(bp, &Index))
            check_send_error(eRc);

      return(eRc);
}
```

```
/*****************************************************************************
      This function moves two axes to the specified position using a
      "contour" motion.  The axes will start and stop at the same time and
      move at the vector feedrate specified.

      Note: This function can be extended to up to four axes.
*****************************************************************************/
AERERR_CODE      U500_MoveXYLinear(BOARD *bp, short nAxes, double lX,
                                                  double lY, double cFeed)
{
      LINEARMOVE  Linear;
      CIRCMOVE      Cmove;
      AERERR_CODE           eRc;

      if(cFeed != 0.0)
      {
                /* set contour feedrate in program steps/sec */
                aerq_contour_feed(bp, cFeed);
      }

      memset(&Linear, 0, sizeof(LINEARMOVE));
      memset(&Cmove, 0, sizeof(CIRCMOVE));

      Linear.dist[0]     = (long) (lX * X_STEPS_PER_MM);
      Linear.dist[1]     = (long) (lY * Y_STEPS_PER_MM);
      Linear.axes        = nAxes;

      if(eRc = aerq_contour(bp, &Cmove, &Cmove, &Linear))
      check_send_error(eRc);

      return(eRc);
}
```

```
/*****************************************************************************
This function performs a circular "contour" motion in the clockwise
direction. The endpoints and center of the move must be specified. The
axes will start and stop at the same time and move at the vector feedrate
specified.

Note: This function can be extended to up to four axes.
*****************************************************************************/
AERERR_CODE     U500_MoveXYCW(BOARD *bp, double e1,
                                        double e2, double c1,
                                        double c2, double cFeed)
{
    LINEARMOVE              Linear;
    CIRCMOVE                CWmove,  CCWmove;
    AERERR_CODE             eRc;

    if(cFeed != 0.0)
    {
        /* set contour feedrate in program steps/sec */
        aerq_contour_feed(bp, cFeed);
    }


    memset(&Linear, 0, sizeof(LINEARMOVE));
    memset(&CWmove, 0, sizeof(CIRCMOVE));
    memset(&CCWmove, 0, sizeof(CIRCMOVE));

    CWmove.mtype = MOVECW;
    CWmove.axis1 = AXIS1;
    CWmove.axis2 = AXIS2;
    CWmove.endpnt1 = (long) (e1 * X_STEPS_PER_MM);
    CWmove.endpnt2 = (long) (e2 * Y_STEPS_PER_MM);
    CWmove.cen1 = (long) (c1 * X_STEPS_PER_MM);
    CWmove.cen2 = (long) (c2 * Y_STEPS_PER_MM);

    if(eRc = aerq_contour(bp, &CWmove, &CCWmove, &Linear))
        check_send_error(eRc);

return(eRc);
}
```

```
/*************************************************************************
This function performs a circular "contour" motion in the counter-
clockwise direction.  The endpoints and center of the move must be
specified. The axes will start and stop at the same time and move
at the vector feedrate specified.

Note: This function can be extended to up to four axes.
*************************************************************************/
AERERR_CODE      U500_MoveXYCCW(BOARD *bp, double e1, double
e2, double c1, double c2, double cFeed)
{
      LINEARMOVE                  Linear;
      CIRCMOVE                    CWmove,  CCWmove;
      AERERR_CODE                 eRc;

      if(cFeed != 0.0)
      {
            /* set contour feedrate in program steps/sec */
            aerq_contour_feed(bp, cFeed);
      }

      memset(&Linear, 0, sizeof(LINEARMOVE));
      memset(&CWmove, 0, sizeof(CIRCMOVE));
      memset(&CCWmove, 0, sizeof(CIRCMOVE));

      CCWmove.mtype = MOVECCW;
      CCWmove.axis1 = AXIS1;
      CCWmove.axis2 = AXIS2;
      CCWmove.endpnt1 = (long) (e1 * X_STEPS_PER_MM);
      CCWmove.endpnt2 = (long) (e2 * Y_STEPS_PER_MM);
      CCWmove.cen1 = (long) (c1 * X_STEPS_PER_MM);
      CCWmove.cen2 = (long) (c2 * Y_STEPS_PER_MM);

      if(eRc = aerq_contour(bp, &CCWmove, &CWmove, &Linear))
            check_send_error(eRc);

      return(eRc);
}
```

```
/*****************************************************************************
    This function waits for the U500 command buffer to clear and
    checks the in-position status information. If an axis error
    condition occurs this function returns and the error must be
    handled elsewhere.
*****************************************************************************/
AERERR_CODE  U500_WaitForMoveDone( PBOARD pb , short axes )
{
    AERERR_CODE eRc;
    int             count = 0 ;
    long            mask ;



    /* make mask to check in position and queue empty */
    mask = 0x11100L | ( (long) axes << 4 ) ;

    while( 1 )
    {
            /* Update status. This will take .25-.50 ms and is not
            really PC dependent. */

            if( eRc = aerq_checkstatus( pb ) )
                    break;

            /* Wait for queue to clear. Give minimum number of tries
            to insure that there is not a command in the U500's receive buffer that the
            "checkstatus" call does not know about. We should not need more that 1ms
            here. Also check for in-position.  */

            if( !(aerq_read_status( pb, 5 ) & mask ) && count > 10 )
                    break;

            /* Check for axis error. Break loop if one occurs. */
            if(aerq_read_status( pb, 1 ) || aerq_read_status( pb, 2 )
            || aerq_read_status( pb, 3 ) || aerq_read_status( pb, 4 ))
                    break;

            count++;
    }
    return( eRc );
}
```

```
/*****************************************************************************
      This function checks Aerotech return codes and decodes them to .
      ASCII strings The driver errors (DRV) are generally not
      recoverable errors. They indicate problems with the U500 hardware
      or registry parameters. The only exception is:
            "U500_E_DRV_PREVGRP_NOT_DONE".
      This code indicates that the U500's receive buffer is full
      and cannot accept any more commands.
*****************************************************************************/
void check_send_error( AERERR_CODE nRc )
{
      char errmsg[80];

      /* this following error is recoverable, and should be handled
            by giving the command queue time to process commands
            (or if using NT, by starting a new thread) */
      if(nRc == U500_E_DRV_PREVGRP_NOT_DONE)
      {
            /* add user code to handle a full command buffer here */
      }

      aerq_errortoascii( nRc , errmsg );
      printf( "%s\n", errmsg ) ;
}
```

```
/*****************************************************************************
        This function checks for an axis error condition on the axes
        specified in "axes". It also prints out the axis error condition.

        Return: 0 every thing is OK
        <>0 axis has fault condition

        Note: Axis faults are cleared by "aerq_faultack()". Most faults
        automatically disable the axis, as configured by the FAULTMASK
        parameters. The only recoverable fault is a hardware limit
        fault.  In this case, aerq_faultack() moves the axis out of the limit.
*****************************************************************************/
int U500_CheckAxisError( BOARD *pb, short axes )
{
        int   error = 0 ;
        short i ;
        short mask ;
        char  errormsg[80];
        long  axiserror;
        AERERR_CODE    eRc;

        /* update status info from U500 card */
        if( eRc = aerq_checkstatus( pb ) )
        {
                check_send_error( eRc );
                return( 2 );
        }



        /* check each axis for error condition */
        mask = 1 ;
        for( i = 1 ; i < 5 ; i++ )
        {
                if( mask & axes )       /* check this axis ? */
                {
                        if( axiserror = aerq_read_status( pb, i ) )
                        {
                                /* axis has error, display message */
                                aerq_axiserrortoascii( axiserror , errormsg ) ;
                                printf( "AXIS ERROR %i \"%s\"\n", i, errormsg);
                                error = 1 ;
                                break;
                        }
                }
                mask <<=1 ;    /* next axis mask */
        }
        return( error );
}
```

10.3. Summary of the UNIDEX 500 WAPI Functions (32 bit)

10.3.1. Overview

The UNIDEX 500 Windows API functions provide an easy interface that is useful for high level programming in the Windows environment. All of the functions are incorporated in the file win50032.dll. The WAPI functions are ideal for developing applications in languages such as Visual BASIC or Delphi.

When the WAPI functions are used in a C environment, the library file "win50032.lib" must be included in the project. If the WAPI functions are used in a Visual Basic environment, the library file "winaer.inc" must be included in the project.

While the WAPI functions are easier to use, they have higher overhead. The functions at the Windows application layer must call the lower level functions in the U500 quick library (incorporated in qlb50032.dll) to actually communicate with the UNIDEX 500.

10.3.2. Device Driver Information

In the 32-bit software, the UNIDEX 500 communications are handled using a device driver. In a custom application, the first step is to open the device driver using the WAPIAerOpen function. Before exiting the application, the WAPIAerClose function must be called to close the device driver. The name and path of the UNIDEX 500 device driver is dependent on the operating system and the bus type. The device driver is installed during installation of the U500 32-bit MMI/Toolkit software.

U500 ISA DEVICE DRIVER INFORMATION

OPERATING SYSTEM	DEVICE DRIVER	PATH
Windows 95/98	U500ISA.VXD	C:\WINDOWS\SYSTEM
Windows NT	U500ISA.SYS	C:\WINNT\SYSTEM32\DRIVERS

U500 PCI DEVICE DRIVER INFORMATION

OPERATING SYSTEM	DEVICE DRIVER	PATH
Windows 95	U500PCI.VXD	C:\WINDOWS\SYSTEM
Windows 98	U500PCI.SYS	C:\WINDOWS\SYSTEM
Windows NT	U500PCI.SYS	C:\WINNT\SYSTEM32\DRIVERS

10.3.3. Registry Information

The 32bit libraries for the UNIDEX 500 require an entry in the WINDOWS registry file for the ISA and PCI boards. The registry entries as stated below are created by the MMI installation software. The user may not need to recreate them.

WINDOWS 95/98:

The WINDOWS 95 registry editor is called REGEDIT.EXE and is located in the C:\WINDOWS directory. A registry entry is made in:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD.

WINDOWS NT

The WINDOWS NT registry editor is called REGEDT32.EXE and is located in the C:\WINNT\SYSTEM32 directory. A registry entry is made in:

HKEY_LOCAL_MACHINE\System\ CurrentControlSet\Services.



The system registry holds time-out and base address information for the UNIDEX 500 ISA card. In the 16-bit software, this information was stored in the configuration file “u500.cfg”.



Refer to the Compiler/Language Requirements section in the QLIB programming help file for more information.

10.3.4. WAPI Library Functions

Table 10-4 provides a brief description of the 32 bit WAPI functions. For more information, syntax, and examples, refer to:

⇒ WAPI Programming Help File located in the U500 MMI program group. The file name is /u500/mmi/help/wapi.hlp.

⇒ Example C and Visual Basic code is located in the /u500/mmi/example directory

Note that while most functions are supported by the U500 PCI and U500 ISA card, some functions are specific to the card type.

Table 10-4. WAPI Library Functions

Function Name	Description	ISA	PCI
WAPIAerAbort	Clears the queue buffer and stops all motion of all active boards. All enabled axes will ramp to a stop using the max accel/decel parameter x16 Maximum Acceleration_Deceleration. The software position registers will then be updated with the new position. This function is similar to pressing the abort key.	√	√
WAPIAerAnalogVector	This function is used to generate an output voltage that is proportional to axis position or velocity.		√
WAPIAerAxisErrorToAscii	Converts the axis status value to a string that can be displayed to the user.	√	√
WAPIAerCheckInitz	Checks if the U500 board is initialized.	√	√
WAPIAerCheckStatus	Updates the status information that is output by the WAPIAerReadStatus and the WAPIAerReadPosition functions. This function retrieves system status bytes from the U500 and fills in appropriate locations of the board structure.	√	√
WAPIAerClose	Closes the device driver when the user is finished with the UNIDEX 500 board. It also frees any memory that was allocated by the WAPIAerOpen function.	√	√
WAPIAerDualSetup	This function allows the U500 PCI to track on any combination of two axes and will generate an output pulse at the vector distance specified. The U500PCI must also factory configured to support this option.		√
WAPIAerDSPMemRead	Reads a memory location from the DSP's memory	√	√
WAPIAerDSPMemWrite	Writes "data" to a specified DSP address. The function can overwrite the existing memory data, AND the new data with the previous data, or OR the new data with the previous data.	√	√

* See Table 10-5 for the return values updated by the WAPIAerCheckstatus function that are read using the WAPIAerReadStatus function.

Table 10-4. WAPI Library Functions (Continued)

Function Name	Description	ISA	PCI
WAPIAerEditFlash	This function is used to copy the contents of the flash memory to an edit buffer. This function should be called before the using the following functions: WAPIAerReadFpgaSetupCode WAPIAerWriteFpgaSetupCode WAPIAerReadOptionCode		√
WAPIAerFaultAck	Acknowledges fault conditions. If an axis is in a limit, the "fault acknowledge" moves the axis out of the limit.	√	√
WAPIAerGetDrvVersion	Returns the version number of the device driver that is in use.	√	√
WAPIAerGetProgAuxCode	This function is used to check the auxiliary code for each command in a program. The code corresponds to a programming command that : ⇒ triggers an event that should be displayed to the user ⇒ requires implementation with code that is not provided by the C libraries ⇒ requires some action by the user.	√	√
WAPIAerGetProgErrMsg	This function is used to return an error message that occurred during program execution.	√	√
WAPIAerGetProgPacket	This function is used to return program information, such as: line number, single/auto mode, program name, pause and cycle status.	√	√
WAPIAerGetQlibVersion	Returns the version number of the quick library that is in use.	√	√
WAPIAerGetWAPIVersion	Returns the version number of the WAPI library that is in use.	√	√
WAPIAerInitialize	Initiates a hard reset of the UNIDEX 500 and loads the system firmware and the system parameter file into the PCs memory. Next, this function downloads the parameters to the memory on the UNIDEX 500 board.	√	√
WAPIAerInitializePSO	Initializes the PC-PSO board and downloads the PSO firmware file to the board.	√	
WAPIAerInPosition	This function is used to set a specified output when one or more axes are within a specified error band of their final positions. The axes must meet the following criteria to be considered in position: • axis must be enabled • commanded velocity must be zero • actual feedback velocity must be less than 1 count / millisecond position error must be less than axis parameter x35 (in position dead band)		√

Table 10-4. WAPI Library Functions (Continued)

Function Name	Description	ISA	PCI
WAPILoadCalFile	Downloads calibration data from the Calibration File (*.CAL) to the UNIDEX 500 board. Axis calibration will be active when: 1. the axis calibration parameter (x15 and/or x71) is set to "yes" 2. the ASCII calibration file is present and has been loaded 3. the axis has been homed Subsequent axis positioning is then interpolated based on the .CAL file data. A maximum of 2,047 data points of correction is available. The calibration files may also contain orthogonality correction data.	√	√
WAPIAerLoadProgram	This function is used to load a U500 parts program. Up to four programs can be loaded and executed at one time.	√	√
WAPIAerLoadArray	This function stores array data on the U500 PCI Card for use with the WAPIAerSingleSetup function. This allows fast access to the array for time critical applications.		√
WAPIAerOpen	Opens the device driver and allocates memory so that the software can communicate with the UNIDEX 500 board.	√	√
WAPIAerOpenParam	Allocates a location in memory and transfers parameter values from a file to that location in memory. After this function is called successfully, all parameter changes are only stored in memory until the WAPIAerSaveParam function is called to save the parameters to a file.	√	√
WAPIAerPosgrabDisable	This function is used to disable the position capture interrupt.		√
WAPIAerPosgrabHardwareEnable	This function is used to enable position capture interrupt. This should be used when it is known that the position capture will be generated faster than the servo loop update rate. The default servo loop update rate is 4Khz. The position capture interrupt will take precedence over the servo loop update interrupt and servo loop beating with the user- interrupt (UINT) input may occur.		√
WAPIAerPosgrabReadArraySize	This function is used to read the current size of the position capture array.		√
WAPIAerPosgrabReadIndex	This function is used to return the current index of the position capture array.		√
WAPIAerPosgrabReadPosition	This function is used to read the position stored in an index of the position capture array.		√
WAPIAerPosgrabResetIndex	This function is used to set the index of the position capture array to 0.		√

Table 10-4. WAPI Library Functions (Continued)

Function Name	Description	ISA	PCI
WAPIAerPosgrabScanEnable	This function is used to enable the position capture interrupt. This function should be used when it is known that the position capture will not be generated faster than the servo loop update rate. The default servo loop update rate is 4Khz. This method is preferred since the servo loop will not be interrupted. This eliminates any potential for servo loop beating with the user-interrupt (UINT) input.		√
WAPIAerQueuePosCompare	This function is used to halt the command queue until the specified axis position is greater than or less than a specified position. The UNIDEX 500 compares the "position" argument to the actual real time feedback position of the specified axis. This is the same position that is displayed in the U500 Diagnostic Window. The "position" argument is specified in program steps.	√	√
WAPIAerReadAtd	Reads the A/D values from one of the four A/Ds on the UNIDEX 500 ISA board or from one of eight A/Ds on the UNIDEX 500 PCI board.	√	√
WAPIAerReadFpgaSetupCode	This function is used to indicate which FPGA image was downloaded during the last reset. These setup codes can be changed from the U500 MMI or Toolkit software under System Options, FPGA Setup.		√
WAPIAerReadiSBX	Reads data from the iSBX port.	√	
WAPIAerReadOptionCode	This function is used to return the option code for U500 PCI Dual PSO option and other factory enabled options.		√
WAPIAerReadParam	Reads a parameter value from memory.	√	√
WAPIAerReadPlaneFeedrate	This function is used to read the current feedrate of the selected plane.	√	√
WAPIAerReadPosition	Reads the relative, absolute, and real time position of the axes. The position returned is in program steps. For the definition of a program step, refer to the English & Metric Conversion parameters in the U500 manual.	√	√
WAPIAerReadPosition_r	Reads the relative, absolute, and real time position of the axes using a pointer variable. The position returned is in program steps. For the definition of a program step, refer to the English & Metric Conversion parameters in the U500 manual.	√	√
WAPIAerReadVariable	Returns the value of a U500 variable. The U500 allows for 256 user variables.	√	√
WAPIAerReadVariable_r	Returns the a pointer to the value of a U500 variable. The UNIDEX 500 allows for 256 user variables.	√	√

Table 10-4. WAPI Library Functions (Continued)

Function Name	Description	ISA	PCI
WAPIAerReadVelocity	Reads the velocity values from one of the four axes on the UNIDEX 500 board.	√	√
WAPIAerReadStatus	Reads the status information from the U500 memory buffer. For up to date status information call the WAPIAerCheckStatus function immediately before calling this function.	√	√
WAPIAerResetIntr	Resets the interrupt lines on the ISA bus. Such interrupts may have been generated by the CINT command or through the fault mask.	√	√
WAPIAerRunProgram	This function is used to start/resume program execution. The program will be executed within the dlls once this function is called successfully. This function should be used to : ⇒ start program execution ⇒ resume program execution when in single step mode ⇒ resume program execution when an auxiliary code is returned that requires special code or user interaction. Refer to the WAPIAerGetProgAuxCode function	√	√
WAPIAerSaveFlash	This function is used to copy the contents of the edit buffer to flash memory. This function should be called after using the W function in order for the changes to take effect during the next reset.		√
WAPIAerSaveParam	Saves the parameters loaded in memory, to the specified file. The parameters are loaded into memory with the WAPIAerOpenParam function. Once loaded, any changes made to parameters value are only stored in the memory space. This function is required to save the changes from memory to the permanent file location so that any parameter changes will be available after a reset.	√	√

* See Table 10-5 for the arguments and return values for the WAPIAerReadStatus function

Table 10-4. WAPI Library Functions (Continued)

Function Name	Description	ISA	PCI
WAPIAerScopeCommand	This function should first be used to prepare the U500 card to collect data, and then used to command the U500 card to collect the data. The following shows the order in which this function should usually be used: 1. Set time base. 2. Set number of samples. 3. Set auxiliary buffer type (optional). 4. Start tracking. 5. Set data type to be returned. This function can be called again after the WAPIAerScopeDump function to set up the return of other types of data. (i.e. with a SCMDCDMVEL argument).	√	√
WAPIAerScopeDump	Retrieves the scope data from card. This function should be called after the WAPIAerScopeCommand function has been used to set the data return type and find that the card is done collecting samples. This function places these samples in an array of type 'long'.	√	√
WAPIAerSend	This function sends a command to the U500 card. This command is placed in the command queue buffer until it is executed.	√	√
WAPIAerSetAtd	Enables/Disables the data fetch of A/D values on the U500. This function must be called before the user can use the WAPIAerReadAtd function to read the A/D values.	√	√
WAPIAerSetProgExecMode	This function is used to control whether the program will be executed in auto or single-step mode.	√	√
WAPIAerSetProgLine	This function transfers program control to a desired line number.	√	√
WAPIAerSetProgNum	This function is used to specify which U500 parts program will be affected by the following functions: WAPIAerLoadProgram WAPIAerUnloadProgram WAPIAerGetProgAuxCode WAPIAerGetProgErrMsg WAPIAerSetProgLine WAPIAerSetProgExecMode This function is only required during multiple program execution.	√	√

Table 10-4. WAPI Library Functions (Continued)

Function Name	Description	ISA	PCI
WAPIAerSetVelocityTimebase	Used to specify how often the velocity information is updated on the UNIDEX 500 board, and to specify the units for the velocity which is derived from the timebase. The axis velocity scan timebase can range from 1 to 2^{23} milliseconds	√	√
WAPIAerSetVelocityTracking	Enables/Disables each axis velocity tracking function. Disabling velocity tracking when not in use reduces U500 firmware overhead.	√	√
WAPIAerSingleSetup	This function allows the U500 PCI to track the position of a single axis encoder channel. It can then generate an output pulse at a specified number of encoder counts. The output pulse width is user programmable. There are also two "window" counters and compare registers which can be used to qualify the output pulse.		√
WAPIAerSoftwareInitialize	Loads the internal buffers with parameter information from the U500 card's memory and does a software reset. This function can be called in place of WAPIAerInitialize if the board is already initialized.	√	√
WAPIAerStopProgram	This function is used to stop the execution of a specified U500 parts program.	√	√
WAPIAerUnloadProgram	This function is used to unload and free the memory allocated for a U500 program.	√	√
WAPIAerWriteFpgaSetupCode	This function is used to set which FPGA image will be downloaded during the next reset. This only writes the data to the flash memory. To save the changes, use the function.		√
WAPIAerWriteiSBX	Writes data to the iSBX port.	√	
WAPIAerWriteParam	Writes a parameter value to memory. This parameter value will only be valid until the system is reset, unless the parameters are saved to a parameter file using the WAPIAerSaveParam function.	√	√
WAPICommReady	Checks if the command queue of the U500 can accept another command.	√	√
WAPIErrortoAscii	Converts the AERERR_CODE error codes to a text message which can be displayed to the user.	√	√
WAPIGetAxis	Accepts an axis number (1 to 4) and returns the axis letter if the axis is mapped correctly.	√	√
WAPIGetBoardType	Returns the model of the current U500 board. The board must be already be initialized using the WAPIAerInitialize function.	√	√
WAPIGetDiag	Acquires the information displayed in the diagnostics screen.	√	√
WAPIGetPlane	Returns the plane to which an axis is mapped.	√	√
WAPIPsoGetDiag	Acquires the PSO diagnostics information such as the position register values, inputs, outputs and A/D value.		√

10.3.5. Arguments and Return Values for the WAPIAerReadStatus Function

Table 10-5. Arguments and Return Values for the WAPIAerReadStatus Function

nStatus – function argument	BIT #	Return Value
0		16 input line condition
1 – 4 fault/trap/limit information for axes 1 – 4 respectively	0	1 = position error, 0 = no fault
	1	1 = RMS current error, 0 = no fault
	2	1 = integral error, 0 = no fault
	3	1 = hardware limit +, 0 = no fault
	4	1 = hardware limit -, 0 = no fault
	5	1 = software limit +, 0 = no fault
	6	1 = software limit -, 0 = no fault
	7	1 = driver fault, 0 = no fault
	8	1 = feedback device error, 0 = no fault
	9	1 = global abort active, 0 = global abort inactive
	10 –11	*unused*
	12	1 = feedrate > max setting error, 0 = no fault
	13	1 = velocity error, 0 = no fault
	14	1 = emergency stop, 0 = no fault
	15 - 30	*unused*
5 returns axis active/ in position/ plane information	0	1 = axis 1 enabled, 0 = disabled
	1	1 = axis 2 enabled, 0 = disabled
	2	1 = axis 3 enabled, 0 = disabled
	3	1 = axis 3 enabled, 0 = disabled
	4	1 = axis 1 not in position, 0 = in position
	5	1 = axis 2 not in position, 0 = in position
	6	1 = axis 3 not in position, 0 = in position
	7	1 = axis 4 not in position, 0 = in position
	8	1 = plane 1 comm. busy, 0 = comm. OK
	9	1 = plane 2 comm. busy, 0 = comm. OK
	10	1 = plane 3 comm. busy, 0 = comm. OK
	11	1 = plane 4 comm. busy, 0 = comm. OK
	12	1 = queue 1 buffer is not empty, 0 = empty *
	13	1 = queue 2 buffer is not empty, 0 = empty *

Table 10-5. Arguments and Return Values for the WAPIAerReadStatus Function (Continued)

nStatus – function argument	BIT #	Return Value
5 returns axis active/ in position/plane information (Continued)	14	1 = queue 3 buffer is not empty, 0 = empty *
	15	1 = queue 4 buffer is not empty, 0 = empty *
	16	1 = plane 1 halted, 0 = plane 1 "running"
	17	1 = plane 2 halted, 0 = plane 2 "running"
	18	1 = plane 3 halted, 0 = plane 3 "running"
	19	1 = plane 4 halted, 0 = plane 4 "running"
	20	1 = global abort active
	21	1 = feedhold active
	22	1 = PC Bus Interrupt high
	23	1 = not ready for next command
	24 -31	*unused*
6		returns the current manual feedrate override % (MFO)
7 returns joystick status	0-1	00 = high velocity mode 01 = low velocity mode 1x = absolute positioning mode
	2-3	00 = plane 1 active 01 = plane 2 active 1x = block delete active (digitizing mode)
	4	1 = joystick interlock open (error) 0 = joystick interlock closed (normal)
	5-7	000 = no current horizontal axis defined 001 = axis 1 active 010 = axis 2 active 011 = axis 3 active 100 = axis 4 active
	8-10	current vertical axis (0-4) 000 = no current vertical axis defined 001 = axis 1 is the active vertical axis 010 = axis 2 is the active vertical axis 011 = axis 3 is the active vertical axis 100 = axis 4 is the active vertical axis

Table 10-5. Arguments and Return Values for the WAPIAerReadStatus Function (Continued

nStatus – function argument	BIT #	Return Value
7	11	1 = received joystick cancel command
returns joystick status	12-13	*unused*
(continued)	14	0 = joystick is deactivated 1 = joystick is now active
8		returns the currently active board number (1-6)
9 – 12		returns the number of number of actions remaining in queue for planes 1-4. This data is meaningless when used with the PLC or QUEUE commands.

* An empty queue means that there are no unprocessed commands. If a plane's queue buffer is marked "not empty", it is processing commands.

10.3.6. Example file for U500 "WAPI" library "WIN50032.DLL"

```
/*****************************************************************************

  Differences between the 16 bit windows library and the 32 library...
        1) name has changed from WINAER.DLL to WIN50032.DLL.
        2) functions now return LONG as error code instead of SHORT
        3) must call WAPIAerOpen to get handle to device driver
        4) must call WAPIAerClose to release handle
        5) include "WIN50032.LIB" in project
        6) config file is not used, this info is kept in the registry
        7) timeout and base address info also kept in the registry

*****************************************************************************/
#include <stdio.h>
#include <string.h>
#include "aerotype.h"
#include "build.h"
#include "wapi.h"

void displaystatus(void);

int main()
{
        long error ;
        char cmd[80];
        LPSTR psErr;

        /* firmware file for the U500 REV C ISA board */
        char firm[] = {"c:\\u500\\mmi\\u500c.jwp"};

        /* generic parameter file, if there is a 5 digit parameter file
        created by the factory, use it in place if u500.prm  */
        char param[] = {"c:\\u500\\mmi\\u500.prm"};

        error = WAPIAerOpen(0); /* open board 1 */
        if( error != 0 )
        {
                printf("Could not open board" );
                exit(0);
        }

        /* check if board has already been initialized */
        if( WAPIAerCheckInitz( 0 ) )
        {
                error = WAPIAerInitialize( NULL, firm , param);
        }
        else
        {       /* reset system software without hardware reset */
                error = WAPIAerSoftwareInitialize( param);
        }
```

```
	if( error )
	{
		printf("Initialization error %i\n", error ) ;
		error = WAPIErrorToAscii(error, psErr);
		printf("\n %s\n", psErr);
	}
		else
	{
		while( error >= 0 )
		{
			/* the WAPICommReady function checks if there is room for the
			command in the buffer, it really is not needed here as the
			commands are only sent as fast as the user can type, however
			this function may be useful with file input */
			if(WAPICommReady () == 0)
			printf("\nBuffer full, Wait before sending command");

			printf("Enter command..., ('quit' when done) ");
			gets( cmd );

			error = WAPIAerSend( cmd );
			printf("COMMAND: \"%s\" RETURN CODE: %li\n", cmd,
				error);
			displaystatus();

			error = WAPIErrorToAscii (error, psErr);
			printf("\n %s\n", psErr);

			if( !stricmp( cmd, "quit" ) )
				error = -1 ;
		}
	}
	printf("Done\n");

	WAPIAerClose(0) ;
	return( 0 ) ;
}
```

```
void displaystatus(void)
{
	long error ;
	short i ;
	error = WAPIAerCheckStatus();
	if( !error )
	{
		printf("STATUS: ");
		for( i = 0 ; i < 6 ; i++ )
		{
			printf("%8lX,", WAPIAerReadStatus(i) );
		}
		printf("\n");
	}
}
```

∇ ∇ ∇

CHAPTER 11: UTILITY PROGRAMS

In This Section:

- Introduction 11-1
- The AERVER.EXE Utility 11-1
- The DSPDMP32.EXE Program 11-2

11.1. Introduction

This chapter explains how to use the UNIDEX 500 utility programs. These programs are located under the ..\UTILITY subdirectory on the PC after the U500 software is installed. This subdirectory also contains a READ.ME text file that gives a brief overview of these programs.

11.2. The AERVER.EXE Utility (32 bit software only)

The AERVER.EXE utility is a program that is located under the ..\UTILITY subdirectory on the PC after the U500 software is installed. This program creates a text file that contains a list of the names and version numbers of the .dll and .ocx files that are installed into the system directory by the U500 software. This is a useful tool when troubleshooting a software installation problem.

The following ASCII files are used/created by this utility:

aerver.lst	: this file contains the names of the .dll and .ocx files that are installed with the MMI software
aerver95.txt	: this file contains the names and the version numbers of the files that should be present when using Windows 95
aerverNT.txt	: this file contains the names and the version numbers of the files that should be present when using Windows NT
aerver.txt	: this file contains the names and version numbers of the files found on the computer after the utility has been run

This program searches for the files listed in the aerver.lst file. Then it writes the results of the search to the file, aerver.txt. The files listed in the aerver.txt should have the same or newer version number as the files in either aerver95.txt or aerverNT.txt.

11.3. The DSPDMP32.EXE Program (32 bit software only)

The DSPDMP32.EXE utility is a program that is located under the ..\utility subdirectory on the PC after the U500 software is installed. This program is used to read memory locations from the U500 or PC_PSO DSP's memory.

The U500's 24 bit DSP stores data in the X: and Y: data memory spaces. A long access (L:) is a combination of X and Y memory addresses with X as the upper 24 bits and Y as the lower 24 bits. The P: memory space is used to store the DSP program data.

Refer to the file memloc.txt in the ..\doc subdirectory for a list of useful DSP memory locations.

∇ ∇ ∇

CHAPTER 12: TECHNICAL DETAILS

In This Section:

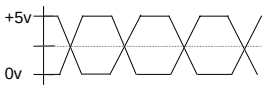
- Technical Details Common between U500 PCI and ISA Cards..... 12-1
- UNIDEX 500 ISA Technical Details..... 12-21
- UNIDEX 500 PCI Technical Details..... 12-37

12.1. Technical Details Common between U500 PCI and ISA Cards

12.1.1. Encoder Signal Specifications

The UNIDEX 500 ISA Revision C board samples the encoder signals at 20 MHz. The UNIDEX 500 PCI board samples the encoder signals at 40 MHz. The sample frequency is not jumper selectable. The maximum input frequency is 5 MHz (4 MHz recommended) for the ISA card and 10 MHz (8 MHz recommended) for the PCI card. The maximum data rate for the ISA card is 20 MHz (16 MHz recommended). The maximum data rate for the PCI card is 40 MHz (32 MHz recommended). The minimum edge separation is 50 ns (62.5 ns recommended) for the ISA card, and 25 ns (31.25 ns) for the PCI card.

12.1.1.1. Differential Encoders



The UNIDEX 500 accepts differential RS-422 type square wave encoder signals. A "times 4" multiplication is always performed on the encoder fundamental line count. For example, if the encoder line count is 1,000 lines, the effective machine resolution will be 4,000 machine steps (or counts) per revolution.

The marker and quadrature signals use 26LS32 type RS-422 receivers. The sine and cosine signals are pulled to +5 V through 10K ohm resistors. The encoders are terminated with 180 ohm resistors.

12.1.1.2. Single Ended Encoders



Third party, single ended encoders may be used with the UNIDEX 500 by connecting a 4.7K ohm 1/4 watt resistor from the unused differential input to signal common as illustrated below in Figure 12-1.

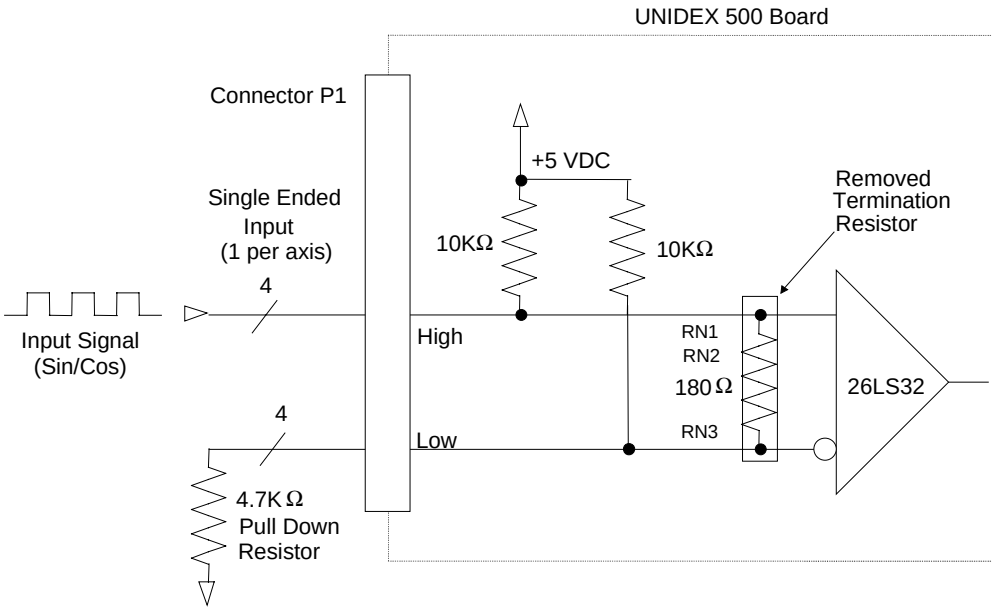


Figure 12-1. Electrical Characteristics of a Single Ended Encoder Interface

12.1.1.3. Software Setup for Single-ended Encoders

Normally, the U500 expects a differential encoder to be connected to it. Once the single-ended encoder has been connected to the BB500 or BB501 Breakout board or the DR500 drive chassis, a software setting will have to be changed to account for it. In the Edit Parameters window, select the parameter tab called "Faults" and find the parameter "*Error mask fault*." In the "Faults" parameter tab, modify this parameter so that the "Feedback Trap" is turned off (do so by unchecking it inside the tab). This parameter will have to be changed for every axis that has a single-ended encoder.



Save the changes, then reinitialize the U500 board for the changes to take effect.

Single ended encoders are not recommended for use with the U500. Loss of the encoder signal cannot be detected with this configuration. The result can be a runaway condition of the motor.

12.1.1.4. Aerotech Stepper Motors with the Home Marker Option

When using Aerotech stepper motors equipped with the home marker option, it is necessary to replace the appropriate termination resistor(s) with a 0.1 µF capacitor (purchased separately). If fewer than four such stepper motors (with the home marker wheel option) are used, be sure to install the 0.1 µF capacitors as appropriate, while keeping in place the termination resistors for the other axes. See Figure 12-2.

The removable termination resistors for axes 1-4 are grouped into three in-line resistor networks (RN1, RN2, and RN3). If your application mixes Aerotech stepper motors (that have the home marker wheel option) with differential or single-ended encoders, you must purchase separate 180 ohm resistors to replace the termination resistors that have been removed as part of the resistor network(s).

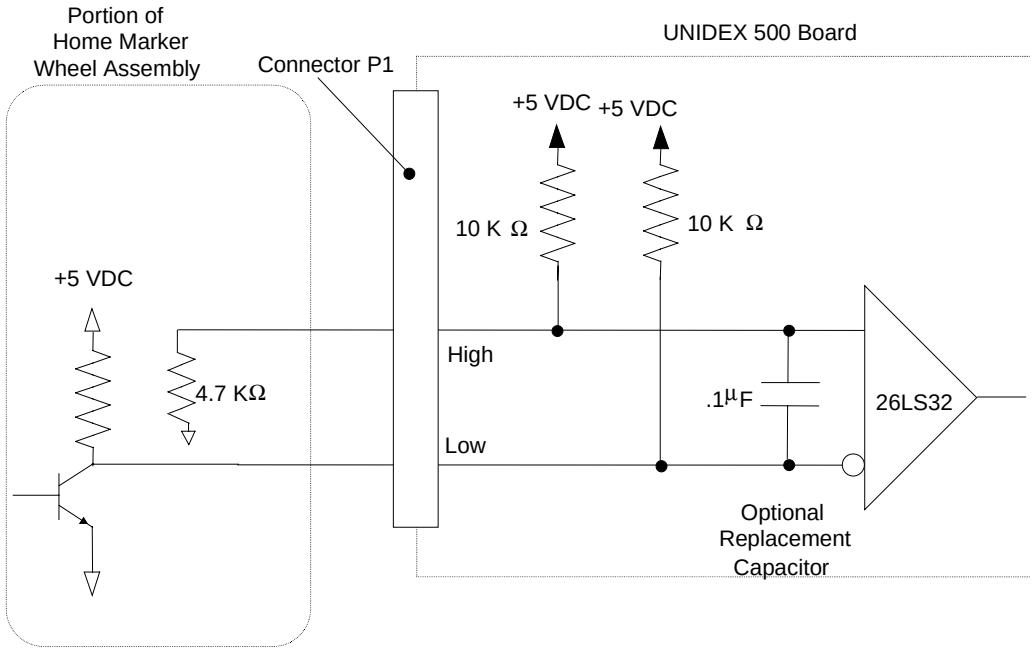


Figure 12-2. Electrical Characteristics of an Aerotech Stepper Motor Home Marker Option

Resistor networks RN1, RN2, and RN3 provide termination resistors for axes 1, 2, 3, and 4. The following tables show the important configuration information for each individual axis. Included are the resistor network number, main pinouts (terminal P1), axis signals, and resistor network pin numbers. See Chapter 11 for more information about encoder types.

Table 12-1. Termination Resistor Configuration for Axis 1 Encoders

Axis Signals	ISA RN#-pin	PCI RN#-pin	Main Pinouts	180 Ω Resistor *	0.1 μF Capacitor *
MRK1-	RN3-2	RN1-4	12		
MRK1+	RN3-1	RN1-3	11		
SIN1-	RN2-2	RN1-8	8		
SIN1+	RN2-1	RN1-7	7		
COS1-	RN2-4	RN1-6	10		
COS1+	RN2-3	RN1-5	9		

* Use a 180 Ω termination resistor for standard differential encoders. When using Aerotech stepper motors with the home marker wheel option, you must replace the termination resistor with a 0.1 μF capacitor.

Table 12-2. Termination Resistor Configuration for Axis 2 Encoders

Axis Signals	ISA RN#-pin	PCI RN#-pin	Main Pinouts	180 Ω Resistor *	0.1 μF Capacitor *
MRK2-	RN3-4	RN2-6	20		
MRK2+	RN3-3	RN2-5	19		
SIN2-	RN2-6	RN1-1	16		
SIN2+	RN2-5	RN1-2	15		
COS2-	RN2-8	RN2-8	18		
COS2+	RN2-7	RN2-7	17		

* Use a 180 Ω termination resistor for standard differential encoders. When using Aerotech stepper motors with the home marker wheel option, you must replace the termination resistor with a 0.1 μF capacitor.

Table 12-3. Termination Resistor Configuration for Axis 3 Encoders

Axis Signals	ISA RN#-pin	PCI RN#-pin	Main Pinouts	180 Ω Resistor *	0.1 μF Capacitor *
MRK3-	RN3-6	RN3-8	28		
MRK3+	RN3-5	RN3-7	27		
SIN3-	RN1-2	RN2-4	24		
SIN3+	RN1-1	RN2-3	23		
COS3-	RN1-4	RN2-1	26		
COS3+	RN1-3	RN2-2	25		

* Use a 180 Ω termination resistor for standard differential encoders. When using Aerotech stepper motors with the home marker wheel option, you must replace the termination resistor with a 0.1 μF capacitor.

Table 12-4. Termination Resistor Configuration for Axis 4 Encoders

Axis Signals	ISA RN#-pin	PCI RN#-pin	Main Pinouts	180 Ω Resistor *	0.1 μF Capacitor *
MRK4-	RN3-8	RN3-1	36		
MRK4+	RN3-7	RN3-2	35		
SIN4-	RN1-6	RN3-6	32		
SIN4+	RN1-5	RN3-5	31		
COS4-	RN1-8	RN3-4	34		
COS4+	RN1-7	RN3-3	33		

* Use a 180 Ω termination resistor for standard differential encoders. When using Aerotech stepper motors with the home marker wheel option, you must replace the termination resistor with a 0.1 μF capacitor.

12.1.2. Encoder Signal Pinouts

Table 12-5 identifies the encoder signals and the corresponding P1 connector pin number and termination locations.

Table 12-5. Encoder Signals and Pinouts

Signal Name	P1 Pin Number	ISA Termination Location	PCI Termination Location
Channel 1			
Sine, Positive	7	RN2-1	RN1-7
Sine, Negative	8	RN2-2	RN1-8
Cosine, Positive	9	RN2-3	RN1-5
Cosine, Negative	10	RN2-4	RN1-6
Marker, Positive	11	RN3-1	RN1-3
Marker, Negative	12	RN3-2	RN1-4
Channel 2			
Sine, Positive	15	RN2-5	RN1-2
Sine, Negative	16	RN2-6	RN1-1
Cosine, Positive	17	RN2-7	RN2-7
Cosine, Negative	18	RN2-8	RN2-8
Marker, Positive	19	RN3-3	RN2-5
Marker, Negative	20	RN3-4	RN2-6
Channel 3			
Sine, Positive	23	RN1-1	RN2-3
Sine, Negative	24	RN1-2	RN2-4
Cosine, Positive	25	RN1-3	RN3-2
Cosine, Negative	26	RN1-4	RN3-1
Marker, Positive	27	RN3-5	RN3-7
Marker, Negative	28	RN3-6	RN3-8
Channel 4			
Sine, Positive	31	RN1-5	RN3-5
Sine, Negative	32	RN1-6	RN3-6
Cosine, Positive	33	RN1-7	RN3-3
Cosine, Negative	34	RN1-8	RN3-4
Marker, Positive	35	RN3-7	RN3-2
Marker, Negative	36	RN3-8	RN3-1

12.1.3. Limit and Amplifier Fault Inputs

The UNIDEX 500 contains three limit inputs per axis; two over–travel and one home. Also, each axis has one amplifier fault input.

The inputs are TTL level signals pulled up to +5 V with a 10K ohm resistor. Open collector drivers or opto-isolators are the preferred electrical interface to this bus. Refer to Figure 12-3 for electrical characteristics of the limit/amplifier input.



To avoid damage to the UNIDEX 500, the input level should never exceed +5 V or go below 0 V.

The active polarity of the limit inputs is software selectable. The amplifier fault input polarity is programmable. Limit and amplifier fault inputs are summarized in Table 12-6.

Table 12-6. Limit and Amplifier Fault Inputs

Axis	Function	Signal	Location
1	Clockwise Rotation Limit Switch	CW1	P1 - 39
	Counter-Clockwise Rotation Limit Switch	CCW1	P1 - 40
	Home Limit Switch	HOME1	P1 - 47
	Amplifier Fault	AFAULT1	P1 - 73
2	Clockwise Limit Switch	CW2	P1 - 41
	Counter-Clockwise Rotation Limit Switch	CCW2	P1 - 42
	Home Limit Switch	HOME2	P1 - 48
	Amplifier Fault	AFAULT2	P1 - 74
3	Clockwise Limit Switch	CW3	P1 - 43
	Counter-Clockwise Rotation Limit Switch	CCW3	P1 - 44
	Home Limit Switch	HOME3	P1 - 49
	Amplifier Fault	AFAULT3	P1 - 75
4	Clockwise Limit Switch	CW4	P1 - 45
	Counter-Clockwise Rotation Limit Switch	CCW4	P1 - 46
	Home Limit Switch	HOME4	P1 -50
	Amplifier Fault	AFAULT4	P1 - 76

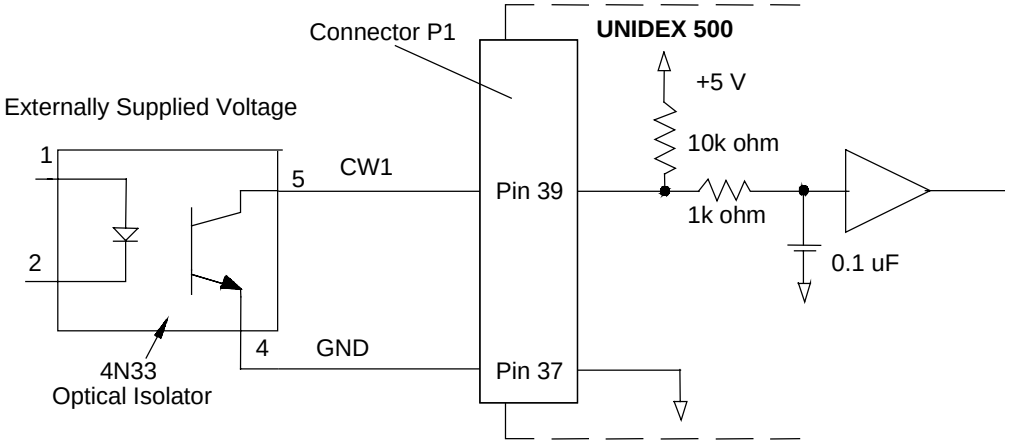


Figure 12-3. Electrical Characteristics of the UNIDEX 500 Limit Interface

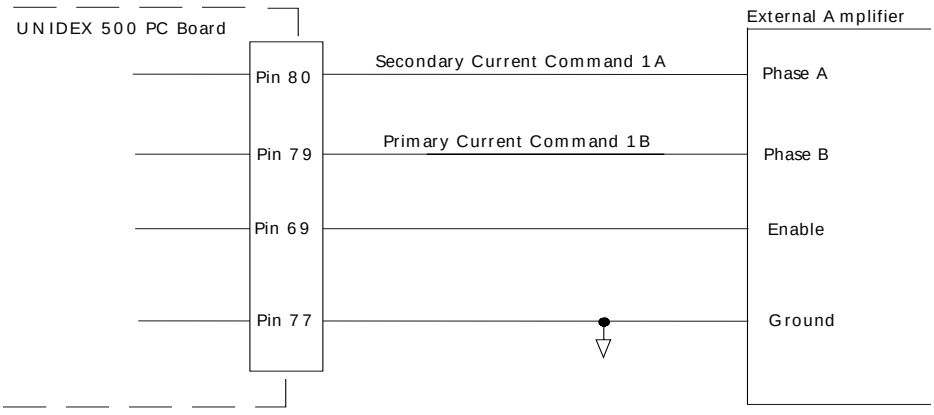
12.1.4. Current Command Output

The UNIDEX 500 has eight 16 bit current command outputs. The output range is +/- 10 V into a 10K ohm load. AC brushless motors and steppers require two current phases. Brush motors or self commutating amplifiers require only one. Refer to Figure 12-4 and Table 12-7 for electrical characteristics of the current command interface.

The UNIDEX 500 must always be connected to an amplifier with an opto-isolated power stage.



To avoid damage to the system, the UNIDEX 500 must not be connected to an amplifier with a non-isolated power stage.



NOTE: Current Command A and B are required for brushless motors and stepper motors.
Current Command B only for brush motors.

Figure 12-4. Electrical Characteristics of the UNIDEX 500 Current Command Output

Table 12-7. Current Command Output Signals and Pin Locations

Axis	Function	Signal	Location
Axis 1	Primary Current Command B	ICMD1B	P1-79
	Secondary Current Command A	ICMD1A	P1-80
Axis 2	Primary Current Command B	ICMD2B	P1-81
	Secondary Current Command A	ICMD2A	P1-82
Axis 3	Primary Current Command B	ICMD3B	P1-83
	Secondary Current Command A	ICMD3A	P1-84
Axis 4	Primary Current Command B	ICMD4B	P1-85
	Secondary Current Command A	ICMD4A	P1-86

12.1.5. Digital Input Bus Specifications

The UNIDEX 500 ISA card has 16 inputs. The PCI card has up to an additional 24 inputs. See Chapter 7, IO and IOSET commands for more details. The inputs are TTL level signals pulled up to +5 V with a 10K ohm resistor. Open collector drivers or opto-isolators are the preferred electrical interface to this bus. Refer to Figure 12-5 for electrical characteristics of the input bus. See Table 12-8 for UNIDEX 500 inputs and locations.



To avoid damage to the UNIDEX 500, the input level should never exceed +5 V or go below 0 V.

Inputs IN0-IN3 are accessible through both the main P1 connector and the P5 connector. On the ISA card, inputs IN4 through IN15 are dual purpose inputs, also being used as Hall effect inputs for AC brushless motors with encoder feedback. The PCI card can use these inputs for Hall effect inputs or can use the dedicated Hall effect lines on the P1 connector. Set general parameter 99 (Option board setup code) bit 3 to 1 (8) to read the hall signals through the OP500 cable connected to P1. If this bit is set to 0, the PCI card will read the hall signals through the P5 connector.



Inputs IN4 through IN15 may be used for only one function at a time.

The UNIDEX 500's P5 connector is compatible with a PB24 interface board. Inputs are read by using the "$INP" or "$INx" commands. Unused inputs are read as "1".

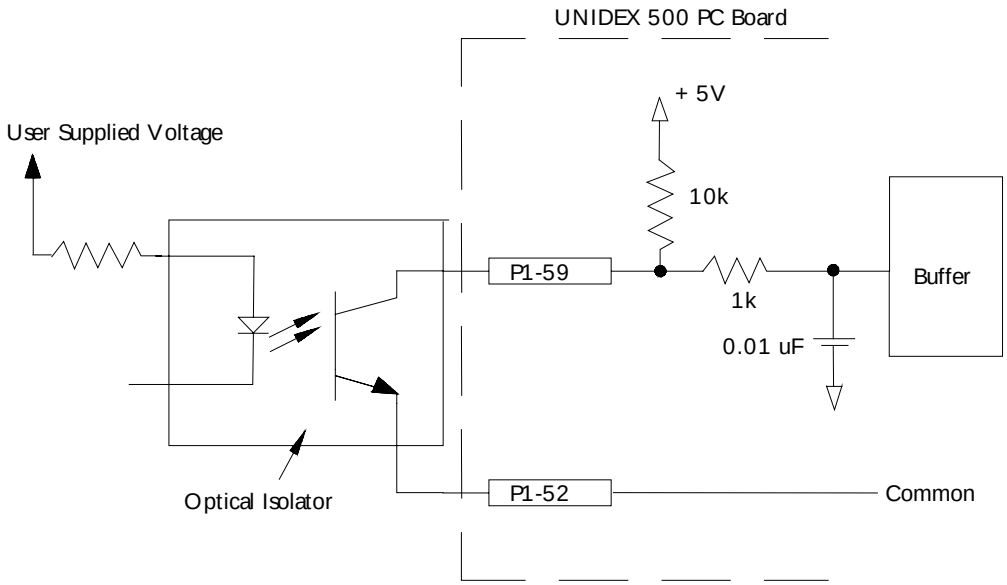


Figure 12-5. Electrical Characteristics of the UNIDEX 500 Input Bus Interface

Table 12-8. UNIDEX 500 Inputs and Locations

Function	Location	Function	Location	Function	Location	Function	Location
IN0	P5 - 31 / P1 - 59	IN5	P5 - 21	IN10	P5 - 11	IN15	P5 - 1
IN1	P5 - 29 / P1 - 60	IN6	P5 - 19	IN11	P5 - 9	GND	P5 2-50 (even)
IN2	P5 - 27 / P1 - 61	IN7	P5 - 17	IN12	P5 - 7	+5 V	P5 - 49
IN3	P5 - 25 / P1 - 62	IN8	P5 - 15	IN13	P5 - 5		
IN4	P5 - 23	IN9	P5 - 13	IN14	P5 - 3		

12.1.6. Output Bus Specifications

The UNIDEX 500 output bus consists of 8 open-collector outputs, which are set by the "OU" command. The PCI card has up to an additional 24 outputs available. See Chapter 7, IO and IOSET commands for more details. All outputs are tristated (high impedance) on reset. The eight outputs (OUT0-OUT7) are accessible through the P5 connector. Four of these outputs (OUT0-OUT3) are also accessible through the P1 connector. The P5 connector is compatible with a PB24 interface board. Table 12-9 shows the pinouts for these outputs.

Table 12-9. UNIDEX 500 Output Pinouts

Output Line	Location	Output Line	Location
OUT0	P5 – 47 / P1 - 63	OUT5	P5 - 37
OUT1	P5 – 45 / P1 - 64	OUT6	P5 - 35
OUT2	P5 – 43 / P1 - 65	OUT7	P5 - 33
OUT3	P5 – 41 / P1 - 66	GND	P5 - 2-50 (even pins)*
OUT4	P5 - 39	+5 V	P5 – 49 / P1 - 3, 4

*Note: See P1 Connector Pinouts for GND pins.

The electrical specifications for the output bus open-collector outputs (OUT0 – OUT7):

U500 ISA:

- Output Device→ 74LS642 (open-collector octal bus transceiver)
- Output ON→24 milliamps sink current with a max. voltage drop of 0.5 V
- Output OFF (high impedance)→Maximum collector voltage of 5.25 V

U500 PCI:

- Output Device→7406 (open-collector)
- Output ON→24 milliamps sink current
- Output OFF (high impedance)→Maximum collector voltage of 24 V

U500 PCI – 8x3 IO (P4)

- IO Device→74ACT16652
- Outputs source or sink 24 milliamps at 5 V

For improved reliability and noise immunity, the outputs should be optically isolated. The example in Figure 12-6 shows output "OUT0" connected to an optocoupler (4N33) and a current limiting resistor**.**

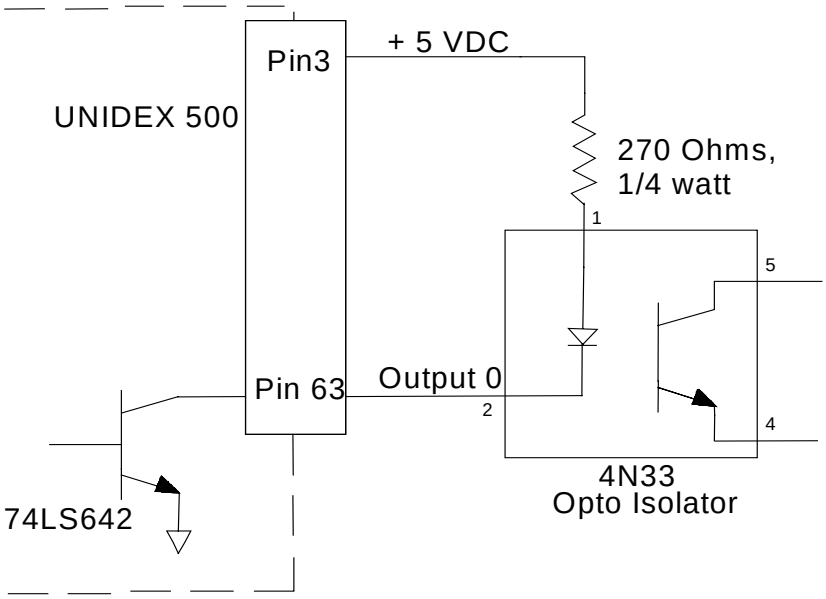


Figure 12-6. Example Optocoupled Output Bus Interface

12.1.7. The Brake Output

The UNIDEX 500 is equipped with a fail–safe brake signal output using a high voltage open collector driver (7407). Refer to Figure 12-7.

When the UNIDEX 500 is in the reset state, this output is in the high impedance state. When the brake is activated, this signal is pulled low. This output signal is referenced to the UNIDEX 500 signal common.

The recommended opening conditions for the brake output signal are as follows:

```
Maximum Voltage          24 V
Maximum Current Sink     24 mA
```

The brake signal is output through pin 94 of the P1 connector.

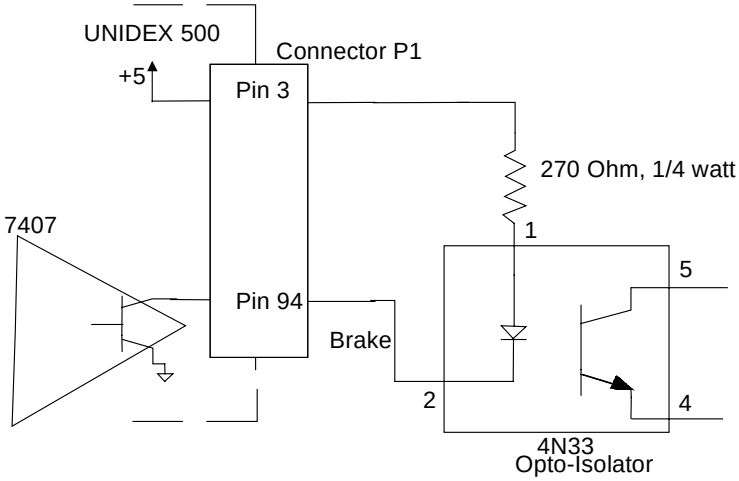


Figure 12-7. Electrical Characteristics of the UNIDEX 500 Brake Signal Output

12.1.8. Opto 22 I/O Bus

The UNIDEX 500's P5 connector provides a simple interface to the PB24 Opto 22 Interface Board (8 out/16 in).

The UNIDEX 500's P5 connector is a locking header, the PB24 connector is a standard edge type connector. A one-to-one ribbon cable can be made to connect the two. The mating connectors are:

- 50 pin ribbon cable header P/N# 3M 3425-6050
- Aerotech P/N ECK 332 (or equivalent)

- 50 pin cable edge connector P/N# 3M 3415-0001
- Aerotech P/N ECK 310 (or equivalent)

Refer to Table 12-10 for connection information. Refer to Figure 12-8 for an illustration of the electrical characteristics of the Opto 22 interface.

The following table lists the UNIDEX 500 Interface to the PB8, PB16 and the PB24 Opto 22 Interface Boards.

Table 12-10. UNIDEX 500/Opto 22 Connection Information

Interface Cable Assembly (model OPC)	PB8, PB16A, PB16C and PB24 Board

Opto Interface (P5)	Control Connection (edge connector on Opto board)	Module Position	Connection Description	Type of Module	Field Connection (barrier strip)
49	49		+5V int supply		
47	47	0	Out 0	output	1 and 2
45	45	1	Out 1	output	3 and 4
43	43	2	Out 2	output	5 and 6
41	41	3	Out 3	output	7 and 8
39	39	4	Out 4	output	9 and 10
37	37	5	Out 5	output	11 and 12
35	35	6	Out 6	output	13 and 14
33	33	7	Out 7	output	15 and 16
31	31	8	In 0	input	17 and 18
29	29	9	In 1	input	19 and 20
27	27	10	In 2	input	21 and 22
25	25	11	In 3	input	23 and 24
23 *	23	12	In 4	input	25 and 26
21 *	21	13	In 5	input	27 and 28
19 *	19	14	In 6	input	29 and 30
17 *	17	15	In 7	input	31 and 32
15 *	15	16	In 8	input	33 and 34
13 *	13	17	In 9	input	35 and 36
11 *	11	18	In 10	input	37 and 38
9 *	9	19	In 11	input	39 and 40

PB8, PB16A and PB16C

PB24

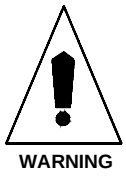
Table 12-10. UNIDEX 500/Opto 22 Connection Information (Cont'd)

Opto Interface (P5)	Control Connection (edge connector on Opto board)	Module Position	Connection Description	Type of Module	Field Connection (barrier strip)	
7 *	7	20	In 12	input	41 and 42	|
5 *	5	21	In 13	input	43 and 44	|
3 *	3	22	In 14	input	45 and 46	PB24
1 *	1	23	In 15	input	47 and 48	↓

* Note: These inputs are not available on AC brushless motors that use Hall effect switches.

All even pins (2-50) are signal common

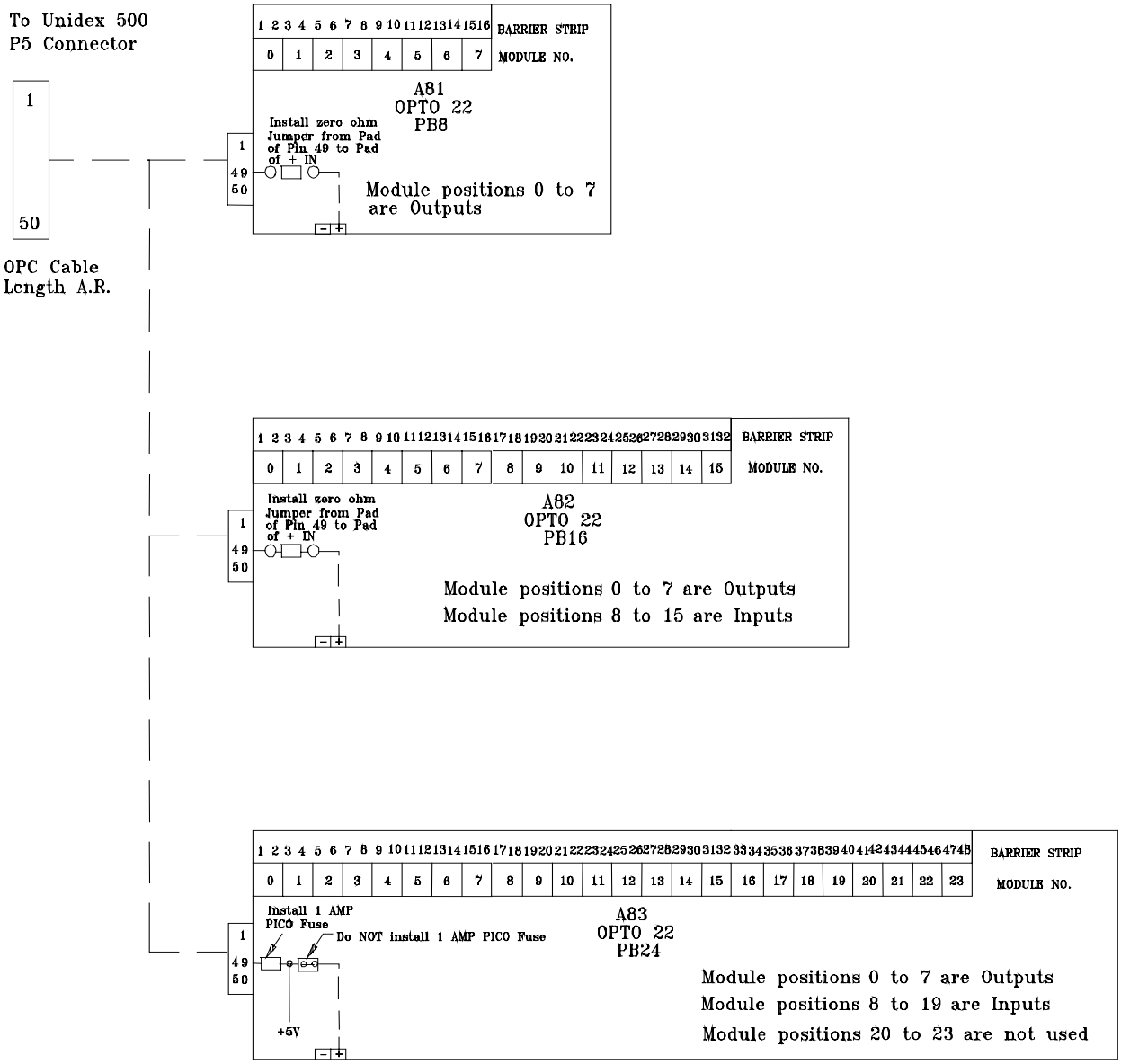
Typical Modules: IDC5, IDC5B, IAC5, IAC5A, ODC5, ODC5A, OAC5, OAC5A



WARNING ! Type of module (input or output) cannot be interchanged. To do so may damage the UNIDEX 500



Pin No. 2 on the U500 P5 connector is removed. This is done to key the connector.



NOTES:

50 Pin Ribbon Cable terminated at Opto 22 Board with 50 Pin Card Edge Connector
(3M P/N 3415-0001, Aerotech P/N ECK310)

Ribbon Cable terminated at Unidex 500 with a 50 Pin Header
(3M P/N 3425-6050, Aerotech P/N ECK 332)

Figure 12-8. Electrical Characteristics of the UNIDEX 500 Opto 22 Connections

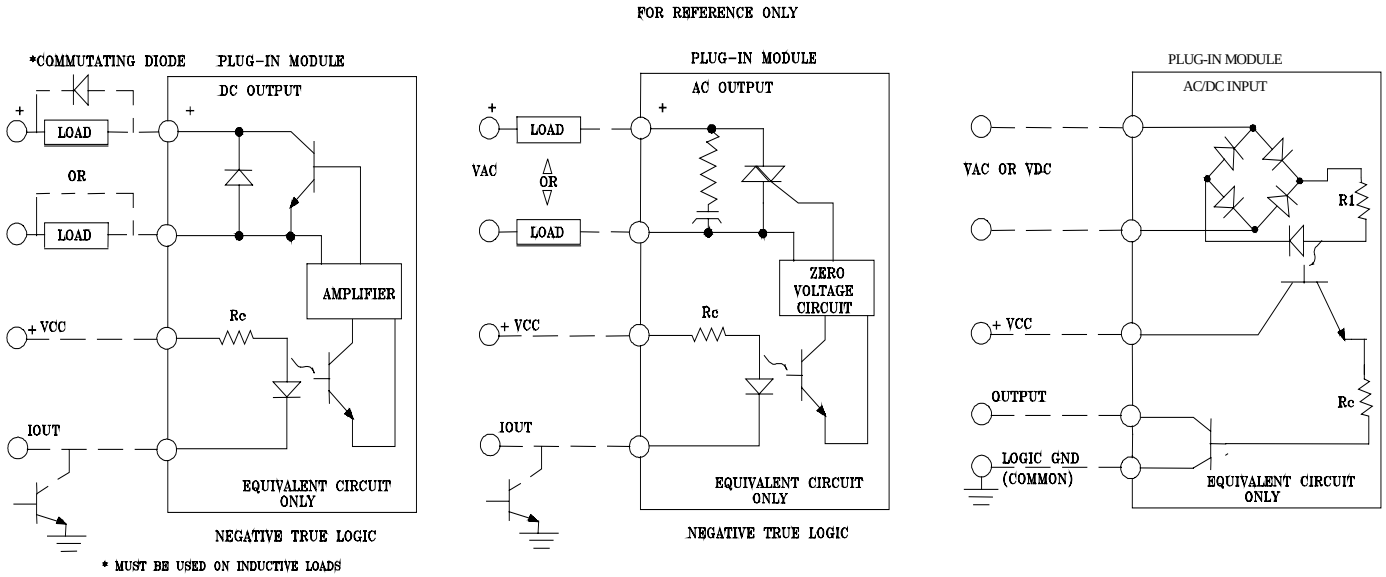


Figure 12-9. Electrical Characteristics of the UNIDEX 500 Opto 22 Connections (page 2)

12.1.9. Main Connector Pinout of the UNIDEX 500

The UNIDEX 500's main interface connector (P1) is accessible from the rear of the PC. The connector is a 100-pin "AMPLIMITE" high density female connector. The mating connector is an "AMPLIMITE" connector, part number 759879-9. This connector accepts two 50-pin ribbon cables and is non-shielded. Refer to Table 12-11 for the main connector pinouts, and Table 12-12 for the analog input locations.

Table 12-11. Main Connector Pinouts for the UNIDEX 500 ISA

Pin	Function	Description	Pin	Function	Description
1	Interlock Send	ILOCKS(GND)	2	20 kHz Synchronization	SYNC+
3	+5 V	+5	4	+5 V	+5
5	Encoder Ground	GND	6	Encoder Ground	GND
7	Encoder Sine Positive, Axis 1	SIN1+	8	Encoder Sine Ground, Axis 1	SIN1-
9	Encoder Cosine Positive, Axis 1	COS1+	10	Encoder Cosine Ground, Axis 1	COS1-
11	Marker Pulse, Axis 1	MRK1+	12	Marker Pulse, Axis 1	MRK1-
13	Encoder Ground	GND	14	Encoder Ground	GND
15	Encoder Sine Positive, Axis 2	SIN2+	16	Encoder Sine Ground, Axis 2	SIN2-
17	Encoder Cosine Positive, Axis 2	COS2+	18	Encoder Cosine Ground, Axis 2	COS2-
19	Marker Pulse, Axis 2	MRK2+	20	Marker Pulse, Axis 2	MRK2-
21	Encoder Ground	GND	22	Encoder Ground	GND
23	Encoder Sine Positive, Axis 3	SIN3+	24	Encoder Sine Ground, Axis 3	SIN3-
25	Encoder Cosine Positive, Axis 3	COS3+	26	Encoder Cosine Ground, Axis 3	COS3-
27	Marker Pulse, Axis 3	MRK3+	28	Marker Pulse, Axis 3	MRK3-
29	Encoder Ground	GND	30	Encoder Ground	GND
31	Encoder Sine Positive, Axis 4	SIN4+	32	Encoder Sine Ground, Axis 4	SIN4-
33	Encoder Cosine Positive, Axis 4	COS4+	34	Encoder Cosine Ground, Axis 4	COS4-
35	Marker Pulse, Axis 4	MRK4+	36	Marker Pulse, Axis 4	MRK4-
37	Encoder Ground	GND	38	Encoder Ground	GND
39	Clockwise Limit, Axis 1	CW1	40	Counter clockwise Limit, Axis 1	CCW1
41	Clockwise Limit, Axis 2	CW2	42	Counter clockwise Limit, Axis 2	CCW2
43	Clockwise Limit, Axis 3	CW3	44	Counter clockwise Limit, Axis 3	CCW3
45	Clockwise Limit, Axis 4	CW4	46	Counter clockwise Limit, Axis 4	CCW4
47	Home Limit, Axis 1	HOME1	48	Home Limit, Axis 2	HOME2
49	Home Limit, Axis 3	HOME3	50	Home Limit, Axis 4	HOME4
51	20 kHz Synchronization	SYNC-	52	Limits Ground	GND
53	+12 V	+12	54	+12 V	+12
55	-12 V	-12	56	-12 V	-12
57	Recirc. Mode Axis 1 (*Reserved*)	MODE1	58	Recirc. Mode Axis 2 (*Reserved*)	MODE2
59	Input 0	IN0	60	Input 1	IN1
61	Input 2	IN2	62	Input 3	IN3
63	Output 0	OUT0	64	Output 1	OUT1
65	Output 2	OUT2	66	Output 3	OUT3
67	Recirc. Mode Axis 3 (*Reserved*)	MODE3	68	Recirc. Mode Axis 4 (*Reserved*)	MODE4
69	Amplifier Enable 1	AEN1	70	Amplifier Enable 2	AEN2
71	Amplifier Enable 3	AEN3	72	Amplifier Enable 4	AEN4
73	Amplifier Fault 1	AFLT1	74	Amplifier Fault 2	AFLT2
75	Amplifier Fault 3	AFLT3	76	Amplifier Fault 4	AFLT4
77	Limits Ground	GND	78	Limits Ground	GND
79	Axis 1 Primary Current Command	ICMD1B	80	Axis 1 Secondary Current Command	ICMD1A
81	Axis 2 Primary Current Command	ICMD2B	82	Axis 2 Secondary Current Command	ICMD2A
83	Axis 3 Primary Current Command	ICMD3B	84	Axis 3 Secondary Current Command	ICMD3A
85	Axis 4 Primary Current Command	ICMD4B	86	Axis 4 Secondary Current Command	ICMD4A
87	Ground	GND	88	Ground	GND
89	Joystick Potentiometer 1 Input	JSW1	90	Joystick Potentiometer 2 Input	JSW2
91	Joystick Button A Input	JSA	92	Joystick Button B Input	JSB
93	Joystick Button C Input	JSC (JS ILOCK)	94	Brake Output	BRAKE
95	Analog Input 0	AIN0	96	Analog Input 1	AIN1
97	Emergency Stop	ESTOP	98	User Interrupt	UINT
99	Opto Isolator Anode (for 97 &98)	OPTOA	100	Interlock Receive	ILOCKR

Table 12-12. Analog Input Locations for the UNIDEX 500 ISA

P1 Connector	U500 Command	U500 Tuning Window	Diagnostic Window	BB500 Pin/Name	BB501 Pin/Name	DR300 Pin/Name	DR500 Pin/Name	DR600 Pin/Name
P1-96	$AD0	Analog(1)	1	P5-19/AIN1	TB1-4/AIN1	J13-25/AIN1	J13-25/AIN1	J13-25/AIN1
P1-95	$AD1	Analog(2)	2	P5-18/AIN0	TB1-3/AIN0	J13-24/AIN0	J13-24/AIN0	J13-24/AIN0
P1-90	$AD2	Analog(3)	3	P5-21/JSW2	J12-6/JSW2	J12-6/JSW2	J12-6/JSW2	J12-6/JSW2
P1-89	$AD3	Analog(4)	4	P5-20/JSW1	J12-3/JSW1	J12-3/JSW1	J12-3/JSW1	J12-3/JSW1

12.1.10. Opto 22 I/O Hall Inputs (P5) Pinouts

Table 12-13 lists the pinouts for the OPTO 22 I/O Hall inputs (P5). The PCI card has the Hall input signals available on the 100-pin connector P1. General parameter 99 selects where the Hall signals are connected for the PCI card. See Chapter 5, Parameters for more information.

Table 12-13. Pinouts for Opto 22 I/O Hall Inputs (P5)

Pin	Description	Pin	Description
1	IN15 - HALL4C	2	GND/Key
3	IN14 - HALL4A	4	GND
5	IN13 - HALL4B	6	GND
7	IN12 - HALL3C	8	GND
9	IN11 -HALL3A	10	GND
11	IN10 - HALL3B	12	GND
13	IN9 - HALL2C	14	GND
15	IN8 - HALL2A	16	GND
17	IN7 - HALL2B	18	GND
19	IN6 - HALL1C	20	GND
21	IN5 - HALL1A	22	GND
23	IN4 - HALL1B	24	GND
25	IN3	26	GND
27	IN2	28	GND
29	IN1	30	GND
31	IN0	32	GND
33	OUT7	34	GND
35	OUT6	36	GND
37	OUT5	38	GND
39	OUT4	40	GND
41	OUT3	42	GND
43	OUT2	44	GND
45	OUT1	46	GND
47	OUT0	48	GND
49	Fused +5	50	GND

12.2. UNIDEX 500 ISA Technical Details

12.2.1. Test Points

Test points are located throughout the UNIDEX 500 control board. They aid in troubleshooting the control board and gaining easy access to the UNIDEX 500 signals.

This chapter arranges test points into functional groups. These functional groups are divided as follows:

- Motor Related Test Points
- PC Interface to the UNIDEX 500
- DSP 56002 Chip Test Points
- Miscellaneous Test Points

Test points for these functional groups listed and explained in tables that follow.

Table 12-14. Motor Related Test Points

Test Point	Meaning
TP25	Axis 1 marker pulse
TP26	Axis 2 marker pulse
TP27	Axis 3 marker pulse
TP28	Axis 4 marker pulse
P6-1	Channel 1 sine after RS-422 receiver
P6-2	Channel 1 cosine after RS-422 receiver
P6-5	Channel 2 sine after RS-422 receiver
P6-6	Channel 2 cosine after RS-422 receiver
P6-9	Channel 3 sine after RS-422 receiver
P6-10	Channel 3 cosine after RS-422 receiver
P6-13	Channel 4 sine after RS-422 receiver
P6-14	Channel 4 cosine after RS-422 receiver
TP16	Axis 1 primary current command output (ICMD1B)
TP20	Axis 1 secondary current command output (ICMD1A)
TP17	Axis 2 primary current command output (ICMD2B)
TP21	Axis 2 secondary current command output (ICMD2A)
TP18	Axis 3 primary current command output (ICMD3B)
TP22	Axis 3 secondary current command output (ICMD3A)
TP19	Axis 4 primary current command output (ICMD4B)
TP23	Axis 4 secondary current command output (ICMD4A)

Table 12-15. PC Interface to the UNIDEX 500

Test Point	Meaning
TP10	Host enable active low. Indicates PC is attempting to communicate with U500 (hen_n).
TP9	Host read/write. Indicates direction of PC/UNIDEX 500 communication (hr/w_n).
TP15	I/O board base address true (active low).

Table 12-16. DSP 56002 Chip Test Points

Test Point	Meaning	Test Point	Meaning
TP7	Timer input/output	TP32	MDPRAM-N
TP11	YPERF	TP33	MBSYR-N
TP12	IRQA	TP34	MIRQR-N
TP13	IRQB	TP35	SMEMSL-N
TP14	NMI	TP41	MAE-N
TP29	MIRQL-N	TP42	BSMEMWR-N
TP30	MBSYL-N	TP43	BSMEMRD-N
TP31	MCEL-N		

Table 12-17. Miscellaneous Test Points

Test Point	Meaning
TP4	Signal common
TP5	SCIOUT
TP6	SCIIN
TP8	ESTOP after opto-isolator
TP24	Brake output signal before O.C. buffer. Brake is off when low
TP39	A/D channel 4 (Joystick, $AD3)
TP40	A/D channel 3 (Joystick, $AD2)
TP37	A/D channel 2 (Spare, $AD1)
TP38	A/D channel 1 (MFO, $AD0)
TP36	A/D select

12.2.2. Jumper Configurations

The following tables summarize all of the jumpers on the UNIDEX 500 PC Board. Relative Jumper locations and the default settings are illustrated in Figure 12-10. An asterisk (*) following a jumper setting indicates the default position.

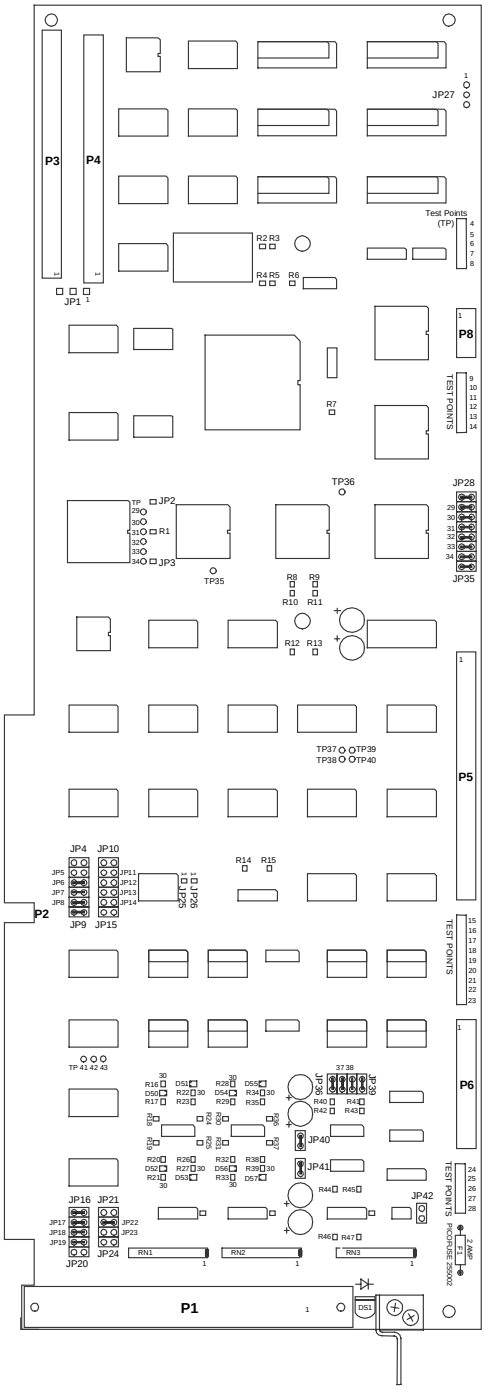


Figure 12-10. UNIDEX 500 PC Board Jumper Locations

12.2.2.1. Marker Pulse Qualification Jumpers (JP28 – JP35)

The marker input signal of an encoder is used as a reference pulse during the homing cycle of the UNIDEX 500 (the UNIDEX 500 expects to see a logical "1" marker). In practice, the stage approaches the home position until it reaches the home limit switch. Then the stage direction reverses until the marker pulse is found.

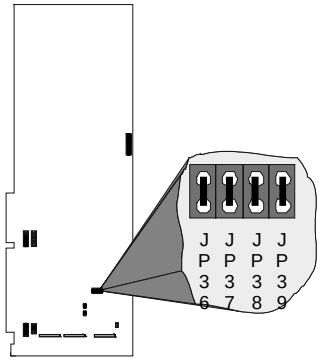
UNIDEX 500 can logically AND the marker with the cosine, inverse cosine, sine AND cosine, or nothing. The marker pulse qualification jumpers determine which signal the marker is paired with. The settings for the marker pulse qualification jumpers are shown in Table 12-18. The locations of the UNIDEX 500 control board jumpers are shown in Figure 12-10.

In applications using an external multiplier box (e.g., the MX multiplier), the marker qualification should not be performed.

Table 12-18. Marker Pulse Qualification Jumper Settings

Axis 1	JP28	JP29	Axis 1 Marker Qualification
	IN	IN	none (default)
	IN	OUT	AND with cosine
	OUT	IN	AND with inverse cosine
	OUT	OUT	AND with (cosine AND sine)
Axis 2	JP30	JP31	Axis 2 Marker Qualification
	IN	IN	none (default)
	IN	OUT	AND with cosine
	OUT	IN	And with inverse cosine
	OUT	OUT	AND with (cosine AND sine)
Axis 3	JP32	JP33	Axis 3 Marker Qualification
	IN	IN	none (default)
	IN	OUT	AND with cosine
	OUT	IN	AND with inverse cosine
	OUT	OUT	AND with (cosine AND sine)
Axis 4	JP34	JP35	Axis 4 marker qualification
	IN	IN	none (default)
	IN	OUT	AND with cosine
	OUT	IN	AND with inverse cosine
	OUT	OUT	AND with (cosine AND sine)

12.2.2.2. Amplifier Enable Jumpers (JP36 – JP39)



If an Aerotech DR500 drive rack has been purchased with the U500 system, the U500 control board is factory configured for the amplifiers. If an Aerotech drive rack is not being used, the amplifier enable jumpers may need to be reconfigured. It is vital for the user to verify these jumper settings. This safety measure is necessary to ensure that all amplifiers default to disabled (off) after the UNIDEX 500 is initially powered up or reset.

The UNIDEX 500 control board has four amplifier enable outputs, each corresponding to a particular axis of the system. Each axis (X, Y, Z, and U) has an associated active polarity jumper (JP39-JP36, respectively) that is selectable. The UNIDEX 500 control board is shipped with the amplifier enable jumpers installed. In this configuration, the amplifier enable outputs are in the high impedance state during reset and pulled low when the drive is enabled. These outputs are also in the high impedance state when the axis is disabled by software. The axes are software disabled by default when the system is initialized or reset. Select a polarity for each axis so the associated amplifier is not active when the UNIDEX 500 is in a reset state.

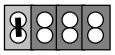
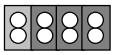
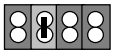
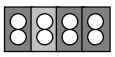
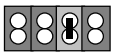
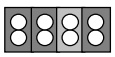
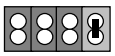
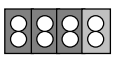


Amplifier polarity should be set so the amplifiers are disabled after a system reset. The amplifiers should remain disabled until the ENABLE command is entered.

Amplifier enable jumpers JP36-JP39 are located near the P6 connector of the control board. The settings for these jumpers are listed in Table 12-19. This table shows each of the four jumpers (JP39-JP36), the corresponding axes (X, Y, Z, and U [1-4]), an illustration of both “IN” and “OUT” jumper settings, and the corresponding open collector amplifier output function (i.e., the impedance) during reset and when enabled.

Each axis has one 7406-type, open collector driver with absolute maximum ratings of 24 volts and 24 mA sink capability.

Table 12-19. Amplifier Enable Jumper Settings

Jumper	Axis	In/Out	Settings	Output Function (Impedance)
JP36	4 (U)	IN (default)	JP36 	High (no connection) during reset, Low (pulled to ground) when enabled
		OUT	JP36 	Low (pulled to ground) during reset, High (no connection) when enabled
JP37	3 (Z)	IN (default)	JP36 	High (no connection) during reset, Low (pulled to ground) when enabled
		OUT	JP36 	Low (pulled to ground) during reset, High (no connection) when enabled
JP38	2 (Y)	IN (default)	JP36 	High (no connection) during reset, Low (pulled to ground) when enabled
		OUT	JP36 	Low (pulled to ground) during reset, High (no connection) when enabled
JP39	1 (X)	IN (default)	JP36 	High (no connection) during reset, Low (pulled to ground) when enabled
		OUT	JP36 	Low (pulled to ground) during reset, High (no connection) when enabled

If an Aerotech drive rack is not being used, the operator may need to reconfigure the amplifier enable jumpers for the amplifiers being used. To determine the appropriate amplifier enable jumper settings for each axis, the operator must review the specifications of his amplifiers. Specifically, he must determine the impedance (high or low) that is required to enable the amplifier. For amplifiers that require a low impedance to be enabled (and therefore a high impedance during reset), the user should keep the associated amplifier enable jumper installed. Conversely, for amplifiers that require a high impedance to be enabled (and therefore a low impedance during reset), the user should remove the associated amplifier enable jumper. Refer to the amplifier's documentation for the necessary information.

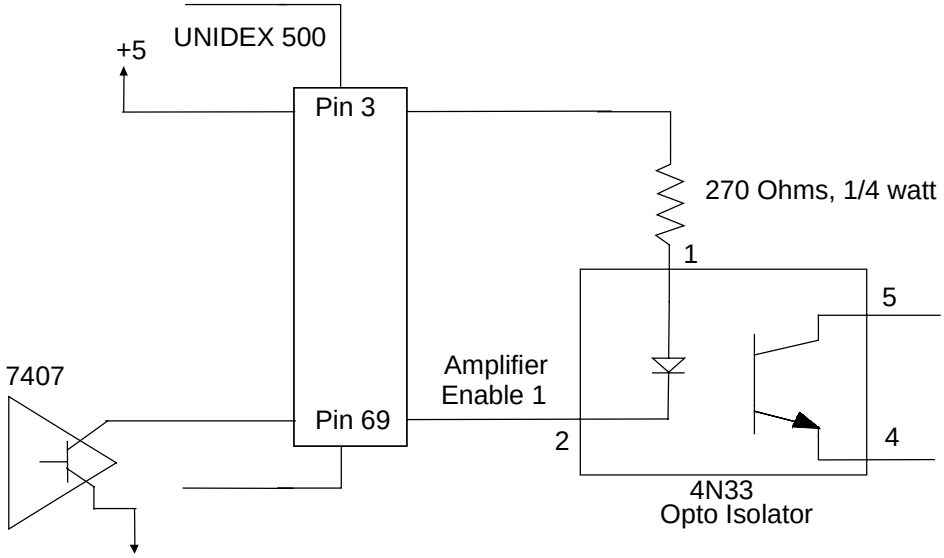


Figure 12-11. Typical Connection of the U500 to an Opto-Isolator

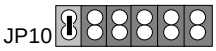
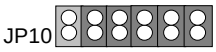
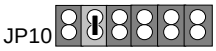
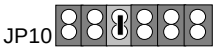
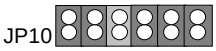
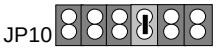
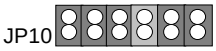
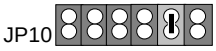
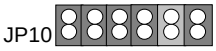
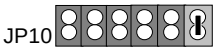
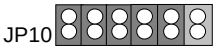
12.2.2.3. PC Bus Interrupt Jumpers (JP10 – JP15)

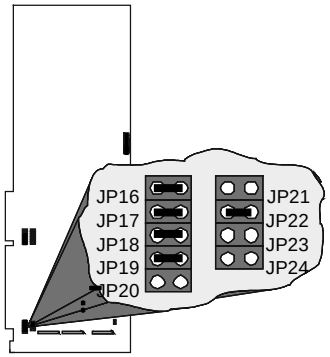
The UNIDEX 500 can be used to generate an interrupt request to the host PC. The UNIDEX 500 interface software does not use this interrupt feature, but custom applications may. The interrupt level is jumper selectable and is outlined in Table 12-20. This table shows the available interrupt request (IRQ) lines that may be assigned using the PC bus interrupt jumpers. The locations of the UNIDEX 500 control board jumpers are shown in Figure 12-10.

The jumper-selected PC bus interrupt can be generated using the command interrupt (CI) programming command or through parameter x57 (*Interrupt PC*). Once these interrupts are generated, they can be cleared using the C library function WAPIAer Reset Intr, which resets the interrupt.

The default jumper configuration has all jumpers removed. In this configuration, the UNIDEX 500 does not send interrupt requests to the host computer.

Table 12-20. PC Bus Interrupt Jumper Settings

Jumper	State	Jumper Settings	Function
JP10	IN		Interrupt IRQ3 (AT unassigned)
	OUT (default)		IRQ3 not selected (default)
JP11	IN		Interrupt IRQ4 (AT unassigned)
	OUT (default)		IRQ4 not selected (default)
JP12	IN		Interrupt IRQ7 (AT unassigned)
	OUT (default)		IRQ7 not selected (default)
JP13	IN		Interrupt IRQ10 (LPT)
	OUT (default)		IRQ10 not selected (default)
JP14	IN		Interrupt IRQ11 (COM)
	OUT (default)		IRQ11 not selected (default)
JP15	IN		Interrupt IRQ12 (COM)
	OUT (default)		IRQ12 not selected (default)



12.2.2.4. Dual-Ported RAM Base Address Select Jumpers JP16 – JP24)

Jumpers JP17 through JP24 are used to select the address range for the PC/DSP dual-ported RAM. This option is not utilized on a factory standard board. A special order is required to implement this feature. The dual-ported RAM base address select jumper settings are shown in Table 12-21.

The position of the JP16 jumper determines whether the dual-ported RAM base address will be enabled or disabled. Removing the jumper (OUT) enables the base address selected by jumpers JP17 through JP24. Installing the JP16 jumper (IN) disables the address. The default for JP16 is IN, which disables this feature.

Table 12-21. Dual Ported RAM Base Address Select Jumpers Settings

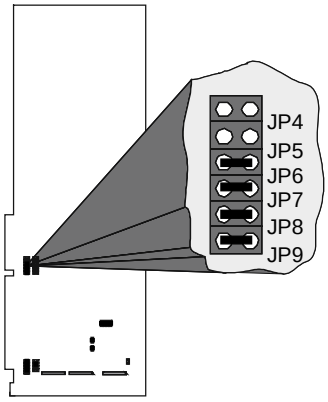
Base Address	Base Address Jumpers and Corresponding Address Lines								Enable Jumper	
	JP24 (A19)	JP23 (A18)	JP22 (A17)	JP21 (A16)	JP20 (A15)	JP19 (A14)	JP18 (A13)	JP17 (A12)	JP16	
									enable	disable (default)
C0000-C0FFF	OUT	OUT	IN	IN	IN	IN	IN	IN	OUT	IN
C8000-C8FFF	OUT	OUT	IN	IN	OUT	IN	IN	IN	OUT	IN
D0000-D0FFF	OUT	OUT	IN	OUT	IN	IN	IN	IN	OUT	IN
D8000-D8FFF (default)	OUT	OUT	IN	OUT	OUT	IN	IN	IN	OUT	IN
E0000-E0FFF	OUT	OUT	OUT	IN	IN	IN	IN	IN	OUT	IN
E8000-E8FFF	OUT	OUT	OUT	IN	OUT	IN	IN	IN	OUT	IN

Table 12-22. Miscellaneous Jumpers

Jumper	Description
JP1 1-2	iSBX MINT1 to DSP (trace jumper)
JP1 2-3	iSBX MINT0 to DSP (trace jumper)
JP1 removed*	no iSBX interrupt
JP2 IN (reserved)	HREQ_N interrupt enabled
JP2 OUT (reserved)*	HREQ_N interrupt disabled
JP3 IN (reserved)	SRDY_N ISA bus cycle
JP3 OUT (reserved)*	no SRDY_N ISA bus cycle
JP25 1-2 (reserved)*	enabled watchdog (trace jumper)
JP25 2-3 (reserved)	disable watchdog (trace jumper)
JP26 1-2 (reserved)*	PC drives reset (trace jumper)
JP26 2-3 (reserved)	PC does not drive reset (trace jumper)
JP40 IN*	PC bus +12V to D/A and P1-53,54
JP40 OUT	Need external +12V for D/A on P1-53,54
JP41 IN*	PC bus -12V to D/A and P1-55,56
JP41 OUT	Need external -12V for D/A on P1-55,56
JP42 IN	20khz_n drive sync. signal on P1-51
JP42 OUT*	no drive sync output
JP27 1-2 (reserved)	ram nc to +5V
JP27 2-3 (reserved)	ram nc to GND
JP27 removed (reserved)*	

* Default Jumper Position.

12.2.2.5. Base Address Jumpers (JP4 – JP9)



Input/Output (I/O) base addresses for the UNIDEX 500 are assigned in hexadecimal address ranges. The UNIDEX 500 control board occupies 16 consecutive memory locations in the I/O channel memory of the PC holding the UNIDEX 500 control board. The UNIDEX 500 control board is factory configured for default address 0x300-0x30F. The UNIDEX 500 interface software is also set to this default address. If the UNIDEX 500 control board does not initialize properly or exhibits sporadic operation, there may be another board in the system computer that is set to the same address. Use the diagnostic utility or CMOS setup program that comes with the PC to analyze which addresses are used, then try another UNIDEX 500 address. Change the U500 address by using the U500 Registry Editor and reboot for changes to take effect.

The map addresses for the system devices can be checked to insure that there are no conflicts. Follow these steps to view map addresses in Windows NT:

1. Select Administrative Tools from the Programs Menu
2. Select the Windows NT Diagnostics Program
3. Select the Resources Tab
4. Select the I/O Port Button to view the map addresses

To view the map addresses in Windows 95:

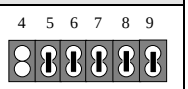
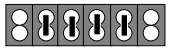
1. Select 'System' from the Control Panel
2. Select Device Manager
3. Highlight 'Computer' and select the Properties Button
4. Select the I/O button to view the map addresses

The address of a UNIDEX 500 board is set from jumpers JP4-JP9. These jumpers are located near the P2 connector of the control board. The pins of each jumper in the series (JP4 through JP9) are either connected ("IN") or disconnected ("OUT") to create a unique base address. The combinations of base address jumper settings are shown in Table 12-23. The locations of the UNIDEX 500 control board jumpers are shown in Figure 12-10.



Each UNIDEX 500 control board must have a distinct address in the same PC.

Table 12-23. Base Address Jumper Settings

PC I/O Base Address	JP4	JP5	JP6	JP7	JP8	JP9	Settings
200 - 20F	OUT	IN	IN	IN	IN	IN	
210 - 21F	OUT	IN	IN	IN	IN	OUT	
300 - 30F (default)	OUT	OUT	IN	IN	IN	IN	
310 - 31F	OUT	OUT	IN	IN	IN	OUT	
350 - 35F	OUT	OUT	IN	OUT	IN	OUT	
360 - 36F	OUT	OUT	IN	OUT	OUT	IN	

12.2.3. Emergency Stop Input

The UNIDEX 500 has an optically isolated emergency stop input (refer to Figure 12-12). The user must provide an external power supply to drive the on-board opto-isolator. External voltages and resistances are enumerated in Table 12-24.

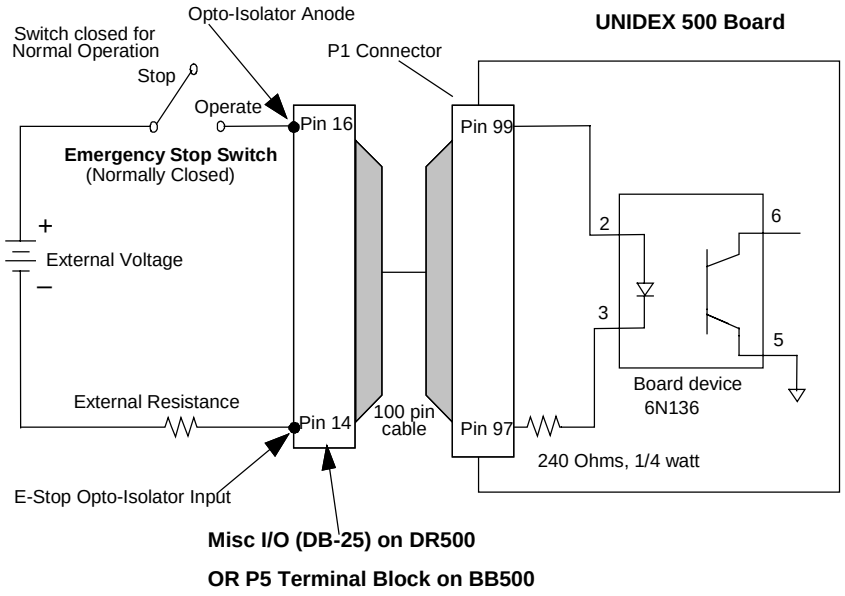


Figure 12-12. Electrical Characteristics of the UNIDEX 500 Emergency Stop Interface

The "Opto–isolator Anode" is Pin 99 of the UNIDEX 500 P1 connector and the "E-stop Opto–isolator Input" is pin 97 of P1. When a BB500 breakout unit is connected to the UNIDEX 500 board with a 100–pin cable, pin 16(OPTOA) of the BB500 P5 connector becomes the Opto–isolator Anode and pin 14 (ESTOP) of P5 becomes the E–stop Opto–isolator Input. Therefore connect the external Opto–isolator Anode to pin 16 and the E–stop Opto–isolator Input to pin 14 of the P5 connector on the BB500 board.

Once the emergency stop input is connected, the software must be set to look for an emergency stop condition. This is set by system parameter x55–Error Mask Fault. For any axis, select the Emergency Stop box for this parameter. This will force an emergency stop if the external circuitry is opened.

Table 12-24. External Voltages and Resistances for the Emergency Stop Input

External Voltage	External Resistance (in Ohms)
5 VDC	0 Ω
12 VDC	290 Ω, 1/4 watt
24 VDC	1KΩ, 1/2 watt

12.2.4. iSBX Connector Pinout (P4)

The pinouts for the iSBX connector are listed in Table 12-25.

Table 12-25. iSBX Connector Pinouts

Pin #	Description	Pin #	Description
43	MD8	44	MD9
41	MDA	42	MDB
39	MDC	40	MDD
37	MDE	38	MDF
35	GND	36	+5v
33	MD0	34	MDRQT
31	MD1	32	MDACK
29	MD2	30	OPT0
27	MD3	28	OPT1
25	MD4	26	TDMA
23	MD5	24	*reserved*
21	MD6	22	MCS0
19	MD7	20	MSC1
17	GND	18	+5v
15	IORD	16	MWAIT
13	IOWRT	14	MINTR0
11	MA0	12	MINTR1
9	MA1	10	*reserved*
7	MA2	8	MPST
5	RESET	6	MCLK
3	GND	4	+5v
1	+12v	2	-12v

12.2.5. DSP Bus Pin Description (P3)

Pinouts for the DSP bus are listed in Table 12-26 below.

Table 12-26. DSP Bus Pinouts

Pin #	Description	Pin #	Description
1	MA3	26	D4
2	GND	27	OPT1
3	MA4	28	D3
4	GND	29	OPT0
5	MCLK	30	D2
6	MRESET	31	MA11
7	MA5	32	D1
8	MA2	33	D16
9	MA6	34	D0
10	MA1	35	D17
11	MINTR1	36	D18
12	MA0	37	D19
13	MINTR0	38	D20
14	IOWR	39	D21
15	MA7	40	D22
16	IORD	41	D23
17	MA8	42	D14
18	GND	43	MWAIT
19	MCS1	44	D12
20	D7	45	D15
21	MCS0	46	D10
22	D6	47	D13
23	MA9	48	D8
24	D5	49	D11
25	MA10	50	D9

12.2.6. Encoder Interface between U500 and PC-PSO (P6)

The P6 connector provides a one–to–one interface to the PC-PSO (See Table 12-27). When the PC-PSO is used, the UNIDEX 500 P6 connector couples to the PC-PSO P6 connector. This links the encoder signals from the U500 to the PC-PSO board. The signals are sent in single ended format to the PC-PSO.

Table 12-27. Pinouts for the Encoder Interface to the PC-PSO

Signal Name	P6 location	Function
SIN1	1	Axis 1 Sine
COS1	2	Axis 1 Cosine
SIN2	5	Axis 2 Sine
COS2	6	Axis 2 Cosine
SIN3	9	Axis 3 Sine
COS3	10	Axis 3 Cosine
SIN4	13	Axis 4 Sine
COS4	14	Axis 4 Cosine
INTBUS	15	Wire–ORed Interrupt
GND	3,4,7,8,11,12, 16,18,20,22,24	Signal Common

12.2.7. Motorola DSP56002 OnCE Port (P8)

The P8 port connects to a Motorola DSP56002 OnCE debugger. It is used at Aerotech, Inc. for software debugging and is not intended for customer use. It is mentioned here for reference only. See Table 12-28.

Table 12-28. OnCE Port Pinouts (P8)

Signal Name	OnCE P8 pin
OS1/DSCK	5
OS0/DSI	1
DSO	3
DR_N	7
GND	2,4,6,10

12.3. UNIDEX 500 PCI Technical Details

12.3.1. Emergency Stop Input

The emergency stop (e-stop) input is enabled by setting the fault mask bit to 1. The opto coupled e-stop input must be forward biased to remove the e-stop condition. The e-stop circuit is designed for 20ma forward current and does not require an external resistor when used with 5V logic. The positive supply voltage should be connected to the OPTOA (opto-isolator anode) pin. The diode is forward biased by completing the circuit through the ESTOP pin back to the power supply. Note that the OPTOA pin is shared with the UINT opto isolator.

Table 12-29. U500 PCI Emergency Stop Electrical Specifications

Voltage	External Resistor
5V	0 (none)
12V	330 (1/4W)
24V	1K (1W)

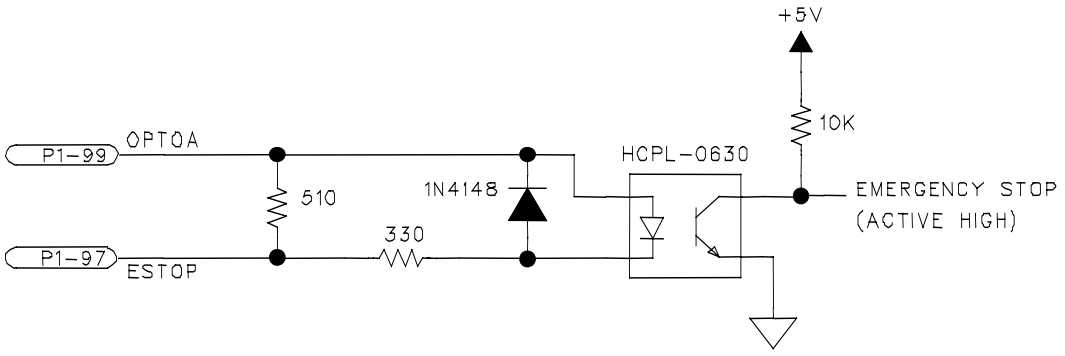


Figure 12-13. U500 PCI Emergency Stop Input

12.3.2. U500 PCI Analog Inputs

The U500 PCI has 8 analog inputs that are sampled every 2 msec.

Input voltage range = ±10V

A/D Converter Resolution = 12 bit (.005 mv/bit)

12.3.3. CLK/DIR Stepper Mode

The U500 BASE, PLUS, and ULTRA cards can generate up to four CLK/DIR outputs.

On the BASE, PLUS, and ULTRA cards, these can be used to make any or all of the first four axes work with a CLK/DIR amplifier. Set the JP6 and JP7 jumpers for the axis and set axis parameter 42 to 4. In this case, the CLK signal is connected to the ICMDB and the DIR signal is connected to ICMDA.

The U500PCI ULTRA may be configured for four servos and four CLK/DIR stepper axes. Servo axes must be between 1-4. Axes 5-8 must be CLK/DIR if used. Axis 1-4 can be configured as CLK/DIR, but the corresponding 5-8 axes use is forfeited (see Figure 12-14.)

Table 12-30. Allowable Combinations for U500 BASE/PLUS

Servo	CLK/DIR Stepper
4 axes (X, Y, Z, and U)	none
3 axes (any X, Y, Z, and U)	1 axis (remaining axis X, Y, Z, or U)
2 axes (any X, Y, Z, and U)	2 axes (remaining axis X, Y, Z, or U)
1 axis (any X, Y, Z, and U)	3 axes (remaining axis X, Y, Z, or U)
none	4 axes (X, Y, Z, and U)

Table 12-31. Allowable Combinations for U500 ULTRA

Axis 1-4	Axis 5-8 (CLK/DIR)
4 servo, 0 CLK/DIR	4 CLK/DIR
3 servo, 1 CLK/DIR	3 CLK/DIR
2 servo, 2 CLK/DIR	2 CLK/DIR
1 servo, 3 CLK/DIR	1 CLK/DIR
0 servo, 4 CLK/DIR	0 CLK/DIR

If axis 1 is configured as CLK/DIR, axis 5 is not available.
If axis 2 is configured as CLK/DIR, axis 6 is not available.
If axis 3 is configured as CLK/DIR, axis 7 is not available.
If axis 4 is configured as CLK/DIR, axis 8 is not available.

- Axis 1-4 may be used with an AM8007C amplifier to control a stepper motor.
- This amplifier is for use with the DR300/DR500 amplifier racks and does not use CLK/DIR signals.
- Axes 5-8 do not support contoured motion. All contoured motion must be located on axes 1-4.
- CLK/DIR outputs are TTL level signals capable or sourcing / sinking 24ma.

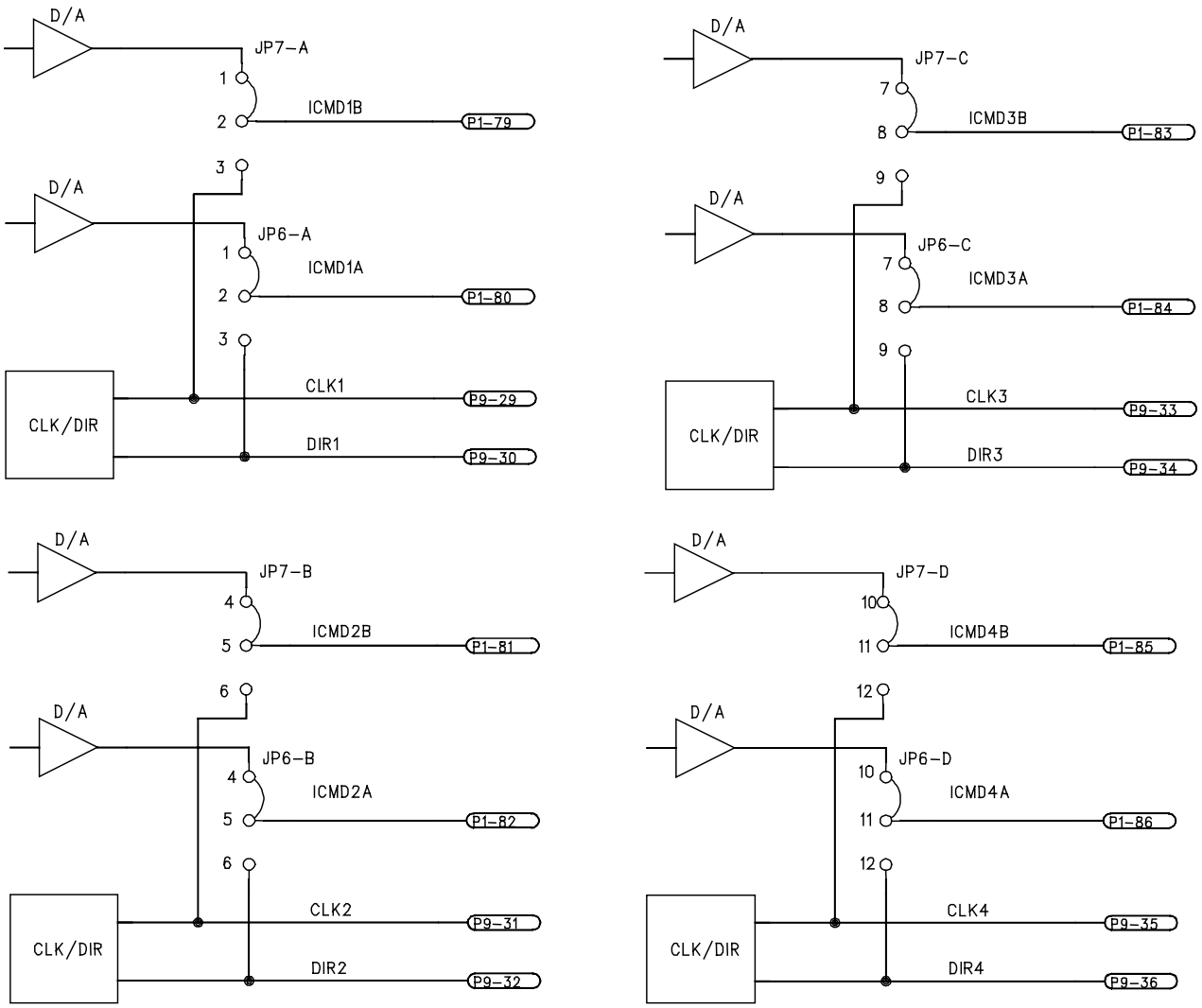


Figure 12-14. CLK/DIR Schematic

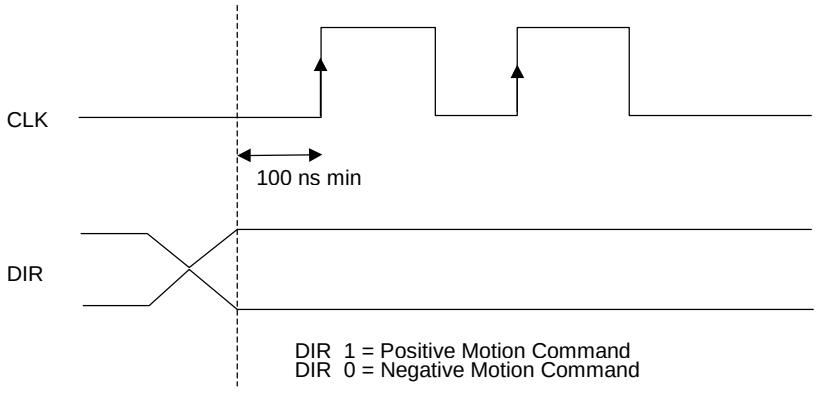


Figure 12-15. U500 PCI CLK/DIR Stepper Section

Table 12-32. U500 PCI CLK/DIR Stepper Mode Jumpers

	AXIS 1	AXIS 2	AXIS 3	AXIS 4
Current Command (JP6, JP7)	1-2, 1-2	4-5, 4-5	7-8, 7-8	10-11, 10-11
CLK/DIR Output (JP6, JP7)	2-3, 2-3	5-6, 5-6	8-9, 8-9	11-12, 11-12

12.3.4. Additional D/A Outputs

The U500PCI has two additional DAC outputs located on the secondary axis interface connector (P9). They are +/-10V analog outputs. See Chapter 7: Programming Commands for additional information.

12.3.5. U500 PCI D/A Analog Outputs

The 8 D/A converters used as current command outputs (DAC1-8) can be isolated from digital GND. This can be used to eliminate ground loops that may occur between the controller and the amplifiers. To do this, the user must supply an external +/- 12V power source. Jumpers JP1, JP3, and JP4 control whether the U500 gets the +/- 12V power from the PC bus or from an external source.

Table 12-33. Non-Isolated System Jumpers

Jumper	Setting	Function
JP1	1-2*	Analog gnd connected to PC's digital ground
JP3	1-2*	+12V from PC bus
JP4	1-2*	-12V from PC bus

* default

Table 12-34. Isolated System Jumpers (external power supply required)

Jumper	Setting	Function
JP1	2-3	Analog gnd connected to P1-54 and P1-56
JP3	2-3	+12V from P1-53
JP4	2-3	-12V from P1-55

Note: DAC 9/10 cannot be opto-isolated.

Table 12-35. D/A Output Specifications

Voltage	Output Current
±10 V	±10 ma maximum

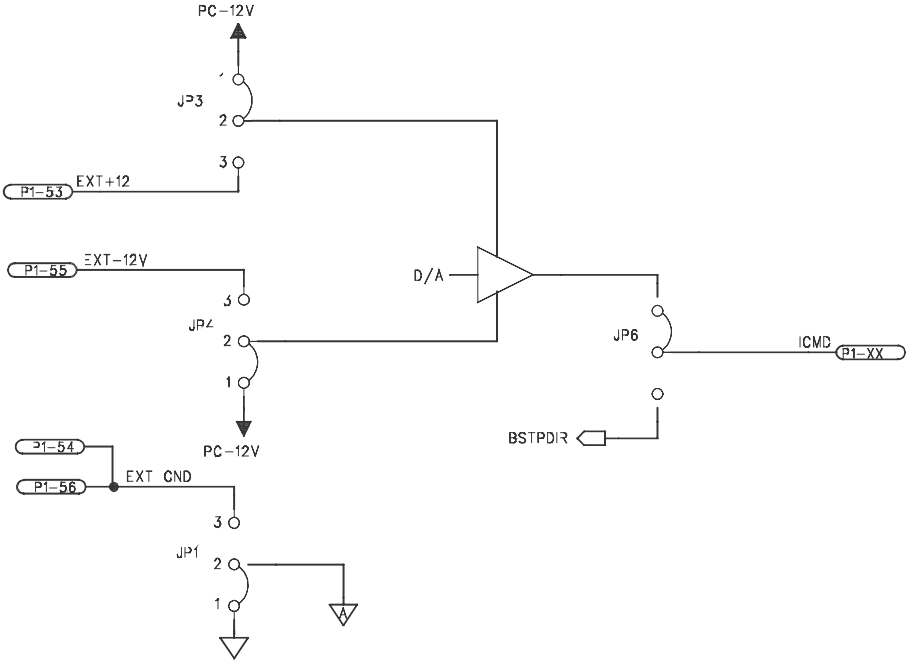


Figure 12-16. U500 PCI D/A Output Diagram

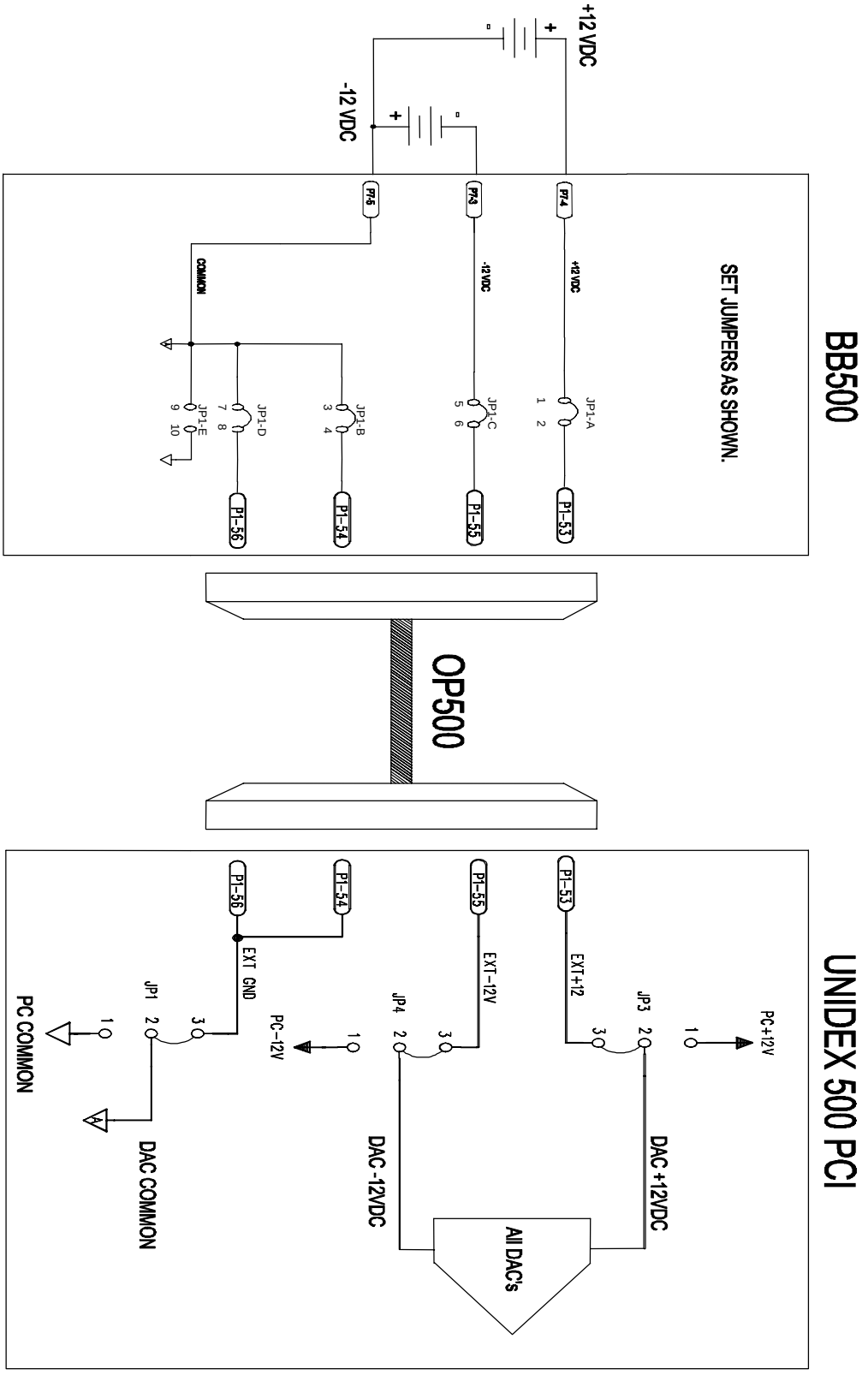


Figure 12-17. U500 PCI Connection to BB500 (Isolated)

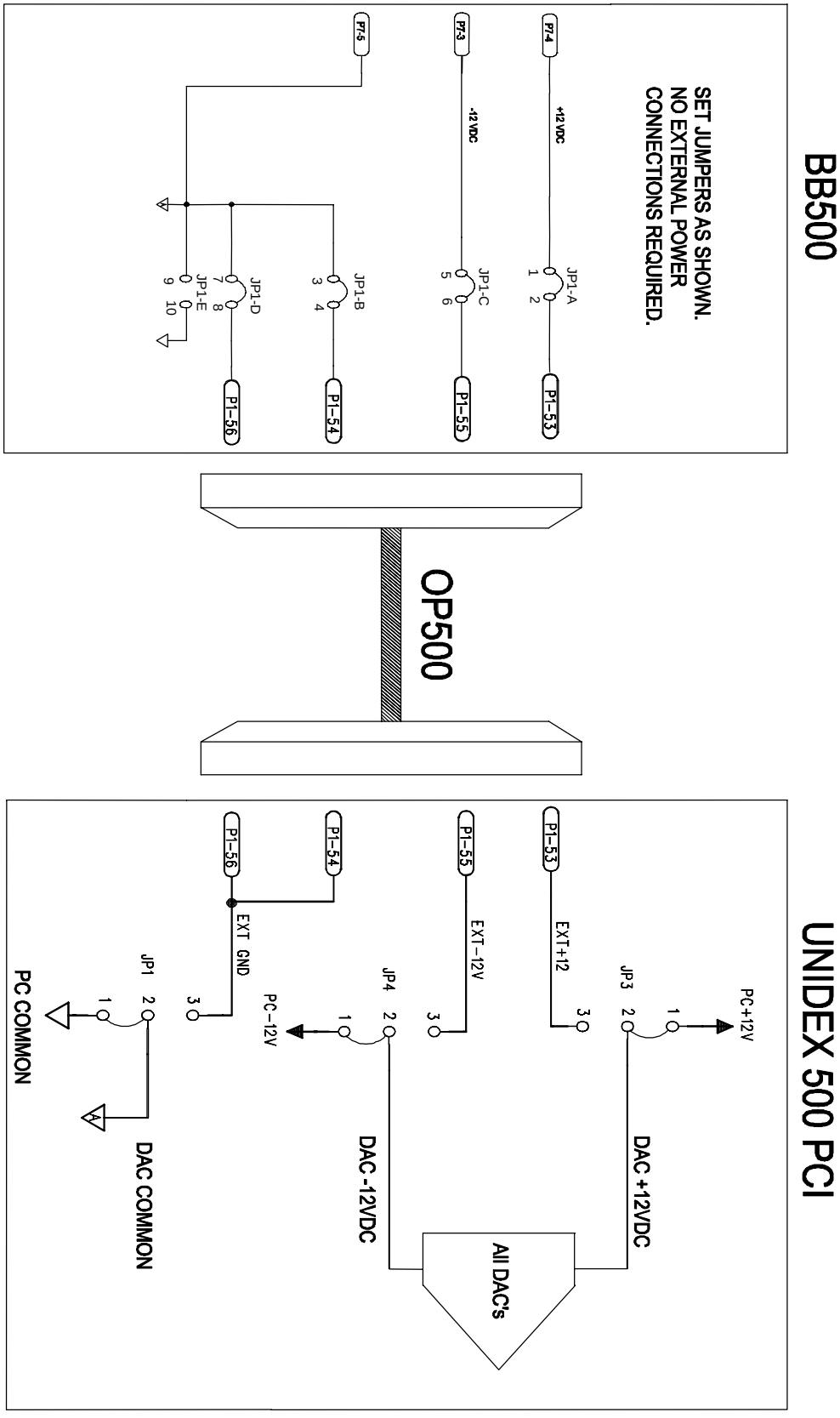


Figure 12-18. U500 PCI Connection to BB500 (Non-Isolated)

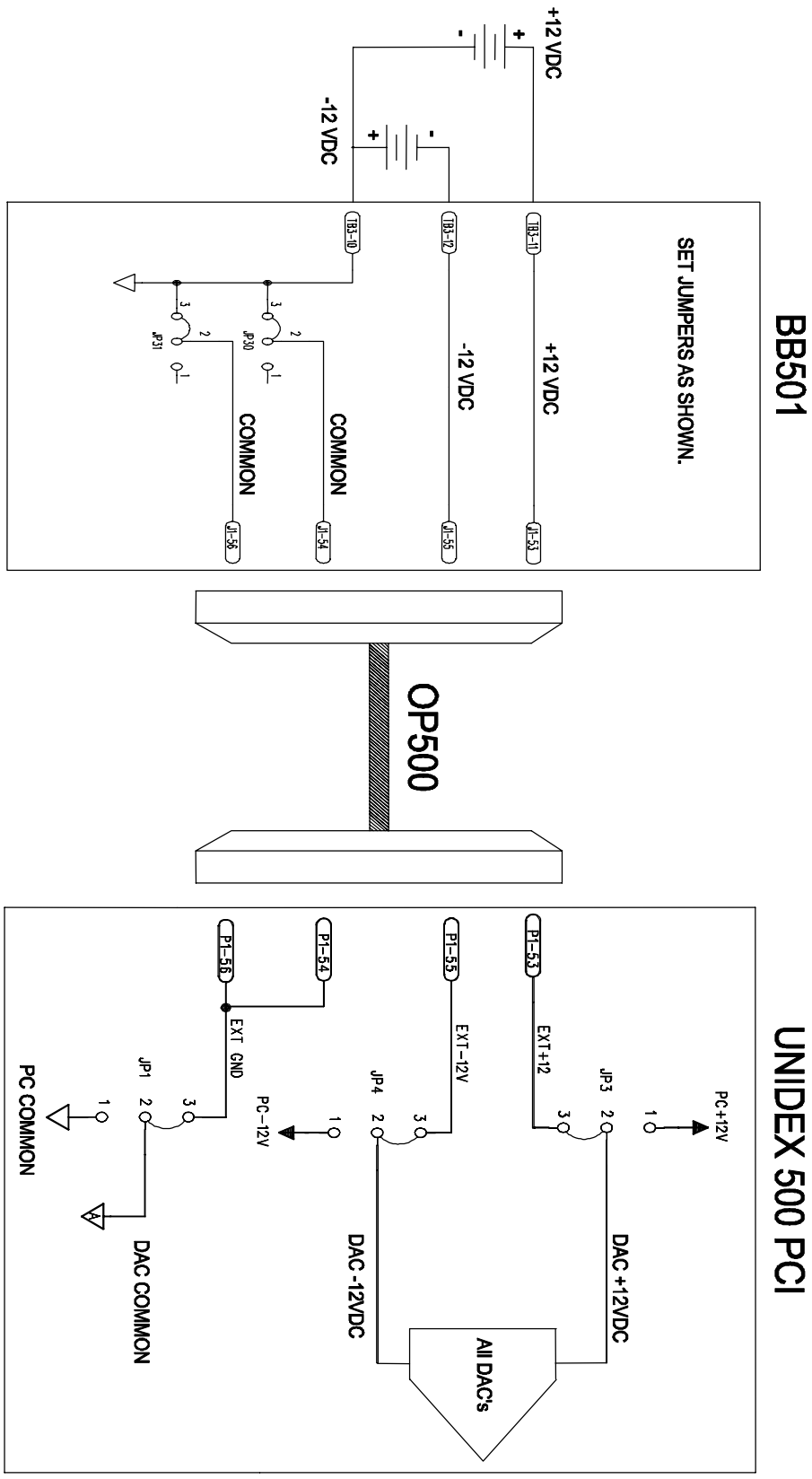


Figure 12-19. U500 PCI Connection to BB501 (Isolated)

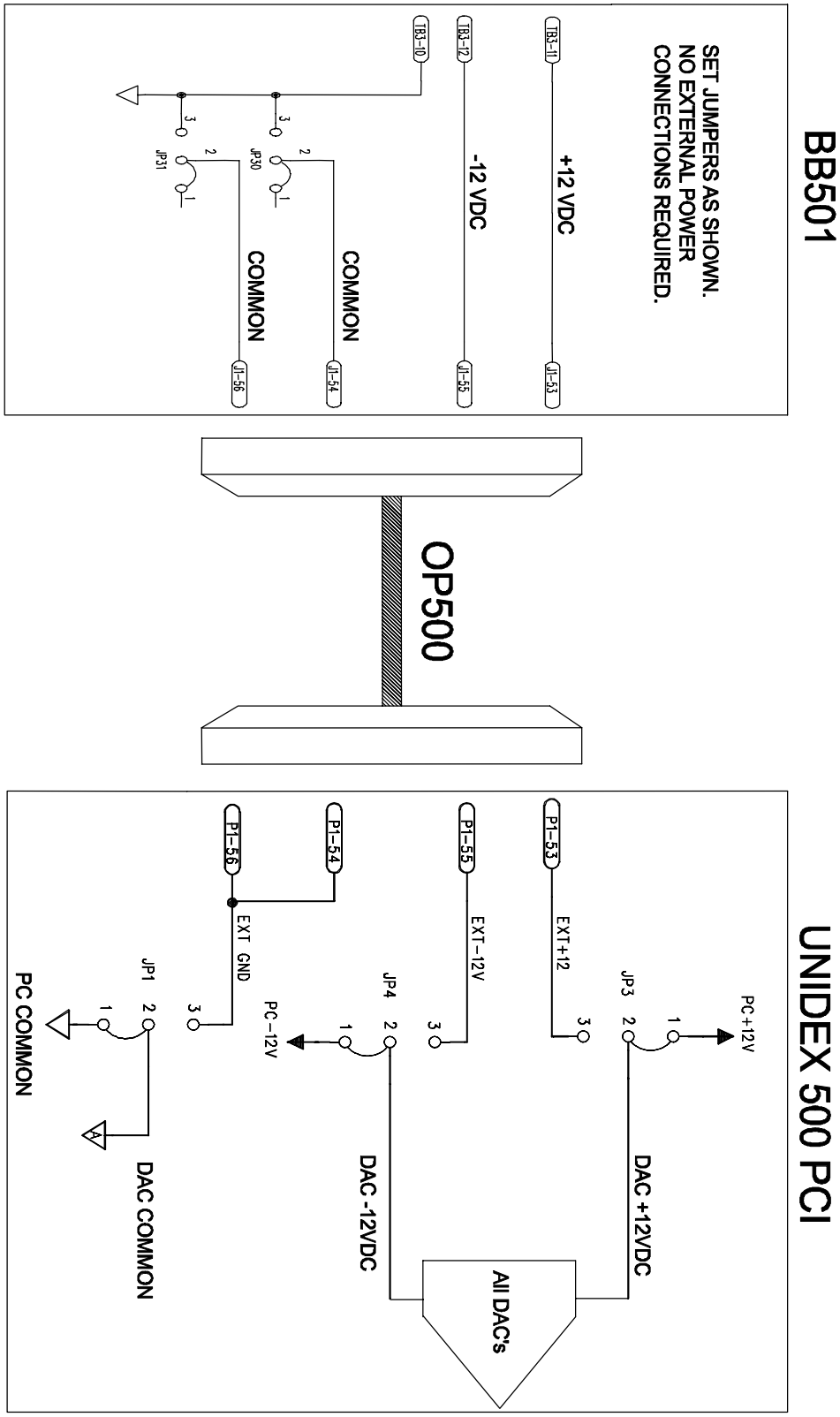


Figure 12-20. U500 PCI Connection to BB501 (Non-Isolated)

12.3.6. U500 PCI Test Points

Table 12-36. U500 PCI Test Points

Test Point	Description	Test Point	Description
TP1	datalatch	TP29	fpga1_n
TP2	ebus_n	TP30	fpga2_n
TP3	pcs_n	TP31	dout1
TP4	digital gnd	TP32	init1_n
TP5	pbusen_n	TP33	pulse output – opto reference
TP6	addlatch	TP34	pulse output – O.C. output
TP7	rd_n	TP35	pulse output – LED anode
TP8	wr_n	TP36	pulse output – opto power
TP9	aa3		
TP10	aa2		
TP11	sc12		
TP12	sc11		
TP13	sc10		
TP14	digital gnd		
TP15	std1		
TP16	aa1		
TP17	srd1		
TP18	aa0		
TP19	prd_n		
TP20	pwr_n		
TP21	brake		
TP22	pulse output		
TP23	marker1		
TP24	marker2		
TP25	marker3		
TP26	marker4		
TP27	dout2		
TP28	init2_n		

12.3.7. U500 Specifications

Table 12-37. U500 Specifications: Current Consumption

Voltage	Milliamps
+5V	750 ma.
+12V	180 ma.
-12V	50 ma.

12.3.8. U500 PCI Jumpers

Table 12-38. U500 PCI Jumpers

Jumper	IN/OUT	Default	Function
JP10	1-2	*	Watch Dog from software
	2-3		Watch dog defeated – for programming flash , PLD, and diagnostics
JP12	1-2	*	Laser fire output active low ; HI-Z during reset and laser off (opto-coupled and TTL output)
	2-3		Laser fire output active high (actually High Z); Low Z to GND during reset and laser off (opto-coupled and TTL output)
JP6	1-2	*	ICMD1A to OP500 (for AC servo, commutated stepper)
	2-3		DIR1 to OP500 (for CLK/DIR stepper interface)
JP7	1-2	*	ICMD1B to OP500 (for AC servo, commutated stepper or DC brush)
	2-3		CLK1 to OP500 (for CLK/DIR stepper interface)
JP6	4-5	*	ICMD2A to OP500 (for AC servo, commutated stepper)
	5-6		DIR2 to OP500 (for CLK/DIR stepper interface)
JP7	4-5	*	ICMD2B to OP500 (for AC servo, commutated stepper or DC brush)
	5-6		CLK2 to OP500 (for CLK/DIR stepper interface)
JP6	7-8	*	ICMD3A to OP500 (for AC servo, commutated stepper)
	8-9		DIR3 to OP500 (for CLK/DIR stepper interface)
JP7	7-8	*	ICMD3B to OP500 (for AC servo, commutated stepper or DC brush)
	8-9		CLK3 to OP500 (for CLK/DIR stepper interface)
JP6	10-11	*	ICMD4A to OP500 (for AC servo, commutated stepper)
	11-12		DIR4 to OP500 (for CLK/DIR stepper interface)
JP7	10-11	*	ICMD4B to OP500 (for AC servo, commutated stepper or DC brush)
	11-12		CLK4 to OP500 (for CLK/DIR stepper interface)
JP8	1-2	*	Boot from FLASH memory
	2-3		Boot from PCI bus (for FLASH update, diagnostic, etc ONLY)
JP9*	1-2	in	User interrupt is opto-coupled by an HCPL-0630 isolator.
	3-4	out	A forward biased opto-coupler diode generates the interrupt edge. P6-15 (BUINT_N) is wire-ored to the output of the opto-coupler.
	5-6	in	Output of the opto coupler, active low.

Table 12-38. U500 PCI Jumpers (Cont'd)

Jumper	IN/OUT	Default	Function
JP9	1-2	out	User interrupt is active low input TTL level and referenced to signal common.
	3-4	in	
	5-6	out	
JP2	1-2	*	3.3V derived from 5V PCI bus
	2-3		3.3V from PCI bus
JP11	1-2	*	auto-detect firmware jumpers
	3-4	*	
	5-6	*	
	7-8	*	
JP13*	1-2	in	Master U500PCI card on SSI
	3-4	out	
	5-6	out	
	7-8	out	
JP13	1-2	out	U500PCI SLAVES on SSI
	3-4	out	
	5-6	out	
	7-8	out	
JP13	1-2	out	Last U500PCI SLAVE on SSI
	3-4	in	
	5-6	in	
	7-8	in	
			Standard non-opto isolated analog outputs (U500PCI BASE and PLUS)
JP1	1-2	*	DAC common connected to PC common
JP3	1-2	*	DAC +12V from PC
JP4	1-2	*	DAC –12V from PC
			Opto coupled analog outputs with external +/- 12V supply (U500PCI ULTRA only)
JP1	2-3		DAC common to external user supplied power supply (P1-54 P1-56)
JP3	2-3		DAC +12V from external user supplied power supply (P1-53)
JP4	2-3		DAC -12V from external user supplied power supply (P1-55)
JP5	out	*	reserved

12.3.9. Connector Pinouts

12.3.9.1. P1 – Main Connector

Table 12-39. P1 – Main Connector Pinouts

(The parenthetical information shows the pin definitions for the U500 ISA card.)

Pin	Definition	Pin	Definition
1	PC GND *(interlock send)*	2	GND
3	PC +5	4	PC +5
5	HB1 *(gnd)*	6	HA1 *(gnd)*
7	SIN1+	8	SIN1-
9	COS1+	10	COS1-
11	MRK1+	12	MRK1-
13	HB2 *(gnd)*	14	HA2 *(gnd)*
15	SIN2+	16	SIN2-
17	COS2+	18	COS2-
19	MRK2+	20	MRK2-
21	HC2 *(gnd)*	22	HB3 *(gnd)*
23	SIN3+	24	SIN3-
25	COS3+	26	COS3-
27	MRK3+	28	MRK3-
29	HA3 *(gnd)*	30	HC3 *(gnd)*
31	SIN4+	32	SIN4-
33	COS4+	34	COS4-
35	MRK4+	36	MRK4-
37	HB4 *(gnd)*	38	HA4 *(gnd)*
39	CW1	40	CCW1
41	CW2	42	CCW2
43	CW3	44	CCW3
45	CW4	46	CCW4
47	HOME1	48	HOME2
49	HOME3	50	HOME4
51	GND	52	HC1 *(gnd)*
53	DAC +12 IN *(pc +12 out)*	54	DAC GND (*pc +12 out*)
55	DAC –12 IN *(pc -12 out)*	56	DAC GND *(pc -12 out)*
57	MODE 1	58	MODE 2
59	IN 0	60	IN 1
61	IN 2	62	IN 3
63	OUT 0	64	OUT 1
65	OUT 2	66	OUT3
67	MODE 3	68	MODE 4
69	AEN 1	70	AEN 2
71	AEN 3	72	AEN 4
73	AFLT 1	74	AFLT 2
75	AFLT 3	76	AFLT 4
77	GND	78	GND
79	ICMD1B	80	ICMD1A
81	ICMD2B	82	ICMD2A
83	ICMD3B	84	ICMD3A
85	ICMD4B	86	ICMD4A
87	GND	88	HC4 *(gnd)*
89	JSW1 ($AD3)	90	JSW2 ($AD2)
91	JSA	92	JSB
93	JSC (interlock)	94	BRAKE
95	AIN0 ($AD1)	96	AIN1 ($AD0)
97	ESTOP	98	UINT
99	OPTOA	100	ILOCKR

12.3.9.2. P3 – Expansion Connector

Table 12-40. P3 – Expansion Connector Pinouts

The parenthetical information shows the pin definitions for the U500 ISA card.

Pin	Definition	Pin	Definition
1	MA03	2	GND
3	MA04	4	GND
5	MCLK	6	MRESET
7	MA05	8	MA02
9	MA06	10	MA01
11	**GND *(MINT1)***	12	MA00
13	**GND *(MINT0)***	14	IOWR_N
15	MA07	16	**IORD_N**
17	MA08	18	GND
19	**+5 *(mcs1_n)***	20	MD7
21	**+5 *(mcs0_n)***	22	MD6
23	MA09	24	MD5
25	MA10	26	MD4
27	**-12 *(opt1)***	28	MD3
29	**+12 *(opt0)***	30	MD2
31	MA11	32	MD1
33	MD16	34	MD0
35	MD17	36	MD18
37	MD19	38	MD20
39	MD21	40	MD22
41	MD23	42	MD14
43	MWAIT_N	44	MD12
45	MD15	46	MD10
47	MD13	48	MD8
49	MD11	50	MD9

12.3.9.3. P4 – 8X3 IO Conncetor

Table 12-41. P4 – 8X3 IO Connector Pinouts

Pin	Definition	Pin	Definition
1	IOC7	2	GND (key)
3	IOC6	4	GND
5	IOC5	6	GND
7	IOC4	8	GND
9	IOC3	10	GND
11	IOC2	12	GND
13	IOC1	14	GND
15	IOC0	16	GND
17	IOB7	18	GND
19	IOB6	20	GND
21	IOB5	22	GND
23	IOB4	24	GND
25	IOB3	26	GND
27	IOB2	28	GND
29	IOB1	30	GND
31	IOB0	32	GND
33	IOA7	34	GND
35	IOA6	36	GND
37	IOA5	38	GND
39	IOA4	40	GND
41	IOA3	42	GND
43	IOA2	44	GND
45	IOA1	46	GND
47	IOA0	48	GND
49	+5 (fused)	50	GND

12.3.9.4. P5 - 16IN 8 OUT (OPTIONAL HALL INPUTS)

Table 12-42. P5 – 16 In 8 Out (Optional Hall Inputs)

See Technical Details Common between U500 PCI and ISA Cards on page 12-1 for more details.

Pin	Definition	Pin	Definition
1	IN15/HC4	2	GND (key)
3	IN14/HA4	4	GND
5	IN13/HB4	6	GND
7	IN12/HC3	8	GND
9	IN11/HA3	10	GND
11	IN10/HB3	12	GND
13	IN9/HC2	14	GND
15	IN8/HA2	16	GND
17	IN7/HB2	18	GND
19	IN6/HC1	20	GND
21	IN5/HA1	22	GND
23	IN4/HB1	24	GND
25	IN3	26	GND
27	IN2	28	GND
29	IN1	30	GND
31	IN0	32	GND
33	OUT7	34	GND
35	OUT6	36	GND
37	OUT5	38	GND
39	OUT4	40	GND
41	OUT3	42	GND
43	OUT2	44	GND
45	OUT1	46	GND
47	OUT0	48	GND
49	+5 (fused)	50	GND

12.3.9.5. U500 PCI – P6 – Internal Encoder/Miscellaneous Interface

Table 12-43. U500 PCI – P6 – Internal Encoder/Misc. Interface Pinouts

Pin	Signal Name	Description
1	SIN 1	single ended 3.3V output from line receiver
2	COS 1	single ended 3.3V output from line receiver
3,4	GND	u500/PC signal common
5	SIN 2	single ended 3.3V output from line receiver
6	COS 2	single ended 3.3V output from line receiver
7,8	GND	u500/PC signal common
9	SIN 3	single ended 3.3V output from line receiver
10	COS 3	single ended 3.3V output from line receiver
11,12	GND	u500/PC signal common
13	SIN 4	single ended 3.3V output from line receiver
14	COS 4	single ended 3.3V output from line receiver
15	BUINT_N (OC) (position capture input)	negative edge triggered , 10KOhm pull up to +5V
16	AUX_INPUT	reserved
17	OUT0	active high output +5V (resets to 0V) (74ACT273)
18	AUXIN0	10k pull up to +5V
19	OUT1	active high output +5V (resets to 0V) (74ACT273)
20	AUXIN1	10k pull up to +5V
21	FPGA2_IN0	10KOhm pullup to 3.3V (input to fpga)
22	AUXIN2	10k pull up to +5V
23	FPGA2_IN1	10KOhm pullup to 3.3V (input to fpga)
24	AUXIN3	10k pull up to +5V
25	FPGA2_OUT0	5V output – active high – 74ACT14
26	FPGA2_OUT1	5V output – active high – 74ACT14

12.3.9.6. P8 – ½ Secondary Axis Interface

Table 12-44. P8 – ½ Secondary Axis Interface Pinouts

Pin	Definition	Pin	Definition
1	PC GND *(interlock send)*	2	GND
3	PC +5 (FUSED) – (KEY)	4	PC +5 (FUSED)
5	HB5 *(gnd)*	6	HA5 *(gnd)*
7	SIN5+	8	SIN5-
9	COS5+	10	COS5-
11	MRK5+	12	MRK5-
13	HB6 *(gnd)*	14	HA6 *(gnd)*
15	SIN6+	16	SIN6-
17	COS6+	18	COS6-
19	MRK6+	20	MRK6-
21	HC6 *(gnd)*	22	HB7 *(gnd)*
23	SIN7+	24	SIN7-
25	COS7+	26	COS7-
27	MRK7+	28	MRK7-
29	HA7 *(gnd)*	30	HC7 *(gnd)*
31	SIN8+	32	SIN8-
33	COS8+	34	COS8-
35	MRK8+	36	MRK8-
37	HB8 *(gnd)*	38	HA8 *(gnd)*
39	CW5	40	CCW5
41	CW6	42	CCW6
43	CW7	44	CCW7
45	CW8	46	CCW8
47	HOME5	48	HOME6
49	HOME7	50	HOME8

12.3.9.7. P9 – ½ Secondary Axis Interface

Table 12-45. P9 – ½ Secondary Axis Interface Pinouts

Pin	Definition	Pin	Definition
1	GND	2	HC5 *(gnd)*
3	(reserved) – (KEY)	4	(reserved)
5	(reserved)	6	(reserved)
7	TDX+	8	TDX-
9	IN 0B	10	IN 1B
11	IN 2B	12	IN 3B
13	OUT 0B	14	OUT 1B
15	OUT 2B	16	OUT3B
17	RXD+	18	RXD-
19	AEN 5	20	AEN 6
21	AEN 7	22	AEN 8
23	AFLT 5	24	AFLT 6
25	AFLT 7	26	AFLT 8
27	GND	28	GND
29	CLK 1	30	DIR 1
31	CLK 2	32	DIR 2
33	CLK 3	34	DIR 3
35	CLK 4	36	DIR 4
37	GND	38	HC8 (*gnd)*
39	A/D INPUT 8 ($AD7)	40	A/D INPUT 7 ($AD6)
41	DAC9	42	DAC10
43	O.C. fire output (must remove opto) (gnd ref)	44	TTL (5V) fire output (gnd ref)
45	A/D INPUT 6 ($AD5)	46	A/D INPUT 5 ($AD4)
47	LASER COMMON	48	LASER FIRE OUT
49	LASER +5	50	ILOCKR

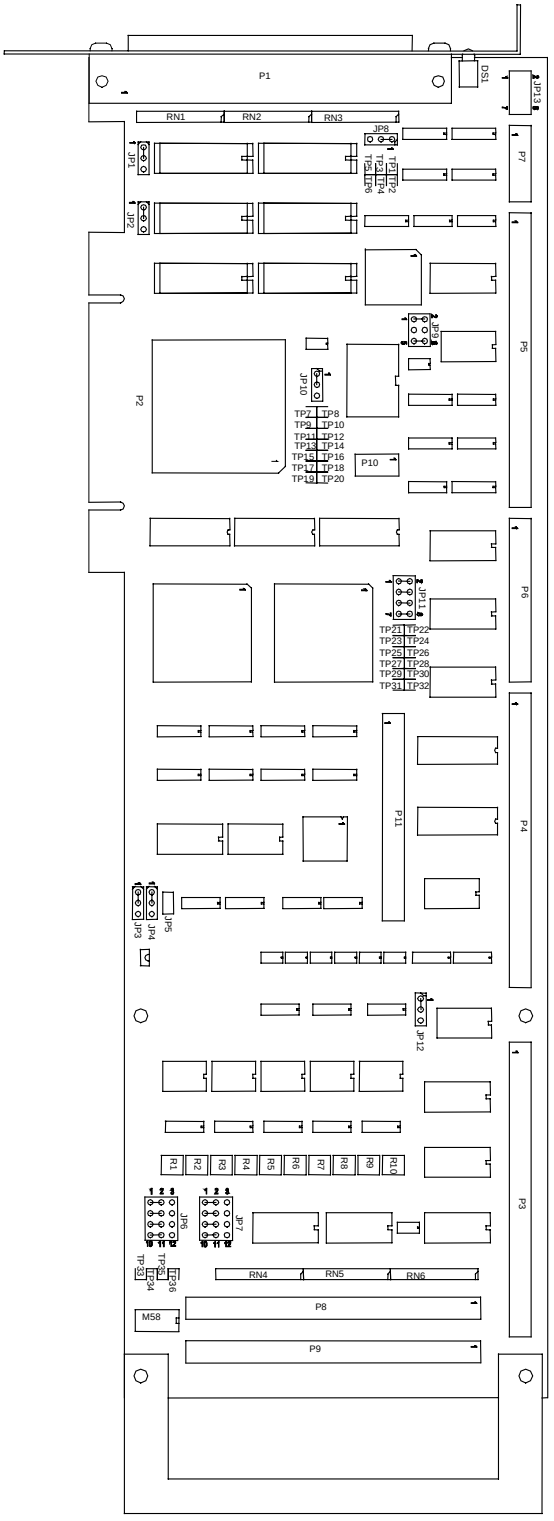


Figure 12-21. U500 PCI Board

∇ ∇ ∇

CHAPTER 13: TROUBLESHOOTING

In This Section:

- Installation, Board Startup, and Communication Problems..... 13-2
- Stepper Motors and Related Problems 13-3
- Servo Related Problems 13-4
- Problems Involving Fault Conditions 13-5
- Homing Related Problems..... 13-7
- General UNIDEX 500 Questions 13-8

If you have technical support questions, please have the following information available before calling:

1. The current version of the software. To obtain the version number from the software program, select the About command under the Help Menu.
2. Your customer order number.
3. We may also need to know the type of PC you are using (brand name, CPU, available memory), and the operating system.
4. If you are developing your own application, we will need to know what compiler and version number you are using (e.g., Borland C v3.1, Microsoft Visual C, etc.).
5. If at all possible, try to be in front of the system where the problems are occurring.

Also, please refer to the "Frequently Asked Questions" for the UNIDEX 500 Motion Controller at our website: http://www.aerotechinc.com/faqhome.html

13.1. Installation, Board Startup, and Communication Problems

Some common problems that relate to installation, startup and communications are listed and diagnosed in Table 13-1.

Table 13-1. Troubleshooting for Common Installation, Startup, and Communication Problems

Problem	Possible Causes / Solutions	See Section
Initialization failure, communication failure, or "Bus Timeout !" error occurs...	The UNIDEX 500 board is not installed. The UNIDEX 500 board is not seated properly. The software address does not match the hardware address. Another device in the PC is set to the same base address as the U500 board. Select a unique base address.	2.1-2.3, 3.4, 12.2
Checksum failure ...	The UNIDEX 500 board is faulty. Another device in the PC is set to the same base address as the U500 board. Select a unique base address.	2.3, 3.3, 3.4, 12.2, Appendix B
The LED never lights up after power up ...	The UNIDEX 500 board is not seated properly. Another device in the PC is set to the same base address as the U500 board. Select a unique base address. The power supply has been overloaded and has shut down. Install a larger power supply to correct the problem.	2.3, 3.3, 3.4, 12.2, PC Manual
The LED lights after power up and remains on even after attempts to initialize ...	The software base address does not match the base address jumper settings on the UNIDEX 500 board. Another device in the PC is set to the same base address as the U500 board. Select a unique base address.	2.3, 3.3, 3.4, 12.2
The PC power supply goes dead when the UNIDEX 500 is installed ...	The power supply has been overloaded and has shut down. Install a larger power supply to correct the problem. External wiring problems exist. Remove the main interface cable and recheck. The UNIDEX 500 board is faulty.	2.3, 3.3, 3.4, 12.2, PC Manual, Appendix B
A PC bus timeout error occurs and the U500 LED is blinking...	If you have a REV _ U500 board, make sure you're using the U500.JWP firmware file. If you have a REV C U500 board, make sure you're using the U500C.JWP firmware file.	3.3, 3.4, Appendix B

13.2. Stepper Motors and Related Problems

Some common problems that relate to the use of stepper motors are listed and diagnosed in Table 13-2.

Table 13-2. Troubleshooting for Stepper Motors (and Related) Problems

Problem	Possible Causes / Solutions	See Also ...
The stepper motor overheats...	The stepper high current parameter is set too high. The stepper low current parameter is set too high.	5.8
The stepper motor drops out...	The ramp time parameter is set too low. The acceleration/deceleration parameter is set too low. Load on motor is too great.	5.5, 5.7, 5.11
The motor rotates in the wrong direction ...	The motor phasing is incorrect.	5.8
The motor has no torque ...	The appropriate axis is not enabled. The motor wiring is faulty. The amplifier fuse is blown.	Motor or Amp Documents
The amplifier LED is on, but the motor will not move (even though it did previously) ...	Check for a blown amplifier fuse.	Motor or Amp Documents

13.3. Servo Related Problems

Some common problems that relate to the use of servo motors are listed and diagnosed in Table 13-3.

Table 13-3. Troubleshooting for Servo Related Problems

Problem	Possible Causes / Solutions	See Also ...
The motor has no torque ...	The appropriate axis is not enabled. The motor wiring is faulty. The amplifier fuse is blown. The amplifier is faulty.	Appendix B, Motor or Amp Documents
The motor buzzes or makes an unusual noise ...	The PID loop gains are not adjusted properly.	5.9, Chapter 6, 4.4
The motor runs away when it is enabled ...	The feedback device is not connected. The wrong feedback channel has been specified. Verify feedback. The wrong feedback setup code has been specified. Verify feedback.	5.8, Appendix B, Motor or Amp Documents
A position or integral trap error occurs when the motor is enabled ...	The feedback device is not connected. The wrong feedback channel has been specified. Verify feedback. The motor has no torque. (See above)	5.8, Appendix B, Motor or Amp Documents
A position or integral trap error occurs when motion is commanded ...	The feedback device is not connected. The wrong feedback channel has been specified. Verify feedback. The wrong feedback setup code has been specified. Verify feedback. The motor has no torque. (See above)	5.8, Appendix B, Motor or Amp Documents
The amplifier does not enable ...	An amplifier fault has occurred. This could be due to an improperly wired or shorted motor. The amplifier is faulty.	Motor or Amp Documents, Appendix B

13.4. Problems Involving Fault Conditions

Some common problems relating to fault conditions are listed and diagnosed in Table 13-4.

Table 13-4. Troubleshooting for Problems Involving Fault Conditions

Problem	Possible Causes / Solutions	See Also ...
A position or integral trap error occurs when the axis is enabled ...	The feedback device is not connected. The wrong feedback channel has been specified. Verify the feedback. The wrong feedback setup code has been specified. Verify the feedback. Feedback device is phased wrong. Make sure feedback counts positive when motor is turned clockwise (CW).	5.8, Motor or Amp Documents, Appendix B
A position or integral trap error occurs when motion is commanded ...	The feedback device is not connected. The wrong feedback channel has been specified. Verify the feedback. The wrong feedback setup code has been specified. Verify the feedback.	5.8, Motor or Amp Documents, Appendix B
Driver Interlock Open message is displayed ...	The OP500 cable is not inserted properly (e.g., there is no connection between pins 1 and 50). Check the cables and then acknowledge the fault (F10 Ack).	4.6
An emergency stop condition occurs ...	The emergency stop input is in the active state. Set the fault mask parameter if an emergency stop is not desired.	5.11
A clockwise (CW) or counter-clockwise (CCW) limit condition always exists ...	Limits are not connected to the UNIDEX 500. Run diagnostics. The active polarity parameters of the limits are set wrong. Check fuse (F1) on DR500 drive chassis.	4.4.1, 5.6, Motor or Amp Documents
An axis is in a CW or CCW limit condition ...	The commanded motion extended past the limit. Acknowledge the fault to move out of the limit range. The system has been powered up in the limit condition. Acknowledge the fault to move out of the limit range. The active limit polarity setup parameter is set wrong. Software limits are improperly set.	4.6, 5.6

Table 13-4. Troubleshooting for Problems Involving Fault Conditions (Cont.)

Problem	Possible Causes / Solutions	See Also ...
An over current trap (RMS over current fault) error has occurred ...	If the motor makes unusual noises or oscillates, the gain parameters may need to be adjusted. The RMS current trap parameter is set too low. The RMS current trap time is set too short. The amplifier gain parameter is set too low. The mechanical system is damaged or jammed. The motor/amplifier may be undersized for the load.	5.9, 5.11, Chapter 6, Appendix B, Motor or Amp Documents
A feedback trap has occurred ...	The incorrect feedback channel has been specified. The incorrect feedback setup code has been specified. The encoder is not connected. Run diagnostics and check tracking display. Single ended encoders are connected. Set the fault mask to ignore encoder faults. A sinusoidal encoder is connected. The UNIDEX 500 accepts square wave encoders only. One or more encoder connections are broken. The encoder is faulty. The resolver is not connected. A resolver-to-digital tracking loop error has occurred. One or more resolver connections are broken. Incorrect setup code for the resolver has been used. The resolver reference has not been adjusted properly.	4.4.1, 5.3, 5.8, 5.11, 12.3, Appendix B
A feedrate trap has occurred …	The commanded feedrate may have exceeded the top feedrate parameter (x17).	5.7, 5.11
Function or sub not defined …	Incorrect version of U500 DLL's. Ensure old copies of WIN50032.DLL and/or QLB50032.DLL do not exist in the path.	

13.5. Homing Related Problems

Some common problems relating to the homing process are listed and diagnosed Table 13-5.

Table 13-5. Troubleshooting for Homing Related Problems

Problem	Possible Causes / Solutions	See Also ...
The axis takes a long time to home ...	The home feedrate parameter (x05) is set too low. The power on home feedrate parameter (x04) is set too low. The maximum acceleration/deceleration parameter (x16) is set too low. The home switch to marker parameter (x07) is not set properly.	5.6
The axis runs into a limit during the home cycle ...	The homing direction parameter is wrong. The home switch is not connected.	5.6
Software limits are not working ...	The home cycle may not have completed yet.	5.6

13.6. General UNIDEX 500 Questions

1. How can I create a custom interface for my application?

   Refer to Chapter 10: Programming Tools.

   - The UNIDEX 500 Windows API functions provide an easy interface that is useful for high level programming in the Windows environment. All of the functions are incorporated in the file **WIN50032.DLL**. The WAPI functions are ideal for developing applications in languages such as Visual BASIC, Delphi or C. For more information and examples, you can refer to the U500 WAPI Help File that is installed with the U500 MMI software package.

   - While the WAPI functions are easier to use, they have higher overhead. The functions at the Windows application layer must call the lower level functions in the U500 quick library (incorporated in **QLB50032.DLL**) to actually communicate with the UNIDEX 500. The UNIDEX 500 quick library functions can be used directly to provide a low-level programming interface to the UNIDEX 500. The functions interact directly with the device driver that handles the communication between the operating system and the UNIDEX 500. All of the functions are incorporated in the dynamic link library qlb50032.dll. The functions are ideal for developing faster applications with little overhead using the C programming language. For more information and examples, you can refer to the U500 QLIB Help File that is installed with the U500 MMI software package.

   - The files 'win50032.dll' and 'qlb50032.dll' should be included in the same path as any application that you create using the WAPI functions. The file 'qlb50032.dll' should be included in the same path as any application that you create using the QLIB functions. These dll's are installed in the u500/mmi/ directory.

2. Where can I find documentation for software changes and programming commands that have not yet been updated in the manual?

   - The U500 online help file is updated with every release of the U500 software. This help file is accessible through the U500 program group, or by pressing the 'F1' key while in the MMI or Toolkit software.

   - The software installation sub-directory "\DOC" also contains a text file "VERSION.DOC" which lists bug fixes and enhancements.

3. Are there LabView drivers available for the UNIDEX 500?

   - Yes. In software versions 5.09 and higher, there is an option during the installation to install the LabView drivers. In previous software versions, the LabView drivers were distributed as a separate installation.

- The U500 VI's are found in the file u500w32.llb. The U500 LabView examples are found in the u500/mmi/example/labview directory. The VI's use a code interface node, (directly accessible in the Unidex_500,CIN.VI) to call the UNIDEX 500 WAPI functions. The Library function input the code interface node determines which function is called. The inputs and outputs that are used are then dependent on the function that is called.
- The files 'win50032.dll' and 'qlb50032.dll' should be included in the same path as any VI's that you use or create. These dll's are installed in the u500/mmi/ directory.

4. How can I find the version numbers for MMI software?

- From the 'Help' Menu on the MMI menu bar, select the 'About' Option. A window will display the version number of the dlls, device driver, and firmware currently being used by the software.

5. How do I use my calibration file when I run my Unidex 500 system?

- First you must check that the calibration file is specified in your project. You can verify this by using the Edit option on the U500 menu bar to edit the project. You must re-initialize the system for the change to take effect.

  Next, you must check that your parameter file is set appropriately. If the calibration file contains linear calibration data, Parameter x15 "Calibration Enable (y/n)" must be set to "yes" for the appropriate axis. If you are using an orthogonality correction table, Parameter x71 "Orthogonality Correction Table Enabled" must be set to yes for the appropriate axes. You must re-initialize the system for the parameter change to take effect.

  Finally, the axes that are affected by the calibration must successfully complete the home cycle before calibration will take effect. See Chapter 5 "Parameters" and Appendix G for more details.

6. How do I setup my Unidex 500 ISA system to use the PC-PSO board?

   Note: The U500 PCI does not require the PC-PSO board to implement PSO functions.

   Refer to Chapter 3 "Software Installation and Fundamentals" for more information.

- You will first have to set up the base address for the PC-PSO card. You can refer to question 1 in this section and follow the same steps used to the set the base address for the Unidex 500 board. The hardware base address jumpers for the PC-PSO board are JP2-JP7. Please refer to the PC-PSO manual Chapter 2, page 4. You must re-initialize the system for the change to take effect.
- Next you must check that the PSO firmware file (pcpso.frm) is specified in your project. You can verify this by using the Edit option on the U500 menu bar to edit the project.

- Also insure that the 50 pin ribbon cable is properly connected between P3 on the UNIDEX 500 board and P3 on the PC-PSO board.
- If you are using the PC-PSO board for position tracking, you must also interface the encoder signals by connecting a ribbon cable between P6 of the PC-PSO card and P6 on the U500 Rev C card.
- The encoder feedback to the PSO can be verified by running the PSO diagnostics.

∇ ∇ ∇

APPENDIX A: GLOSSARY OF TERMS

Appendix A

In This Section:
- Terms Used In This Manual
- Definitions

This appendix contains definitions of terms that are used throughout this manual.

absolute positioning - Absolute positioning is positioning that is done with respect to an initial starting position (typically referred to as the home position) and typically uses a standard coordinate system (using [X,Y] coordinates is an example of absolute positioning). In contrast, incremental (or relative) positioning is done using a series of relative moves. These moves are relative to the previous location rather than a single reference point (for example, relative changes in position [ΔX,ΔY] are examples of incremental positioning).

acceleration feed forward - Acceleration feed forward is a control strategy (represented as a dimensionless gain value that is sometimes used during the motor tuning process) in which acceleration commands are sent directly to the amplifier.

accuracy - Accuracy is the difference between an expected value and an actual value expressed as a percentage.

aer - aer is a case-sensitive prefix that is used with C library function calls. For example, aer_send, aer_initialize, and aer_reset are C functions that use the aer prefix.

amplifier - An amplifier is a hardware device having an output that is a function of the input signal.

axis - An axis is a direction along which movement occurs.

axis calibration - Axis calibration is the process by which the current position of an axis is adjusted to match the actual position (as determined by a laser for example) of the axis.

backlash - Backlash is a movement that occurs between two or more interacting mechanical parts as a result of looseness.

ballscrew - A ballscrew is a precision motion component of mechanical stages and consists of a precisely threaded shaft (or channel) and a housing that rides along the shaft as the shaft is rotated. The housing of a ballscrew contains ball bearings that ride in the channel of the shaft as the shaft rotates. A small tube on the housing recycles the bearings as the shaft rotates. The conversion factor parameter calculation is different for ballscrew systems compared to other systems. (Compare with leadscrew.)

base address - A base address is a number that represents the memory location in the computer where input/output (I/O) information can be stored. All devices (e.g., the U500 card, network cards, tape backup cards, etc.) within a computer must have unique I/O base addresses. The default I/O base address of the U500 card is 0x300 (which represents the 16 memory locations 0x300 through 0x30F). This base address can be

changed using base address jumpers JP4 through JP9 on the U500 card. The base address must also be configured in the UNIDEX 500 Startup software.

BASIC - BASIC is an acronym for Beginner's All-purpose Symbolic Instruction Code and refers to a widely-used programming language that is available in both compiler and interpreter versions. BASIC is a *conversational* language that is considered to be one of the easiest programming languages to learn. A BASIC program can be used to write a customized software interface for the UNIDEX 500, rather than using the U500 software that is supplied with the system. Refer to Chapter 10: Programming Tools for more information.

batch file - A batch file is a file that contains a series of commands (e.g., the AUTOEXEC.BAT file is a batch file).

BB500 Breakout Module - The BB500 Breakout Module is a hardware device that connects directly to the U500 card to provide direct signal access (in the form of screw terminals) when a DR500 chassis is not used.

bit - The term bit is an acronym for "Binary digIT" and represents a single binary number (i.e., a "1" or a "0"). In digital computers, a bit's two states can represent an off state and an on state, a high voltage and a low voltage, the numbers 0 and 1, etc.

brushless motor - Aerotech brushless motors are three-phase, rare earth permanent magnet servo motors which generate a sinusoidal back EMF voltage and are usually referred to as AC brushless motors. Another type, usually referred to as the DC brushless motors, generate a trapezoidal back EMF and produce more torque ripple.

byte - A byte is a common unit of information storage made up of eight binary digits (bits). A byte can be used to represent a single ASCII character (e.g., "A"= 01000001 [binary]) or binary numbers from 00000000 to 11111111 (from 0 to 255 decimal), depending on how it is used.

C - C is a high-level programming language (developed at Bell laboratories) that is able to manipulate a computer at a low level like assembly language. A C program can be used to write a customized software interface for the UNIDEX 500, rather than using the software program that is supplied with the system. If a customized interface program is created using C, the appropriate C library must be included when the program is compiled. Refer to Chapter 10: Creating a Customized Software Interface for more information.

cascading menu - *see pull-down menu*.

circular interpolation - Circular interpolation refers to the UNIDEX 500's ability to coordinate two axes to produce accurate circular motion using minimal reference information (e.g., the center point and a radius).

closed loop system - A closed loop system is a drive system that uses sensors for direct feedback of position and/or velocity. Contrast with open loop system

commutation - Commutation refers to the process by which every other cycle of an alternating current is reversed so that a single unidirectional current is supplied. In the case of motors, commutation refers to the switching of current to motor windings which causes the motor to rotate. In a DC servo motor, this is done mechanically using brushes

and a commutator. An AC brushless motor is electronically commutated using a position feedback device such as an encoder that is mounted to the rotor. Stepping motors are electronically commutated without feedback (in an open loop fashion).

constant velocity motion - Constant velocity motion refers to the U500's ability to perform motion while maintaining a constant velocity during the motion. For an application example, consider an irregularly shaped pattern that requires a series of *perforations* (made using a series of on/off laser pulses, for example). Constant velocity motion ensures that the length of each perforation is the same.

cubic spline interpolation - Cubic spline interpolation is a mathematical process used by the U500 in which a smooth curve (path) is based on two sets of coordinates ([X1,Y1] and [X2,Y2]) on the curve. Unlike linear interpolation, however, a previous coordinate ([X0,Y0]) and following coordinate ([X3,Y3]) are used to determine the curve's slope entering the path and exiting the path.

derivative gain - Derivative gain is a dimensionless motor tuning parameter that serves to dampen the system response by producing a dampening force as long as the system is progressing toward error reduction.

DOS - DOS is an acronym for Disk Operating System--a master control program that runs a computer and acts as a scheduler, providing job, task, data and device management. DOS is a generic term that refers to an operating system. MS-DOS is a single-user operating system for PCs that was designed by Microsoft Corporation. The terms DOS and MS-DOS are frequently interchanged.

double word - A word is a number of bytes that are processed as a single unit by a computer. In the U500, a word consists of two bytes or 16 bits. A double word is twice that amount (i.e., four bytes or 32 bits).

DR500 Chassis/Drive Rack - The DR500 Chassis (or Drive Rack) is a housing for the axis amplifiers (for microstepping, DC brush and AC brushless drivers) and the driver power supply. The DR500 is available in rack mount, panel mount and desktop packaging.

dynamic link library (.DLL) - A dynamic link library (or DLL) is a set of program routines (typically used in Windows environments) that are available to applications at run time. DLLs are only loaded into memory as they are needed (as compared to static libraries that are completely loaded into memory, even if portions of the library are not called). The file WINAER.DLL provides the function declarations for use when creating Windows-based user interfaces from either C or Visual BASIC.

electronic gearing - Electronic gearing is the process of moving one or more slave axes in coordination with a master axis without continuously sending commands from the host program. By establishing a series of relationships between axes, the servo processor will continuously update the positions and velocities of the slave axes based on the commanded motion of the master axis. The master can be a physical axis in the system or a virtual axis used for synchronization purposes only.

encoder - An encoder is a rotary device that transmits a pulsed signal based on the number of revolutions of the device.

faults - A fault is an error condition that occurs when a component of the UNIDEX 500 system operates outside certain parameters. Fault masks are used to allow the U500 system to detect and act on any fault condition of the system. Examples of major fault conditions include position faults, velocity faults, integral faults, RMS over current faults, amplifier faults, and feedback faults.

feedrate error - A feedrate error is a type of fault that is generated by the UNIDEX 500 if the current speed of an axis exceeds a programmable maximum speed (called the *Top feedrate* [x17] parameter). Feedrate errors are necessary because certain stages or motors have a maximum operating speed above which components may be damaged.

fillet - A fillet is a concave junction where two surfaces meet. When machining a part using the U500, a fillet can be created using corner rounding (rather than specifying dimensions for circular motions). This gives a smooth *curved* junction rather than a sharp 90° angle for example.

floating point number format - Floating point number format is a method of representing numbers without defining a fixed number of decimal places. Two common forms of floating point number format are fixed-style format (e.g., 12.345, 0.000001, -2, etc.) and scientific notation (e.g., 12.3E4, -1.2E-3, etc.). The UNIDEX 500 uses fixed-style format for floating point numbers.

G codes - see RS-274 commands.

Hall effect switch - A Hall effect switch is a solid state switch that is activated by a magnetic field. Some AC brushless motors use Hall effect switches.

handwheel - A handwheel is an encoder-based manual control input device that can be used to simplify machine setup or testing.

helical interpolation - Helical interpolation refers to the UNIDEX 500's ability to coordinate three axes to produce accurate helix motion (e.g., an upward circular spiral) using minimal reference information (e.g., the center point and a radius of the circle portion of the spiral and a feedrate).

hexadecimal number format - Hexadecimal number format is a method of representing large numbers using base 16 rather than the standard base 10. In base 16 or hexadecimal number format (often abbreviated "hex"), the number positions represent powers of 16 (rather than powers of 10 in decimal). The decimal number positions (1's, 10's, 100's, 1,000's, 10,000's, etc.) are replaced with hexadecimal number positions (1's, 16's, 256's, 4096's, etc.). Also, while the individual numerals for the decimal system are 0-9, the numerals for the hexadecimal number system (which requires 16 unique "numerals") are 0-9 then A-F (where $A_{16}=10_{10}$, $B_{16}=11_{10}$, $C_{16}=12_{10}$, $D_{16}=13_{10}$, $E_{16}=14_{10}$, and $F_{16}=15_{10}$). For simplicity in this manual, hexadecimal numbers are written with a preceding "0x" rather than using the subscript 16. For example, the hexadecimal number 12A5 is written 0x12A5. Numbers without the preceding "0x" are assumed to be decimal unless otherwise indicated.

home cycle - The home cycle is series of motions that are used to move the specified axes to a hardware referenced position. There are two feedrates (in the form of parameters) associated with the home cycle: the Power On Feedrate and the Normal Home Feedrate. The power on home cycle (the first commanded home cycle following a power up) uses

the *Power on home feedrate* parameter and the normal home cycle (all subsequent home cycles) uses the *Normal home feedrate* parameter.

home marker option - The home marker option is a type of encoder that can be used with stepper motors. This option provides an inexpensive way of establishing a very accurate home reference (usually within 0.1 microns, in most Aerotech equipment). The home marker is protected in a rugged housing that also provides terminal connections for the encoder, the motor and the limit switch.

in-position integrator - An in-position integrator is a motor tuning adjustment that can be used to help remove steady-state position errors as well as reduce the effects of tachometer loop drift. In-position integration is accomplished at a rate that is directly proportional to the velocity loop integrator (Ki).

incremental positioning - Incremental (or relative) positioning is done using a series of relative moves. These moves are relative to the previous location rather than a single reference point (for example, relative changes in position [ΔX,ΔY] are examples of incremental positioning). In contrast, absolute positioning is positioning that is done with respect to an initial starting position (typically referred to as the home position) and typically uses a standard coordinate system (using [X,Y] coordinates is an example of absolute positioning).

inductosyn - An Inductosyn is a rugged, very accurate, multi-pole electromagnetic transducer with an operating principle similar to that of a resolver.

initialization - Initialization is the process in which the current configuration (.CFG) file and the current parameter (.PRM) file are sent down into the memory of the U500 board from the PC. In addition, the actual control program (the *firmware* file U500.JWP) is also sent down to the U500 board. The last step of the initialization process is the restarting of the U500 firmware program. Initialization can be accomplished in any one of four ways: (1) selecting the Reset option of File menu, selecting the F9 Reset soft key at the bottom of the main screen, selecting the Reset option from the Functions menu, or pressing the F9 function key on the keyboard.

integral error - Integral error is the summation of position errors over time. An integral error fault is generated if the integral error for an axis exceeds a programmable maximum integral error value (parameter x20). Integral error is reset every time the axis is reset.

integral gain - Integral gain is a dimensionless motor tuning parameter that serves to help remove steady-state position errors as well as reduce the effects of tachometer loop drift.

IRQ - IRQ (interrupt request) is a term associated with generating an interrupt request to the PC. A PC has many IRQs (e.g., IRQ3 and IRQ4 are typically configured as COM ports on the PC, IRQ7 is typically configured as the LPT port, et. al.). Although the U500 does not use such interrupts, custom software applications may. In these cases, the interrupt number used by the custom software program (to interrupt the PC) must be selected with jumpers on the UNIDEX 500 board as well as configured in the U500 software.

iSBX expansion port - The iSBX expansion port (P4 on the UNIDEX 500 card) is a standard Intel interface that uses either an 8 or 16 bit data bus and is used primarily for communications-oriented additions to the system. The iSBX expansion port has a communications data rate of approximately 1 MB/sec (1,048,576 bytes per second).

jog move - A jog move is a momentary movement of an servo drive to provide manual control of axis motion.

joystick - A joystick is manual input control device that digitizes a path using two axes. A joystick offers direct motion control for easy machine setup and testing.

jumpers - Jumpers are hardware *ties* that you manually position onto different posts to configure the hardware platform. Jumpers on the UNIDEX 500 board are used to configure the base address, the encoder sampling frequency, termination resistors, and other features.

leadscrew - A leadscrew is a motion component of stages and consists of a threaded shaft and a housing that rides along the shaft as the shaft is rotated. The housing of a leadscrew contains a similar thread which rides along the shaft thread as the shaft rotates. Leadscrews are more economical, but less accurate than ballscrews.

LED - LED is an acronym for light-emitting diode. An LED is a semiconductor diode that converts electrical energy into visible electromagnetic radiation. The UNIDEX 500 board has an LED (visible from the back of the PC after installation) that is used for diagnostic purposes.

linear interpolation - Linear interpolation is a mathematical process used by the U500 in which a straight line (path) is based on two sets of coordinates ([X1,Y1] and [X2,Y2]) on the line. Unlike cubic spline interpolation (which uses two additional coordinates to determine the slope of the smoothed curve), linear interpolation only uses two sets of coordinates and generates a straight (not smoothed) path.

linear motor driver - A linear motor driver is a non-switching type of DC servo amplifier that drives the motor with direct current. Linear amplifiers do not generate electrical noise or switching losses in the motor. They do, however, have a higher rate of power dissipation than a pulse-width modulated (PWM) amplifier.

M codes - see RS-274 commands.

machine step - A machine step is the smallest feedback device step that can be taken. This is the smallest possible increment of movement as measured by the feedback device.

microstepping - Microstepping is a technique for driving stepping motors more smoothly and with higher resolution than full step control. Current is divided in a sine-cosine fashion between motor phases to provide intermediate positions between full step positions.

multitasking - Multitasking is a software technique that gives several functions (or tasks) the appearance of individually having sole access to the resources of the system (e.g., the microprocessor). In its simplest form, a multitasking system assigns a small time slice to each task in a round-robin fashion. Only one task at a time has access to the multitasking system's resources. When each successive task has had the opportunity to use the system resources (for a brief period), the cycle repeats.

notch filter - A notch filter is a software filter that is used to remove a section of frequencies in order to stabilize a system with a known mechanical resonance.

off-line U500 programming - Off-line programming is a method of writing UNIDEX 500 programs remotely (away from the Startup software [from the Program menu] and away from the PC that holds the U500), rather than using the built-in program editor of Startup. In off-line programming, the program file is usually saved to a floppy disk as a .PRG file. This file can be transferred to the U500 PC (using a standard DOS file copying procedure) and executed.

OP500 - The OP500 is an optional cable that is used to connect the UNIDEX 500 controller card to the DR500 chassis.

open loop system - An open loop system is a drive system that does not employ feedback sensors to monitor position or velocity. Most stepper motor applications are open loop (they have no feedback). The commanded position is the assumed motor position. Contrast with closed loop system.

operator - (1) An operator is one who uses the UNIDEX 500 system.

operator - (2) An operator is a programming element that is used to link terms in an expression. Programming operators include the standard arithmetic operators (e.g., +, -, * and /), comparison operators (e.g., < and >) and Boolean operators (e.g., AND, OR and NOT) and others.

orthogonality - Orthogonality is a state of two axes in which one is perpendicular to the other. The UNIDEX 500 provides orthogonality correction capabilities that allow an axis to be corrected (using absolute machine step correction data) based on a position-dependent axis. Orthogonality correction, if used, is incorporated into the axis calibration (.CAL) file.

plane - A plane is an axis or group of axes that can be coordinated (for example, a particular action of one plane can trigger an action on another plane) or independent (for example, one plane can be milling a part while another plane is etching circles). Planes can also be virtual planes, which are not linked to any particular axis, but act as queues or buffers.

point-to-point motion - Point-to-point motion simply involves specifying a target position. After the target position is commanded, the controller strives to attain that position with no time or path constraints.

popup window - A popup window is a box that appears “on top of” the existing screen or display. Popup windows typically contain fields (for additional information to be supplied), buttons (to acknowledge or cancel a particular operation, for example) or a combination of both.

position error - Position error is the difference between the commanded position of an axis and the feedback position (i.e., the difference between the desired position and the actual position). A position error fault occurs if the current position error exceeds a programmable maximum position error (parameter x19). Position error is measured in machine steps.

position synchronized output card - The position synchronized output card is an optional PC-bus based card that can be used in conjunction with the U500 (via connection P3 on the U500 card) to provide programmable laser-firing control.

power-up home cycle - The power-up home cycle refers to the positioning of an axis to a known, hardware-referenced position when the axis is sent “home” for the first time after a power up. Subsequent home commands use the *runtime home cycle*.

program - A program is a set of instructions that are carried out in some predefined logical order. A UNIDEX 500 program is a sequential list of UNIDEX 500 programming commands (see Chapter 7 and Appendix C) which tell the U500 control board how to perform specific motions for a particular application. UNIDEX 500 programs may be created/edited on-line (from within the Program menu of the Startup software) or off-line (using any standard ASCII text editor). U500 program files (which are optional components of project or .PRJ files) use .PRG as their extension.

program step - A program step is the smallest programmable increment of motion that can be commanded. A program step equals the programming units * 10^{ndec} where "ndec" is the number of decimal digits set by parameters 029, 030, 047 and 048 for Metric mode, and parameters 065, 066, 083 and 084 for English mode.

program unit - A program unit is a user-defined measurement unit such as inches, millimeters, degrees, etc. Program units are used within the application program and provide the operator with flexibility and ease of use. For example, it is more meaningful for an operator to command a “100 mm” move than it is to command a “752 machine step” move.

project - a project is a filename (having a .PRJ extension) that references a collection of two or three of the following file types: a configuration (.CFG) file, a parameter (.PRM) file and an optional firmware (.JWP) file. A project may be opened as a single unit, rather than opening the individual components (files) associated with it. Component files (.CFG, .PRM and .JWP files) may be shared among projects to decrease the duplication of information. For example, two unique projects may share the same configuration file and parameters file, but may have unique program files.

proportional gain - Proportional gain is a dimensionless motor tuning parameter that produces an output which is related to the Velocity Error in the servo loop.

pull-down menu - A pull-down menu is a vertical list of commands. When the menu is closed (or “rolled up”), only the menu name is visible on the menu bar. When the menu name is selected, the menu “unrolls” and the list of commands is displayed. Some options in pull-down menus are the names of other menus (called *cascading menus*).

qms - QMS is an abbreviation for quarter millisecond - a unit of time that is used when determining values such as velocity error, for example, which is measured in machine units per quarter millisecond (i.e., machine units/qms). One qms is equivalent to 0.25 milliseconds.

quadrature - Quadrature is the state of two signals that are displaced 90 degrees with respect to each other. In most rotary incremental optical encoders, light (from an LED, for example) is measured after it is passed through slits in a grating disk (which is attached to the axis being measured). Typically, two tracks on the disk have their gratings displaced 90 degrees with respect to each other (that is, the tracks are said to be in quadrature).

registers - In the U500, registers are used by the Startup interpreter to designate axis positions. Relative position registers ($XRP, $YRP, $ZRP and $URP) represent the commanded axis positions (in machine steps) with respect to the software home position.

Absolute position registers ($XAP, $YAP, $ZAP and $UAP) represent the commanded axis positions (in machine steps) with respect to the hardware home position.

reset - *see initialization.*

resolution multiplier card (RMX-PC) - The RMX-PC is a PC bus, 2-channel, 256-times resolution multiplier card that can be used with the UNIDEX 500. This card multiplies the sinusoidal position feedback by up to 256 times. It is ideal for high-resolution applications such as wafer inspection.

resolver - A resolver is a two-phase, rotary, electromagnetic transducer in which inductive coupling (between the rotor and stator windings) and trigonometric principles are employed to provide absolute position information over one electrical cycle (which is one revolution for "single-step" resolvers).

resolver-to-digital card (RDP-PC) - The RDP-PC card is an optional PC-based R/D card that is used to receive resolver or Inductosyn feedback. Resolution is selectable among 10-bit, 12-bit, 14-bit or 16-bit.

RMS current trap - RMS current trap is an error that occurs if the current being commanded to a motor exceeds a programmable limit (see parameters x48 and x49). RMS current trap is analogous to a software “fuse”. Essentially, this fault functions the same as a physical fuse, but is done through software. One obvious advantage is that a “software fuse” does not have to be replaced like a physical fuse.

RS-274 - The term RS-274 refers to a set of standardized motion control programming commands. These commands consist of the letter “G” or “M” followed by a one- or two-digit number (e.g., M47, G3, M2, G70, etc.). A subset of the RS-274 command set is available for certain UNIDEX 500 program commands for programmers who may be more familiar with the G/M codes rather than the corresponding UNIDEX 500 programming commands. Not all U500 programming commands have an RS-274 G code or M code counterpart. Refer to Chapter 7: Programming for more information.

runtime home cycle - The runtime home cycle refers to the positioning of an axis to a known, hardware-referenced position. The UNIDEX 500 uses this sequence for all home cycles except when the axis is sent “home” for the first time after a power up. In this latter case, the *power-up home cycle* is used instead.

servo control system - A servo control system (servo loop) is a motion control system which continuously compares desired position/velocity to actual position/velocity and produces an error correction command. Servo systems use sensors to feedback actual position/velocity.

shaft runout - Shaft runout is an expression of the total indicated reading of wobble or nonconcentricity as measured at the end of a motor shaft when the shaft is rotated one complete revolution.

soft keys - Soft keys are software “buttons” that are analogous to function keys on a standard PC keyboard. In the U500 software, eight soft keys are located across the bottom of the main window and perform various functions. Clicking the mouse on these soft keys (F2 through F9) has the same effect as pressing the respective function keys on the keyboard.

software - The term software refers to a computer program. Contrast software with hardware, the physical machinery, components and support peripherals through which the software runs.

spherical interpolation - Spherical interpolation refers to the UNIDEX 500's ability to coordinate multiple axes to produce accurate spherical motion using minimal reference information (e.g., the center point and a radius of the sphere).

task - A UNIDEX 500 task is one of four sets of instructions that are executed sequentially at such a high speed that each task has the impression that it alone has full access to all of the microprocessor's time.

time-based motion - A time-based motion is a motion that arrives at a specified location in a desired amount of time. After the target position of a move is programmed, the controller chooses any speed to achieve that position on time.

traps -*See* faults.

tuning - Tuning is the process of optimizing the operation of a servo system.

variables - Variables are programming terms that are used as temporary storage locations for calculations. Direct variables (V0 through V255) are general purpose, double precision storage locations. Indirect variables (VV0 through VV255) are used to indirectly address other variables. For example, if V35=999 and V1=35, then you can indirectly address the contents of V35 using the statement V0=VV1. In this case, the content of V1 (35) is used as an index to V35. The value of V35 (999) is placed in V0.

velocity error - Velocity error is the difference between the commanded velocity and the velocity derived from the feedback position (i.e., the difference between the desired velocity and the actual velocity). Velocity error is measured in machine steps per quarter millisecond (machine steps/qms). A velocity error fault occurs if, at any time, the velocity error of the system exceeds a programmable velocity error (specified by parameter x18).

velocity feed forward - Velocity feed forward is a control strategy (represented as a dimensionless gain value that is sometimes used during the motor tuning process) in which current velocity disturbances are converted into corrective actions now in order to minimize the future effects of the disturbances.

velocity profiled motion - Velocity profiled motion is a move of a programmed distance and speed from the current position. Velocity profiled motions are executed only after the previous motion has reached its deceleration point.

word - A word is a number of bytes that are processed as a single unit by a computer. In the U500, a word consists of two bytes or 16 bits.

∇ ∇ ∇

APPENDIX B: WARRANTY AND FIELD SERVICE

Appendix B

In This Section:

- Laser Product Warranty
- Return Products Procedure
- Returned Product Warranty Determination
- Returned Product Non-warranty Determination
- Rush Service
- On-site Warranty Repair
- On-site Non-warranty Repair

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, whether 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.

Laser Products

Aerotech, Inc. warrants its laser products to the original purchaser for a minimum period of one year from date of shipment. This warranty covers defects in workmanship and material and is voided for all laser power supplies, plasma tubes and laser systems subject to electrical or physical abuse, tampering (such as opening the housing or removal of the serial tag) or improper operation as determined by Aerotech. This warranty is also voided for failure to comply with Aerotech's return procedures.

Return Procedure

Claims for shipment damage (evident or concealed) must be filed with the carrier by the buyer. Aerotech must be notified within (30) days of shipment of incorrect materials. No product may be returned, whether in warranty or out of warranty, without first obtaining approval from Aerotech. No credit will be given nor repairs made for products returned without such approval. Any returned product(s) must be accompanied by a return authorization number. The return authorization number may be obtained by calling an Aerotech service center. Products must be returned, prepaid, to an Aerotech service center (no C.O.D. or Collect Freight accepted). The status of any product returned later than (30) days after the issuance of a return authorization number will be subject to review.

Returned Product Warranty Determination

After Aerotech's examination, warranty or out-of-warranty status will be determined. If upon Aerotech's examination a 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 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
TWX: (710) 795-3125

∇ ∇ ∇

APPENDIX C: PARAMETER SUMMARY

Appendix C

In This Section:

- General Parameters.....C-1
- Parameters Dedicated to Planes 1, 2, 3, and 4.....C-4
- Axis Parameters.....C-5

The following section provides a quick reference for UNIDEX 500 parameters. UNIDEX 500 parameters are divided into three logical groups: general parameters, plane parameters, and axis parameters.

General parameters range from 000 to 017 and 090 through 098. Plane parameters range from 018 to 089 (consecutive groups of 18 parameters each are assigned to planes 1 through 4, that is parameters 018-035 are assigned to plane 1, parameters 036-053 are assigned to plane 2, etc.). Axis parameters range from x00 to x99 where "x" corresponds to the axis number 1-4 (therefore, axis parameters 100, 200, 300 and 400 represent the same parameter for the four different axes, respectively).

Summaries of general, plane and axis parameters are listed in Table C-1, Table C-2, and Table C-3, respectively. Accompanying each parameter number is the parameter name, a description of the parameter's function, the default value of the parameter and the page number from this manual where additional information can be found.

Table C-1. General Parameters

#	Parameter	Function	Default	Page
000	Number of contour planes	Selects the number of programming memory areas	1	5-17
001	Keep axis position after reset	Maintains (y) or clears (n) position information after a reset	no	5-6
002	MFO, 0 no MFO-pot, 1-255 pot offset	Enables (1-255)or disables (0) manual feedrate override option	0	5-34
003	Axis 1 map to plane n as (XYZU)	Maps axis/drive to a contour plane and assigns a programming name	1,X (map axis 1 to plane 1 as X)	5-19
004	Axis 2 map to plane n as (XYZU)	Maps axis/drive to a contour plane and assigns a programming name	1,Y (map axis 2 to plane 1 as Y)	5-19

Table C-1. **General Parameters (continued)**

#	Parameter	Function	Default	Page
005	Axis 3 map to a plane n as (XYZU)	Maps axis/drive to a contour plane and assigns a programming name	1,Z (map axis 3 to plane 1 as Z)	5-19
006	Axis 4 map to a plane n as (XYZU)	Maps axis/drive to a contour plane and assigns a programming name	1,U (map axis 4 to plane 1 as U)	5-19
007	Axis 1 gantry (y/none) slave {2, 3, 4}	Pairs axes in a master/slave relationship	none	5-20
008	Axis 2 gantry (y/none) slave {1, 3, 4}	Pairs axes in a master/slave relationship	none	5-20
009	Axis 3 gantry (y/none) slave {1, 2, 4}	Pairs axes in a master/slave relationship	none	5-20
010	Axis 4 gantry (y/none) slave {1, 2, 3}	Pairs axes in a master/slave relationship	none	5-20
011	Axis 1 rollover machine steps	Establishes a value at which the position register will return to zero	0	5-7
012	Axis 2 rollover machine steps	Establishes a value at which the position register will return to zero	0	5-7
013	Axis 3 rollover machine steps	Establishes a value at which the position register will return to zero	0	5-7
014	Axis 4 rollover machine steps	Establishes a value at which the position register will return to zero	0	5-7
015	PSO mailbox dual-port RAM base address	Permits proper communication between PC-PSO and U500	0	5-8
016	PSO-host and PC interface address	Permits proper communication between PC-PSO and U500	0	5-9
017	Spare			
090	A/D channel 1 - joystick deadband	Defines deadband associated with center position of joystick	0	5-29
091	A/D channel 1 – joystick center position	Specifies the center position of the A/D inputs used for joystick mode	0	5-30
092	A/D channel 2 - joystick deadband	Defines deadband associated with center position of joystick	0	5-29
093	A/D channel 2 – joystick center position	Specifies the center position of the A/D inputs used for joystick mode	0	5-30
094	A/D channel 3 - joystick deadband (joystick vertical axis)	Defines deadband associated with center position of joystick	0	5-29

Table C-1. General Parameters (continued)

#	Parameter	Function	Default	Page
095	A/D channel 3 – joystick center position (joystick vertical axis)	Specifies the center position of the A/D inputs used for joystick mode	0	5-30
096	A/D channel 4 - joystick deadband (joystick horizontal axis)	Defines deadband associated with center position of joystick	0	5-29
097	A/D channel 4 – joystick center position (joystick horizontal axis)	Specifies the center position of the A/D inputs used for joystick mode	0	5-30
098	Safe zone output bit (0-8)	Specifies which U500 output to turn on (low) when all axes are in specified safe zone	0	5-30
099	Option board setup code	Indicates which option board (4EN Option or iSBX encoder card) is being used	0	5-31
500	User interrupt setup code	Sets the usage of the user interrupt input	0	5-32
501	Abort on input high	Generates a global abort when input 1-16 is high	0	5-32

Table C-2. Parameters Dedicated to Planes 1, 2, 3, and 4

Plane Numbers							
1	**2**	**3**	**4**	**Parameter**	**Function**	**Default**	**Page**
018	036	054	072	Plane n indexing segment time	Specifies the time (in ms) allocated for each indexing segment during trajectory generation	10 ms	5-23
019	037	055	073	Plane n contour ramping time	Sets acceleration/deceleration time for contour motion	150 ms	5-35
020	038	056	074	Plane n Metric system	Sets system for default use of Metric/English scale factor	yes (Metric)	5-10
021	039	057	075	Linear type ac/de	Sets system default for linear acceleration/deceleration	no	5-24
022	040	058	076	Contour feedrate	Sets the default feedrate for all contour type motion	16.0	5-36
023	041	059	077	X point-to-point feedrate	Sets default feedrate for all index type moves of the X axis	16.0	5-36
024	042	060	078	Y point-to-point feedrate	Sets default feedrate for all index type moves of the Y axis	16.0	5-36
025	043	061	079	Z point-to-point feedrate	Sets default feedrate for all index type moves of the Z axis	16.0	5-36
026	044	062	080	U point-to-point feedrate	Sets default feedrate for all index type moves of the U axis	16.0	5-36
027	045	063	081	Clamp feedrate	Sets the maximum feedrate for all contour type motion	256.0	5-25
028	046	064	082	Corner rounding non-ramp time	Sets the amount of time remaining for deceleration at which the next block of motion will begin	150 ms	5-26
029	047	065	083	Metric mode number of decimal digits	Number of digits to follow the decimal point when in the Metric mode	3	5-11
030	048	066	084	English mode number of decimal digits	Number of digits to follow the decimal point when in the English mode	4	5-12
031	049	067	085	Contouring mode	Select between normal and alternate contouring mode	0	5-28
032	050	068	086		Spare		
033	051	069	087		Spare		
034	052	070	088		Spare		
035	053	071	089		Spare		

The parameters for axes/drives are numbered x00 through x99 where x represents the axis number (1, 2, 3 or 4). Since the function of each of these parameter sets is identical, this section will list x00 through x99 as being representative of all axes/drives.

Table C-3. Axis Parameters

#	Parameter	Function	Default	Page
x00	Metric conversion factor	Scale factor to convert metric programming units to machine steps	1.0	5-93
x01	English conversion factor	Scale factor to convert English programming units to machine steps	1.0	5-93
x02	Home direction is CCW	Sets the direction the motor will turn to find the home limit switch during a home cycle	yes	5-40
x03	Home switch normal open	Configures axis for the type of home limit switch being used	yes	5-41
x04	Power on home feedrate	Sets feedrate for a home move that immediately follows a power-up	25	5-42
x05	Normal home feedrate	Sets feedrate for a runtime home move	25	5-43
x06	Home offset	Number of machine steps between home position and the home marker (null)	0	5-43
x07	Home switch to marker	Distance that the axis will move at maximum speed before slowing to look for marker (null)	0	5-44
x08	Home/limit switch debounce	Time required for the home/limit switch signal to remain "off" for UNIDEX 500 to consider it inactive	100	5-45
x09	Limit switch normally open	Configures the axis for the polarity of the limit switch being used	yes	5-45
x10	Switch to mechanical stop	The number of machine steps separating the limit switch from the mechanical stop	2000	5-46
x11	Positive (+) move is CW	Establishes relationship between a positive move and CW motor rotation	yes	5-96
x12	Positive (+) jog is same direction as + move	Establishes relationship between a positive jog move and CW motor rotation	yes	5-97
x13	Pause enable in freerun	Enables or disables the "pause" key while the axis is in freerun	yes	5-97
x14	MFO enable in freerun	Determines whether the optional MFO pot can be used to affect freerun speed	yes	5-98
x15	Calibration enable	Enables the UNIDEX 500 to process calibration data	no	5-98
x16	Max ac/de	Sets maximum acceleration/deceleration speed for ramping during a freerun or a home cycle	1.0	5-51
x17	Top feedrate	Sets highest speed for which the axis is mechanically configured. Used as basis for jog feedrate	440	5-52

Table C-3. Axis Parameters (Cont'd)

#	Parameter	Function	Default	Page
x18	Max velocity error	The maximum amount of velocity error (difference between actual and programmed velocity) before a "Velocity Trap" is produced	1,000	5-53
x19	Max position error	The maximum amount of position error (difference between the actual position and the requested position) before a "Position Trap" is produced	4,000	5-54
x20	Max integral error	The maximum acceptable amount of integral error	655,360	5-55
x22	CCW software limit	Establishes a CCW software travel distance referenced from the hardware home	-2^{47}	5-46
x23	CW software limit	Establishes a CW software travel distance referenced from the hardware home	2^{47}	5-46
x24	Notch filter	Enables or disables the notch filter function	no	5-83
x25	Kpos	Position loop gain	50	5-83
x26	Ki	Velocity loop integral gain	5000	5-83
x27	Kp	Velocity loop proportional gain	100,000	5-84
x28	Vff	Sets the velocity feed forward value used in the servo loop	256	5-84
x29	Aff	Sets the acceleration feed forward value used in the servo loop	0	5-84
x30	Notch filter N0	Sets the notch filter coefficient N0	0	5-85
x31	Notch filter N1	Sets the notch filter coefficient N1	0	5-85
x32	Notch filter N2	Sets the notch filter coefficient N2	0	5-85
x33	Notch filter D1	Sets the notch filter coefficient D1	0	5-85
x34	Notch filter D2	Sets the notch filter coefficient D2	0	5-85
x35	In position dead-band	Sets the number of machine steps forming a position deadband. If the axis position is less than this value the UNIDEX 500 will consider it "in position"	10	5-99
x37	Backlash	Position accuracy is enhanced by setting this value to the number of machine steps necessary to compensate for any backlash present in the system	0	5-100

Table C-3. Axis Parameters (continued)

#	Parameter	Function	Default	Page
x38	Primary/position feedback channel	Configures the system for the channel number of the primary feedback device being used	Axis 1 = 1 Axis 2 = 2 Axis 3 = 3 Axis 4 = 4	5-72
x39	Secondary/velocity feedback channel	Configures the system for the channel number of the secondary feedback device	0	5-73
x40	Primary feedback setup code	Specifies the bit resolution when a resolver is used as the primary feedback device	3 (14 bit)	5-74
x41	Secondary feedback setup code	Specifies the bit resolution when a resolver is used as the secondary feedback device	3 (14 bit)	5-74
x42	Drive type	Configures the UNIDEX 500 for the motor type	0	5-75
x43	AC brushless motor commutation factor	If an AC brushless motor is being used, the electrical cycles per revolution (commutation factor) is set by this parameter	4	5-76
x44	Encoder feedback (steps/rev/(× 4))	Machine steps/rev, used for commutation	4,000	5-77
x45	AC brushless motor phase offset	Adjusts phasing for AC brushless motors	0	5-77
x46	Stepper high current (0-100%)	Sets the peak of the sinusoidal current command	70	5-78
x47	Stepper low current (0-100%)	If the command velocity is zero for 500 ms the current will go to this parameter setting	35	5-78
x48	RMS current trap	Specifies the percentage of maximum output voltage that may be commanded to the amplifier before a fault is generated	30	5-56
x49	RMS current sample time	If the RMS current exceeds the specified RMS level longer than the sample time specified by this parameter, an RMS overcurrent fault is generated	10,000	5-58
x50	Joystick: high speed	Joystick "high" speed setting	40,960	5-101
x51	Joystick: low speed	Joystick "low" speed setting	2,560	5-101
x52	Absolute mode scale	Sets a "window" of Joystick axis movement for fine positioning	10	5-102
x53	Clamp current output	The maximum output voltage of the control loop may be clamped to this value to limit amplifier current and motor torque	100	5-59

Table C-3. Axis Parameters (continued)

#	Parameter	Function	Default	Page
x54	Which output bit for AUX OUTPUT	Specifies the output bit to be set if an AUX.OUTPUT fault is met	Axis 1 = 1, Axis 2 = 2, Axis 3 = 3, Axis 4 = 4	5-60
x55	Error fault mask	A global bit pattern enabling or disabling detection of a fault condition	FFFFFFFF319F	5-106
x56	Disable axis fault mask	Disables the axis if any condition in the fault mask is true	FFFFFFF0EF87	5-106
x57	Interrupt PC fault mask	A PC bus interrupt condition is generated if the fault mask is true	FFFFFFF00000	5-106
x58	AUX OUTPUT fault mask	A user specified output bit is set if the fault mask is true	FFFFFFF00000	5-107
x59	Halt queue fault mask	The UNIDEX 500 will stop reading information from the internal queue if the Fault Mask is true	FFFFFFF08E00	5-107
x60	Abort motion fault mask	The axis will ramp to a stop and wait for an acknowledge if this fault mask is true	FFFFFFFF9E78	5-107
x61	Enable brake fault mask	The brake output is immediately activated if this fault mask is true	FFFFFFF00000	5-108
x62	Servo Loop Update Rate	This parameter specifies how often the servo control loop is to be updated by the UNIDEX 500	1	5-90
x63	Stepper microstepping resolution	Sets the microstepping resolution of an open loop stepper	4,000	5-78
x64	Stepper encoder verification	Specifies whether or not encoder verification is enabled	yes	5-79
x65	Stepper encoder speed	Specifies a correction speed in microsteps per millisecond for each axis that is configured as an open loop stepper	1 microsteps/ms	5-79
x66	AC brushless motor phase advance base speed	Base speed as specified by the user in machine steps / ms	0	5-80
x67	AC brushless motor base speed advance (degrees)	Base speed advance in degrees at the specified base speed. (Number of degrees that the torque angle is increased)	0	5-81
x68	AC brushless motor phase speed	Phase speed as specified by the user in machine steps / ms	0	5-81
x69	AC brushless motor phase speed advance (degrees)	Phase speed advance in degrees at the specified phase speed. (Number of degrees that the torque angle is increased)	0	5-81
x70	Drive fault signal active low	Specifies the polarity of the drive fault signal input to the UNIDEX 500	yes	5-61

Table C-3. Axis Parameters (continued)

#	Parameter	Function	Default	Page
x71	Orthogonality correction table enabled	Used to enable and disable the use of orthogonality data in a calibration (.CAL) file	no	5-102
x72	2-D error mapping enabled	Used to enable and disable 2-dimensional error mapping of a pair of axes	no	5-103
x73	Enable Vel/Accel feedforward during home	Used to enable or disable velocity and acceleration feedforward during home cycle	yes	5-47
x74	Use home limit during home cycle	Used to accept or ignore the home limit input	no	5-48
x75	Safe zone - limit	Specifies safe zone "-" limit for an axis	0	5-48
x76	Safe zone + limit	Specifies safe zone "+" limit for an axis	0	5-48
x77	Home limit switch debounce	Specifies deceleration distance when moving out of home or limit switch	750	5-49
x78	Servo loop type	Changes the configuration of the servo loop	0	5-90
x79	Primary current command offset	Provides DC offset to current command	0	5-81
x80	Secondary current command offset	Provides DC offset to current command	0	5-81
x81	Home marker search speed	Speed at which and axis searches for home marker	2000 mach steps/s	5-49
x82	Encoder multiplication	Used to change position feedback resolution or invert polarity (-1)	0 mV	5-81
x83	Filter time constant	Used in conjunction with the alternate contour mode to activate an exponential filter on the specified axis	0	5-90
x84	Auxiliary output active high	Sets the active state of the auxiliary output bit	yes	5-61
x85	Reverse Joystick Direction	Controls relation between joystick direction and commanded motion.	no	5-103

For additional information about UNIDEX 500 parameters, refer to Chapter 5: Parameters. For information on how to edit and/or view parameter values, refer to Chapter 4: The Software Interface.



∇ ∇ ∇

APPENDIX D: REV_ BOARD TECHNICAL DETAILS

Appendix D

In This Section:

- Introduction D-1
- Test Points D-1
- Jumper Configurations..... D-3
- Encoder Signal Specifications D-6
- Encoder Signal Pinouts..... D-10
- Amplifier Enable Outputs..... D-11
- Main Connector Pinout of the UNIDEX 500 ..... D-12

D.1. Introduction

The current version of the UNIDEX 500 control card is Revision C (Rev C). The previous production version, Revision _ (Rev_), differs physically from Rev C board. Appendix D is a reference resource that documents several technical aspects of the Rev_ board.

D.2. Test Points

Test points are located at the top of the UNIDEX 500 control board. They are used as an aid in troubleshooting the control board and to gain easy access to the UNIDEX 500 signals.

This section arranges test points into the following functional groups:

- Motor Related Test Points
- PC Interface to the UNIDEX 500
- DSP 56001 Chip Test Points
- Miscellaneous Test Points.

Test points for these functional groups are listed and explained in the tables that follow.

Table D-1. Motor Related Test Points

Test Point	Meaning
TP24	Axis 1 marker pulse
TP27	Axis 2 marker pulse
TP30	Axis 3 marker pulse
TP32	Axis 4 marker pulse
TP10	Channel 1 sine after RS-422 receiver
TP18	Channel 1 cosine after RS-422 receiver
TP12	Channel 2 sine after RS-422 receiver
TP19	Channel 2 cosine after RS-422 receiver
TP11	Channel 3 sine after RS-422 receiver

Table D-1. Motor Related Test Points (Continued)

Test Point	Meaning
TP17	Channel 3 cosine after RS-422 receiver
TP16	Channel 4 sine after RS-422 receiver
TP14	Channel 4 cosine after RS-422 receiver
TP6	Encoder sampling frequency (either 2.5,5, or 10 MHz) (See jumper JP18)
TP45	Axis 1 primary current command output (ICMD1B)
TP41	Axis 1 secondary current command output (ICMD1A)
TP44	Axis 2 primary current command output (ICMD2B)
TP40	Axis 2 secondary current command output (ICMD2A)
TP43	Axis 3 primary current command output (ICMD3B)
TP47	Axis 3 secondary current command output (ICMD3A)
TP42	Axis 4 primary current command output (ICMD4B)
TP46	Axis 4 secondary current command output (ICMD4A)

Table D-2. PC Interface to the UNIDEX 500

Test Point	Meaning
TP13	Host request active low (not used, normally high). (freq_n)
TP15	Host enable active low. Indicates PC is attempting to communicate with U500. (hen_n)
TP20	Host read/write. Indicates direction of PC/UNIDEX 500 communication. (hr/w_n)
TP39	I/O board base address true (active low)

Table D-3. DSP 56001 Chip Test Points

Test Point	Meaning	Test Point.	Meaning
TP21	56001 wr_n	TP34	56001 rd_n
TP22	56001 bs_n	TP8	Program RAM select (active low)
TP23	56001 wt_n	TP9	Data RAM select (active low)
TP25	56001 ps_n	TP31	IRQA
TP29	56001 ds_n	TP28	IRQB

Table D-4. Miscellaneous Test Points

Test Points	Meaning
TP4	Signal common
TP5	Cable interlock send. Normally low
TP7	Brake output signal before O.C. buffer. Brake is off when low
TP33	U500 reset active low. LED "on" indicates reset low
TP35	A/D channel 4
TP36	A/D channel 3
TP37	A/D channel 2
TP38	A/D channel 1
TP26	A/D select

D.3. Jumper Configurations

The following two tables summarize all of the jumpers on the UNIDEX 500 Rev_ Board. Relative Jumper locations and the default settings are illustrated in Figure D-1. An asterisk (*) following a jumper setting indicates the default position.

The jumpers listed in Table D-5 are used to set up the PC base address.

Each UNIDEX 500 PC Board must have a unique address.



Table D-5. PC Base Address Jumper Settings

PC I/O Base Address	JP4	JP5	JP6	JP7	JP8	JP9
200-20F (game)	2-3	1-2	1-2	1-2	1-2	1-2
210-21F (expansion)	2-3	1-2	1-2	1-2	1-2	2-3
300-30F (prototype)*	2-3	2-3	1-2	1-2	1-2	1-2
310-31F(prototype)	2-3	2-3	1-2	1-2	1-2	2-3
350-35F	2-3	2-3	1-2	2-3	1-2	2-3
360-36F	2-3	2-3	1-2	2-3	2-3	1-2

* Default Setting

Table D-6. Jumper Configurations

Jumper Number	Setting	Explanation
JP1	1-2	iSBX interrupt MINT1
	2-3	iSBX interrupt MINT0
JP2	1-2*	Enable watchdog timer to DAC strobe
	2-3	Override watchdog timer by connecting to clock
JP3A	IN	Interrupt IRQ12 (COM)
	OUT*	Interrupt not selected
JP3B	IN	Interrupt IRQ11 (COM)
	OUT*	Interrupt not selected
JP3C	IN	Interrupt IRQ10 (LPT)
	OUT*	Interrupt not selected
JP3D	IN	Interrupt IRQ7 (AT unassigned)
	OUT*	Interrupt not selected
JP3E	IN	Interrupt IRQ4 (AT unassigned)
	OUT*	Interrupt not selected
JP3F	IN	Interrupt IRQ3 (AT unassigned)
	OUT*	Interrupt not selected
JP10	IN*	Amp 4 - enable low
	OUT	Amp 4 - enable high
JP11	IN*	Amp 3 - enable low
	OUT	Amp 3 - enable high
JP12	IN*	Amp 2 - enable low
	OUT	Amp 2 - enable high
JP13	IN*	Amp 1 - enable low
	OUT	Amp 1 - enable high
JP14	1-2*	+12V PC bus
	2-3	+12V from external source
JP15	1-2*	-12V from PC bus
	2-3	-12V from external source
JP16	IN*	Marker qualified with cosine
	OUT	Direct marker pulse
JP17	OUT	Direct marker pulse
	1-10	Set iSBX option to 80 KHz
	2-9	Set iSBX option to 40 KHz
	3-8	Set iSBX option to 20 KHz
	4-7	Set iSBX option to 10 KHz
	5-6	Set iSBX option to 10 MHz
JP18	1-6	10 MHz encoder sampling
	2-5*	5 MHz encoder sampling
	3-4	2.5 MHz encoder sampling
JP19	1-2*	PC able to drive board reset
	2-3	PC not able to drive board reset
JP20	1-2*	32K x 8 program RAM installed
	2-3	128K x 8 program RAM installed

* Default Position

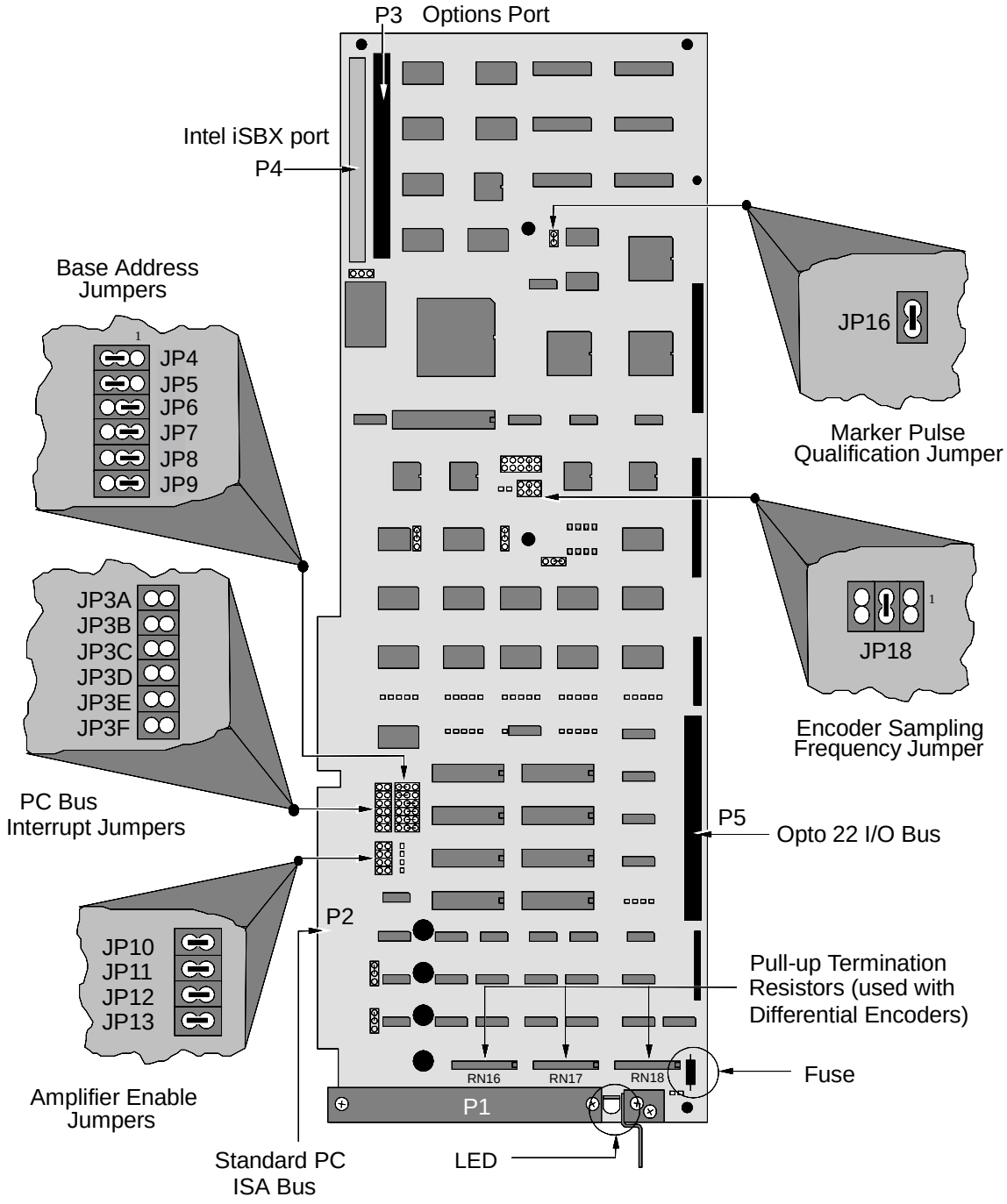
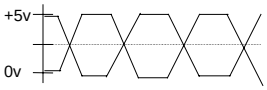


Figure D-1. UNIDEX 500 PC Board Jumper Locations

D.4. Encoder Signal Specifications

Motion control systems are frequently equipped with encoding devices. Signals from these devices are sent to the controller where they are used to sense and control the movement of the positioning equipment. The UNIDEX 500 can be configured for a variety of encoding devices.

D.4.1. Differential Encoders



The UNIDEX 500 accepts differential RS-422 type square wave encoder signals. A "times 4" multiplication is always performed on the encoder fundamental line count. For example, if the encoder line count is 1,000 lines, the effective machine resolution will be 4,000 machine steps (or counts) per revolution.

The marker and quadrature signals use 26LS32 type RS-422 receivers. The sine and cosine signals are pulled to +5 volts through 10K ohm resistors.



D.4.2. Single Ended Encoders

Third party, single ended encoders may be used with the UNIDEX 500 by connecting a 4.7K ohm, 1/4 watt resistor from the unused differential input to signal common as illustrated below in Figure D-2. In this configuration, only a single-ended active high (or active low) signal is provided.

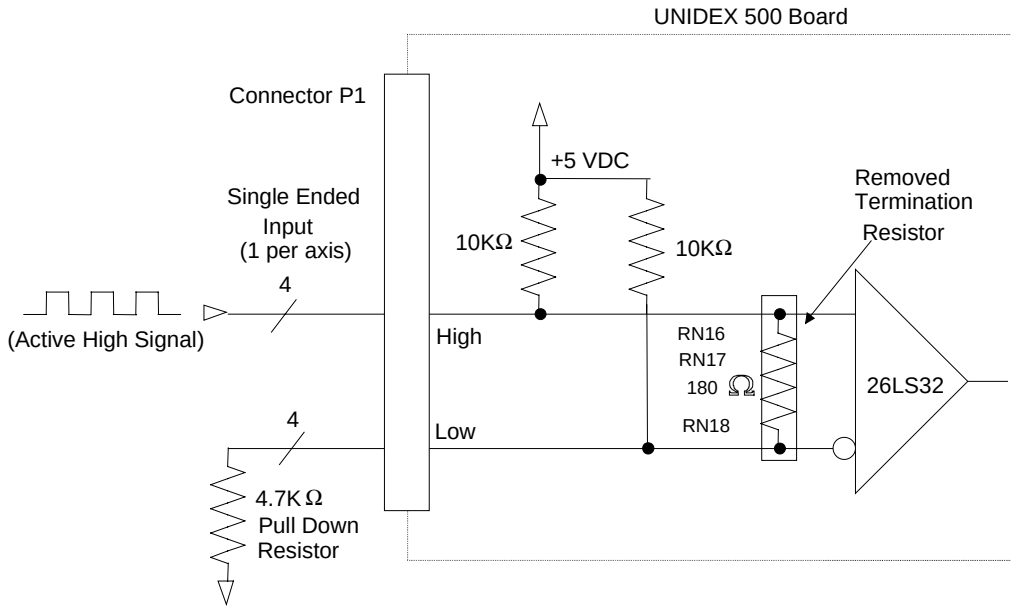


Figure D-2. Electrical Characteristics of a Single Ended Encoder Interface

D.4.3. Software Setup for Single-ended Encoders

Normally, the UNIDEX 500 expects that a differential encoder will be connected to it. Once the single-ended encoder has been connected to the BB500/or BB501 Breakout board or the DR500 drive chassis, a software setting that will have to account for this. In the Edit Parameters window, select the parameter tab called "Faults" and find the parameter "*Error mask fault*". Every axis that has a single-ended encoder will have to have this parameter changed. In the "Faults" parameter tab modify this parameter so that the "Feedback Trap" is turned off (do so by unchecking it inside the tab).

Save the changes, then reinitialize the UNIDEX 500 board for the changes to take effect.



D.4.4. Aerotech Stepper Motors with the Home Marker Option

When using Aerotech stepper motors equipped with the home marker option, it is necessary to replace the appropriate termination resistor(s) with a 0.1 µF capacitor (purchased separately). If fewer than four such stepper motors (with the home marker wheel option) are used, be sure to install the 0.1 µF capacitors as appropriate, while keeping in place the termination resistors for the other axes. See Figure D-3..

The removable termination resistors for axes 1-4 are grouped into three in-line resistor networks (RN16, RN17 and RN18). If your application mixes Aerotech stepper motors (that have the home marker wheel option) with differential or single-ended encoders, you must purchase separate 180 ohm resistors to replace the termination resistors that have been removed as part of the resistor network(s).



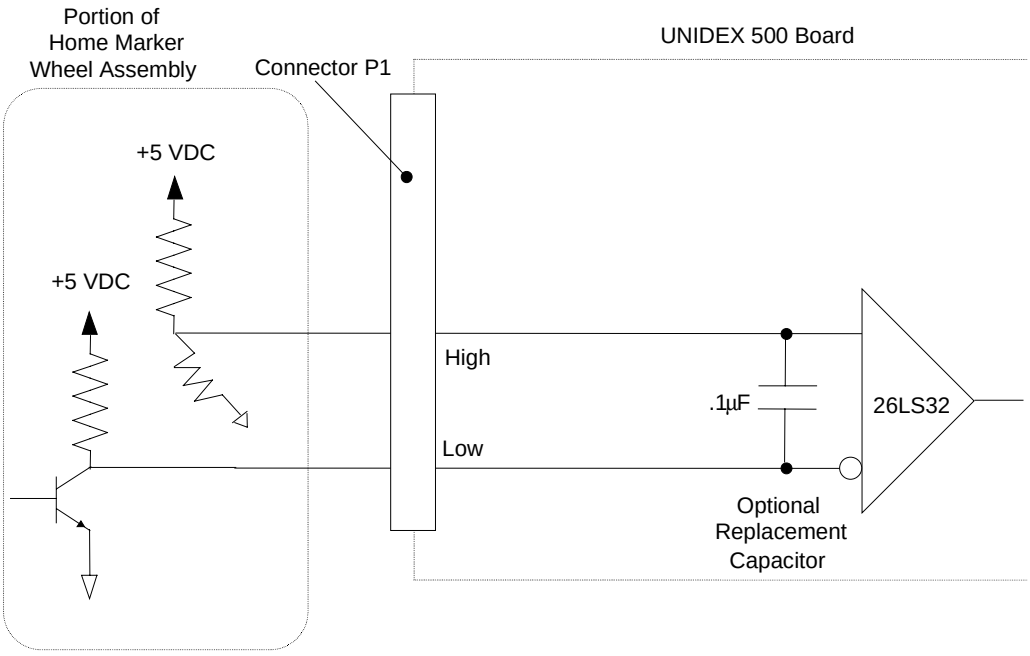
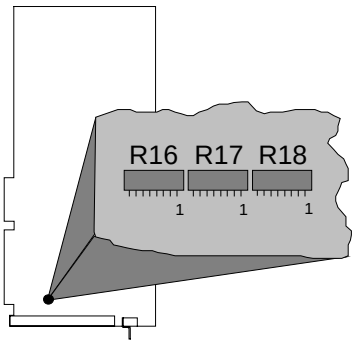


Figure D-3. Electrical Characteristics of an Aerotech Stepper Motor Home Marker Option



Resistor networks RN16, RN17, and RN18 provide termination resistors for axes 1, 2, 3 and 4. The following tables show the important configuration information for each individual axis. Included are the resistor network number, main pinouts (terminal P1), axis signals, and resistor network pin numbers.

Table D-7. Termination Resistor Configuration for Axis 1 Encoders

RN #	Main Pinouts	Axis Signals	RN Pin Numbers	180 Ω Resistor *	0.1 µF Capacitor *
RN16	12	MRK1-	1		
	11	MRK1+	2		
RN17	8	SIN1-	1		
	7	SIN1+	2		
RN17	10	COS1-	5		
	9	COS1+	6		

* Use a 180 Ω termination resistor for standard differential encoders. When using Aerotech stepper motors with the home marker wheel option, you must replace the termination resistor with a 0.1 µF capacitor.

Table D-8. Termination Resistor Configuration for Axis 2 Encoders

RN #	Main Pinouts	Axis Signals	RN Pin Numbers	180 Ω Resistor *	0.1 µF Capacitor *
RN16	20	MRK2-	5		
	19	MRK2+	6		
RN17	16	SIN2-	7		
	15	SIN2+	8		
RN17	18	COS2-	3		
	17	COS2+	4		

* Use a 180 Ω termination resistor for standard differential encoders. When using Aerotech stepper motors with the home marker wheel option, you must replace the termination resistor with a 0.1 µF capacitor.

Table D-9. Termination Resistor Configuration for Axis 3 Encoders

RN #	Main Pinouts	Axis Signals	RN Pin Numbers	180 Ω Resistor *	0.1 µF Capacitor *
RN16	28	MRK3-	7		
	27	MRK3+	8		
RN18	24	SIN3-	1		
	23	SIN3+	2		
RN18	26	COS3-	5		
	25	COS3+	6		

* Use a 180 Ω termination resistor for standard differential encoders. When using Aerotech stepper motors with the home marker wheel option, you must replace the termination resistor with a 0.1 µF capacitor.

Table D-10. Termination Resistor Configuration for Axis 4 Encoders

RN #	Main Pinouts	Axis Signals	RN Pin Numbers	180 Ω Resistor *	0.1 µF Capacitor *
RN16	36	MRK4-	3		
	35	MRK4+	4		
RN18	32	SIN4-	7		
	31	SIN4+	8		
RN18	34	COS4-	3		
	33	COS4+	4		

* Use a 180 Ω termination resistor for standard differential encoders. When using Aerotech stepper motors with the home marker wheel option, you must replace the termination resistor with a 0.1 µF capacitor.

D.5. Encoder Signal Pinouts

Table D-11 identifies the encoder signals and the corresponding P1 connector pin number and termination locations.

Table D-11. Encoder Signals and Pinouts

Signal Name	P1 Pin Number	Termination Location
Channel 1		
Sine, Positive	7	RN-17-2
Sine, Negative	8	RN-17-1
Cosine, Positive	9	RN-17-6
Cosine, Negative	10	RN-17-5
Marker, Positive	11	RN-16-2
Marker, Negative	12	RN-16-1
Channel 2		
Sine, Positive	15	RN-17-8
Sine, Negative	16	RN-17-7
Cosine, Positive	17	RN-17-4
Cosine, Negative	18	RN-17-3
Marker, Positive	19	RN-16-6
Marker, Negative	20	RN-16-5
Channel 3		
Sine, Positive	23	RN-18-2
Sine, Negative	24	RN-18-1
Cosine, Positive	25	RN-18-6
Cosine, Negative	26	RN-18-5
Marker, Positive	27	RN-16-8
Marker, Negative	28	RN-16-7
Channel 4		
Sine, Positive	31	RN-18-8
Sine, Negative	32	RN-18-7
Cosine, Positive	33	RN-18-4
Cosine, Negative	34	RN-18-3
Marker, Positive	35	RN-16-4
Marker, Negative	36	RN-16-3

D.6. Amplifier Enable Outputs

Each axis has one open collector amplifier enable output (refer to Table D-12 and Figure D-4.). The active polarity of this signal is selectable by the configuration of jumpers 10-13. (Refer to Table D-5 and Table D-6 for additional jumper information.)

Each output is a 7404 type open collector driver with absolute maximum ratings of 30 volts and 40 mA sink capability.

Exceeding the amplifier output ratings may cause damage to the UNIDEX 500 control board.



Table D-12. Amplifier Enable Output Locations

Signal	Location
Amplifier Enable 1 (AEN1)	P1 - 69
Amplifier Enable 2 (AEN2)	P1 - 70
Amplifier Enable 3 (AEN3)	P1 - 71
Amplifier Enable 4 (AEN4)	P1 - 72

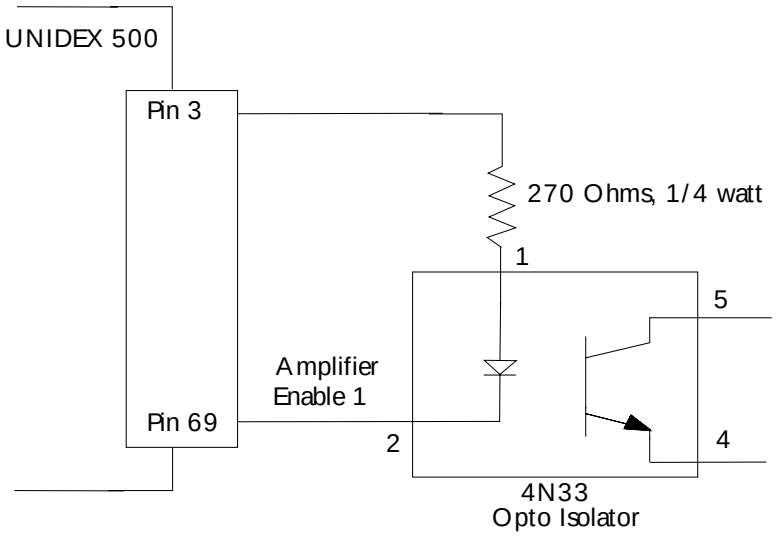


Figure D-4. Electrical Characteristics of the UNIDEX 500 Amplifier Enable Output

D.7. Main Connector Pinout of the UNIDEX 500

The UNIDEX 500's main interface connector (P1) is accessible from the rear of the PC. The connector is a 100-pin "AMPLIMITE" high density female connector. The mating connector is an "AMPLIMITE" connector, part number 759879-9. This connector accepts two 50-pin ribbon cables and is non-shielded.

Refer to Table D-13 for the main connector pinouts.

Table D-13. Main Connector (P1) Pinouts for the UNIDEX 500

Pin	Function	Description	Pin	Function	Description
1	Interlock Send	ILOCKS (GND)	2	20 KHz Synchronization	SYNC+
3	+5 Volts	+5	4	+5 Volts	+5
5	Encoder Ground	GND	6	Encoder Ground	GND
7	Encoder Sine Positive, Axis 1	SIN1+	8	Encoder Sine Ground, Axis 1	SIN1-
9	Encoder Cosine Positive, Axis 1	COS1+	10	Encoder Cosine Ground, Axis 1	COS1-
11	Marker Pulse, Axis 1	MRK1+	12	Marker Pulse, Axis 1	MRK1-
13	Encoder Ground	GND	14	Encoder Ground	GND
15	Encoder Sine Positive, Axis 2	SIN2+	16	Encoder Sine Ground, Axis 2	SIN2-
17	Encoder Cosine Positive, Axis 2	COS2+	18	Encoder Cosine Ground, Axis 2	COS2-
19	Marker Pulse, Axis 2	MRK2+	20	Marker Pulse, Axis 2	MRK2-
21	Encoder Ground	GND	22	Encoder Ground	GND
23	Encoder Sine Positive, Axis 3	SIN3+	24	Encoder Sine Ground, Axis 3	SIN3-
25	Encoder Cosine Positive, Axis 3	COS3+	26	Encoder Cosine Ground, Axis 3	COS3-
27	Marker Pulse, Axis 3	MRK3+	28	Marker Pulse, Axis 3	MRK3-
29	Encoder Ground	GND	30	Encoder Ground	GND
31	Encoder Sine Positive, Axis 4	SIN4+	32	Encoder Sine Ground, Axis 4	SIN4-
33	Encoder Cosine Positive, Axis 4	COS4+	34	Encoder Cosine Ground, Axis 4	COS4-
35	Marker Pulse, Axis 4	MRK4+	36	Marker Pulse, Axis 4	MRK4-
37	Encoder Ground	GND	38	Encoder Ground	GND
39	Clockwise Limit, Axis 1	CW1	40	Counter clockwise Limit, Axis 1	CCW1
41	Clockwise Limit, Axis 2	CW2	42	Counter clockwise Limit, Axis 2	CCW2
43	Clockwise Limit, Axis 3	CW3	44	Counter clockwise Limit, Axis 3	CCW3
45	Clockwise Limit, Axis 4	CW4	46	Counter clockwise Limit, Axis 4	CCW4
47	Home Limit, Axis 1	HOME1	48	Home Limit, Axis 2	HOME2
49	Home Limit, Axis 3	HOME3	50	Home Limit, Axis 4	HOME4
51	Limits Ground	GND	52	Limits Ground	GND
53	+12 Volts	+12	54	+12 Volts	+12
55	-12 Volts	-12	56	-12 Volts	-12
57	Recirc. Mode Axis 1 (*Reserved*)	MODE1	58	Recirc. Mode Axis 2 (*Reserved*)	MODE2
59	Input 0	IN0	60	Input 1	IN1
61	Input 2	IN2	62	Input 3	IN3
63	Output 0	OUT0	64	Output 1	OUT1
65	Output 2	OUT2	66	Output 3	OUT3
67	Recirc. Mode Axis 3 (*Reserved*)	MODE3	68	Recirc. Mode Axis 4 (*Reserved*)	MODE4
69	Amplifier Enable 1	AEN1	70	Amplifier Enable 2	AEN2
71	Amplifier Enable 3	AEN3	72	Amplifier Enable 4	AEN4
73	Amplifier Fault 1	AFLT1	74	Amplifier Fault 2	AFLT2
75	Amplifier Fault 3	AFLT3	76	Amplifier Fault 4	AFLT4
77	Limits Ground	GND	78	Limits Ground	GND
79	Axis 1 Primary Current Command	ICMD1B	80	Axis 1 Secondary Current Command	ICMD1A

Table D-13. Main Connector (P1) Pinouts for the UNIDEX 500 (continued)

Pin	Function	Description	Pin	Function	Description
81	Axis 2 Primary Current Command	ICMD2B	82	Axis 2 Secondary Current Command	ICMD2A
83	Axis 3 Primary Current Command	ICMD3B	84	Axis 3 Secondary Current Command	ICMD3A
85	Axis 4 Primary Current Command	ICMD4B	86	Axis 4 Secondary Current Command	ICMD4A
87	Ground	GND	88	Ground	GND
89	Joystick Potentiometer 1 Input	JSW1	90	Joystick Potentiometer 2 Input	JSW2
91	Joystick Button A Input	JSA	92	Joystick Button B Input	JSB
93	Joystick Button C Input	JSC (JS ILOCK)	94	Brake Output	BRAKE
95	Analog Input 0	AIN0	96	Analog Input 1	AIN1
97	Emergency Stop	ESTOP	98	User Interrupt	UINT
99	Opto Isolator Anode (for 97 &98)	OPTOA	100	Interlock Receive	ILOCKR

∇ ∇ ∇

APPENDIX E: SETTING UP AN AC BRUSHLESS MOTOR WITH THE UNIDEX 500

Appendix E

In This Section:

- Introduction E-1
- Setup Procedure E-1

E.1. Introduction

AC brushless motor commutation differs from that of a DC brush motor. The servo loop output for a DC brush motor is a signal between -10 and +10 volts. This signal is connected to an amplifier that converts the voltage into motor current. The current produces torque in the motor. The servo loop output is sometimes called a current command even though it is a voltage.

The DC brush motor has a pair of fixed magnets mounted in the motor's shell and a rotor which has a series of windings called the armature. When current passes through the armature windings, it produces a magnetic field that interacts with the fixed magnets to produce a force. In order to maximize this force, the fields should be separated by 90 degrees. In a DC brush motor, the brushes and commutator do this. The brushes carry the current to the commutator and, as the motor rotates, a new set of armature windings is energized. This keeps the armature magnetic field perpendicular to the magnets.

AC brushless motor construction is the reverse of a DC brush motor–the permanent magnets are on the rotor and the windings are mounted to the motor housing. Because the windings are stationary, there is no need for brushes. However, the current in the windings must change so that the generated magnetic field is perpendicular to the field produced by the rotor magnets. A feedback device is used to keep track of the rotor position.

E.2. Setup Procedure

The UNIDEX 500 generates two phases of current commands separated by 120 degrees. The servo loop output is multiplied by these phases and sent to the amplifier. A third phase may be generated by adding the first two and inverting its polarity. Aerotech amplifiers do this automatically on the amplifier. The phases are as follows:

Phase A = sin(ang)

Phase B = sin(ang + 120)

Phase C = - (Phase A + Phase B)

The UNIDEX 500 has two test functions for setting up AC brushless motors–"MSET" and "MCOMM." The MSET function sends a "two phase" vector at a specified angle and voltage. The MCOMM function sends a (commutated) torque command at a specified voltage level. The syntax of these commands is as follows:

MSET *axis,voltage,angle*

where:

axis = axis number (1-4)

voltage = amplitude of vector output (0-10 V)

angle = angle of output vector (0-359 °)

MCOMM *axis,voltage*

where:

axis = axis number (1-4)

voltage = peak magnitude of torque output.

To use these commands in setting up an AC brushless motor, the following procedure is recommended.

1. Disconnect motor from load.
2. Connect motor leads A, B, and C to amplifier A, B, and C, respectively.
3. Set axis parameter x42, *Drive (0-DC Brush, 1-AC Brushless, 2/3 Stepper)*, to 1 for AC brushless servo.
4. Set servo loop gains Kpos, Ki, and Kp to 0.
5. Defeat position error and integral error faults.
6. Enable the axis.
7. Check motor direction by sending a series of MSET commands with increasing angles. See the example program below. As the angle increases, the motor should move in the positive or CW direction. If it does not, reverse two of the motor leads.

   Example Program

   This program increments the vector by 10 degrees each time through the loop. It is written for the X axis. To change to a different axis, change line #1 and V0. The motor should not be connected to the load during this test.

```
ENABLE X          ; enable axis to check (X, Y, Z, or U)
V0=1              ; axis number (1,2,3, or 4)
V1=.5             ; peak amplifier output voltage
V2=0              ; starting angle (0 degrees)
V3=1000           ; time in milliseconds between vectors (step speed)
WAIT ON           ; wait for previous commands to finish
ABORT             ; abort internal queue buffer
LOOP 36           ; loop for one electrical cycle
MSET V0,V1,V2     ; send vector
DWELL V3          ; delay between vectors
V2=V2+10          ; angle of next vector
NEXT
QUEUE AGAIN       ; run program continuously
```

The feedback device's position display should also be increasing (See diagnostic window in Figure E-1).



See Engineering Specification for definition of motor direction.



8. Check the Hall signals. The Hall signals can be viewed in the U500 diagnostic window. See Figure E-1. As the motor is moved in the positive direction, the Hall signals should cycle through the proper states. Swap Hall signals until the proper sequence is obtained. See Table E-1. Aerotech linear and rotary AC brushless motors require Hall effect feedback except for rotary AC brushless motors with resolvers. Resolvers provide absolute position information for one revolution of the motor.

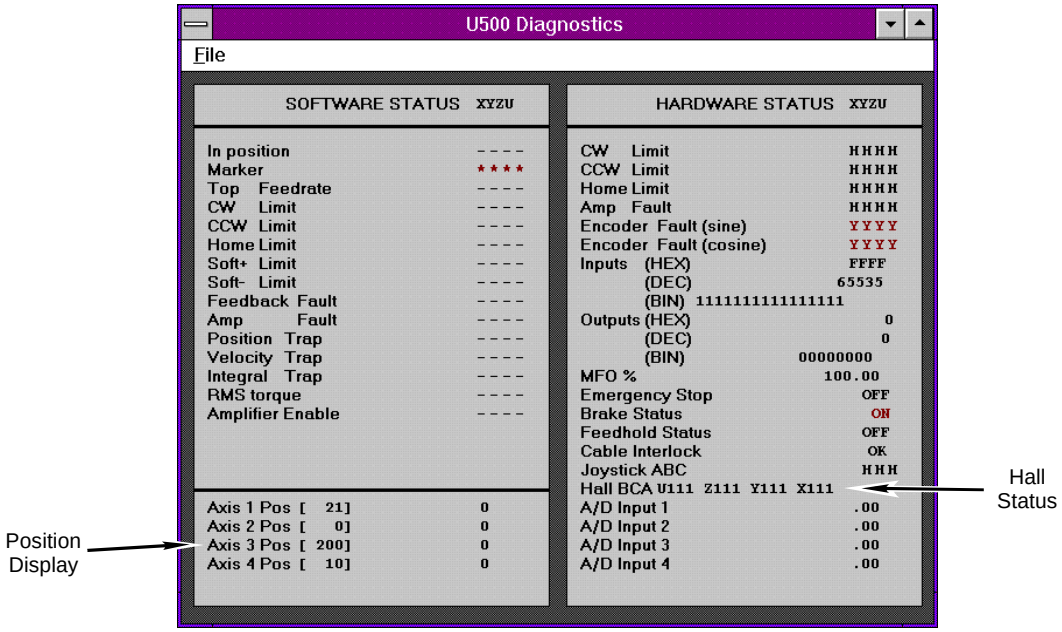


Figure E-1. U500 Diagnostic Window

Table E-1.Hall State Table

MSET angle (degrees)	HC(msb)	HA	HB (lsb)
330-30	1	0	0
30-90	1	0	1
90-150	0	0	1
150-210	0	1	1
210-270	0	1	0
270-330	1	1	0

9. Set commutation offset. Record the Hall states using the MSET command every 10 degrees. Compare with the desired Hall states. Calculate the needed shift in degrees to make the recorded Hall states align with the desired Hall states. Enter this number (in degrees) for axis parameter x45, *Phase offset (degrees)*. Re-check Hall signals.

10. Verify motor commutation:

    a. Set axis parameter x43, *AC brushless motor commutation factor (cycles/rev)*, to 0. This will force the controller to commutate from the Hall signals only (six step mode).

    b. Linear motor:
    Send the MCOMM command and verify the motor output force. Since the motor will move open loop, make sure the positioning stage is secured before sending the command. The motor force output should be smooth and in the same direction over the length of travel. Motor force can be measured with a spring scale. Test the other direction by specifying a negative voltage in the MCOMM command.

    c. Rotary motor:
    If the motor is disconnected from the load, the open loop speed of the motor can be checked using the MCOMM command. A properly phased motor will go the same speed in each direction. Use the axis tuning scope to observe the velocity.

    d. Set axis parameter x43 back to its original value. It should be 1 for linear motors and 2, 3, or 4 for rotary motors. Also, make sure that axis parameter x44, *Encoder feedback (steps/rev/(*4))*, is set properly. For linear motors, enter the number of steps per electrical cycle. UNIDEX 500 will now commutate in sinusoidal mode. The above commutation checks can be repeated if desired. The motor torque output should feel "smoother" in sinusoidal mode than in six step mode. If the motor is commutating properly in six step mode but not sinusoidal mode, check axis parameter x44. Also verify encoder operation. The encoder should count in the positive direction when a positive torque is commanded.

11. Re-enable axis position and integral faults in the fault mask and tune servo loop.

See related Aerotech Engineering Specification ES12731-n for signal definitions, color coding, and motor revision information.



∇ ∇ ∇

APPENDIX F: UNIDEX 500 APPLICATIONS

Appendix F

In This Section:

- Using the Encoder Position Capture Input - "UINT_N" F-1
- Additional UINT features on the U500 PCI card. F-2

F.1. Using the Encoder Position Capture Input - "UINT_N"

The "UINT_N" input to the UNIDEX 500 can be used to capture encoder positions. It is an opto-isolated input that generates an interrupt when an external switch completes the circuit to ground through the +5 V OPTOA power supply. This interrupt is negative-edge triggered. The maximum interrupt latency is approximately 6 µsec for Rev_ boards and 3 µsec for RevC boards. This function can only be used to capture the on-board encoder channels (Position feedback channels 1-4).

When an interrupt occurs, all four axes' positions are sampled and stored in internal memory. These positions are in machine steps and are referenced to hardware home. The U500 will generate a PC bus interrupt after UINT_N is processed. Set JP10 through 15 to select which PC interrupt to generate.

The UNIDEX 500 can store 4096×4 axis positions. Each position is stored as a 48 bit quantity internally. UNIDEX 500 memory location X:$1B00 contains the current write pointer to memory. The buffer area is from L:$4000 to L:$7FFF. Once this area is filled, the write pointer will reset to the beginning of the circular buffer. Each time an interrupt occurs, the write pointer will increment by 4. The write pointer always points to the next available memory location. The DOS utility program "DSPDUMP" can be used to examine these memory locations.

The interrupt is disabled after reset by default. The interrupt is enabled by ORing "X" memory location $ffff with "0x03" using the aer_DSPMemWrite function (See the Programming Tools chapter). The interrupt status can be polled by checking bit #22 of aer_read_status(5). JP10 through 15 do not need to be installed if this method is used. The interrupt request is cleared by calling "aer_reset_intr()".

The following table lists the locations of the relevant signals.

Table F-1. Pin Locations of UINT_N Signals

Signal	Description	U500	DR500	BB500	BB501
UINT_N	User Interrupt	P1, pin 98	J13, pin 2	P5, pin 15	TB1, term 7
OPTOA	Opto Power +5 V	P1, pin 99	J13, pin 4	P5, pin 16	TB1, term 8

The position capture information is stored in the same memory block as the axis scope / tuning. The axis scope in the MMI or TOOLKIT software will overwrite any position capture data stored in this area.





This function will only work with U500 firmware version 4.10 and above and requires a PLUS or an ULTRA card. The U500 firmware file is named "U500.JWP" for Rev_ boards and "U500C.JWP" for RevC boards.

The UINT SETUP CODE parameter (#500) defines the operation of the UINT_N input (See the Parameters chapter).



If UINT_N is always enabled, the user must make sure that the interrupt signal does not bounce.

The following "C" functions can be used to access individual memory locations in the U500's memory (see Programming Tools chapter).

aer_DSPMemRead(short *dspbase***, short****reterr***, long** *dspmem***, long** *mtype***, long ****dspmsw***)**	Reads a memory location from the DSP's memory
aer_DSPMemWrite(short *dspbase***, short ****reterr***, long** *dspmem***, long** *mtype***, long** *data***, long** *mode***)**	Modifies a location in the DSP's memory
aer_reset_intr(void)	Used to reset the interrupt lines on the ISA bus

See "ENCAPT.C" on the installation disks for more information.

F.2. Additional UINT features on the U500 PCI card

The UNIDEX 500 PCI card has the ability to scan for a position capture occurrence at the servo loop update rate. This should be used when it is known that the position capture will not be generated faster than 4Khz. This method is preferred since the servo loop will not be interrupted. This eliminates any potential for servo loop beating with the UINT input. Set general parameter #500 (User interrupt setup code) bit #3 to 1 (8) to scan for a position capture occurrence at the servo loop update rate.

The opto coupler requires a 10 ma forward bias current. When used with a 5V signal, no external resistor is required. The positive supply voltage should be connected to the OPTOA (opto-isolator anode) pin. The diode is forward biased by completing the circuit through the ESTOP pin back to the power supply. Note that the OPTOA pin is shared with the ESTOP opto isolator.

The UINT signal on the PCI card can also be configured as a non-isolated TTL level input (see chart below).

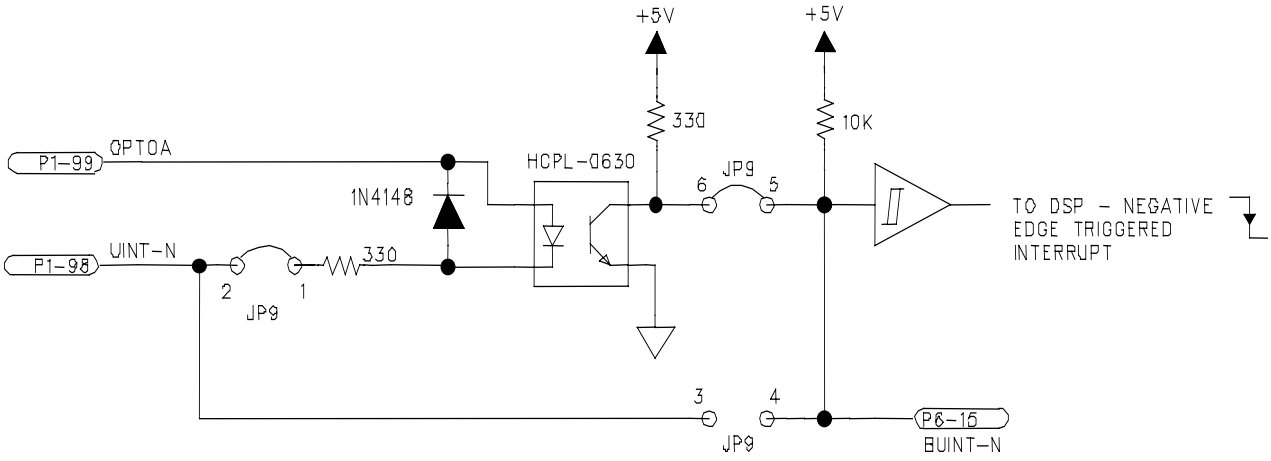


Figure F-1. U500PCI UNIT-N Circuit

Jumper Configuration:

PCI UINT opto-coupled	JP9 1-2 IN, 5-6 IN, 3-4 OUT (DEFAULT)
PCI UINT from main connector or P6 (TTL format)	JP9 3-4 IN, 1-2 OUT, 5-6 OUT

∇ ∇ ∇

APPENDIX G: U500 CALIBRATION OPTION

In This Section:
- Introduction G-1
- Axis Calibration.....G-2
- Orthogonality Correction.....G-5
- 2D Error Mapping G-8

G.1. Introduction

There are three types of corrections available with the U500. Axis calibration is used to correct for linear errors in the ball screw or linear encoder. Orthogonality correction is used to correct angular errors in a pair of axes. 2D error mapping corrects up to 3 axis positions based on an X-Y grid of points.

- All calibration functions require the U500 Plus or Ultra models.
- Correction data entered is “absolute” and is in units of machine steps. All calibration is referenced to the hardware home position.
- The U500 Diagnostic window shows the axis encoder position with the correction data added. This is useful for verifying / debugging operation.
- The MMI feedback display does not show the effects of the axis calibration, ortho correction, or 2D error mapping.

Multiple correction tables can be applied within a single ASCII calibration (.CAL) file. It is conceivable that a single calibration file might contain certain combinations of:

- up to four axis calibration sections
- up to four orthogonality correction sections
- up to two 2D error mapping sections

This calibration file should be specified in U500 project file when using the Toolkit or MMI software.

G.2. Axis Calibration

Axis calibration is used to correct errors inherent in the ball screw or linear encoder. These errors are sometimes referred to as “lead” errors. The errors used to generate the correction table are mapped with a precision reference source such as a laser interferometer system.

Each axis can contain a calibration table of up to 2048 correction points. The calibration block is delimited by the keywords “:START” and “:END”. The first number in the calibration section is the axis number 1-4. The second number is the encoder distance between corrections. The remaining data is the actual correction data. The first correction data point corresponds to the first sample distance from home. The home position is an implied “0” correction point and does not have a table entry. The U500 linearly interpolates between correction data points and calculates a new correction for each axis every 8 milliseconds.

The correction data represents the number of absolute machine steps needed to correct the current position. Feedback polarity is always increasing for CW rotation and decreasing for CCW rotation. Correction numbers should be entered accordingly, regardless of home direction and programmed polarity. Correction can be positive or negative.

Axis calibration will be active when:

- the axis calibration parameter (x15) is set to “yes”
- the ASCII calibration file is present in the U500 project files and U500 has been initialized
- the axis has been homed

These points are loaded to the UNIDEX 500 during initialization from a calibration (.CAL) file. A calibration file can also be dynamically loaded from the users application by calling the “**CAL *filename***” function.

Axis calibration is repeating. A move outside the calibration window will be mapped back. If this operation is undesirable, the end of the calibration file should be "padded" with the last error. In the case of circular calibration, the calibration table should equal one revolution.

A sample of the ASCII file format for the calibration file is shown in Figure G-1.

```
; ***********Comments ************************************
;  The :START and :END statements surround the calibration information.
;  First non-comment line is axis number (1-4).  Second non-comment line
;  is the sample distance (in machine steps) that the axis must travel before
;  the next sequential correction datum is added to the current position to
;  correct it.  Correction data is space-separated and may be entered on
;  multiple lines.
;*****************  SAMPLE.CAL  ***************************
:START                                  ;start axis call block
1                                       ;axis number  (1,2,3, or 4)
1000                                    ;sample distance in machine steps (>256)
1 1 1 2 4 2 5 10 8 8 7 –6 –6 –2 –1 –1   ;absolute machine step correction data
:END                                    ;end axis 1 cal block
:START                                  ;next axis to be calibrated
2                                       ;axis number (1, 2, 3, or 4)
1000                                    ;sample distance in machine steps
1 1 1 2 4 2 5 10 8 8 7 6 6 2 1 1        ;absolute machine step correction data
:END                                    ;end axis 2 cal block
```

Figure G-1. Sample ASCII Calibration File

Sample calibration data is listed in Table G-1.

Table G-1. Sample Calibration Data

Commanded Distance (machine steps)	Actual Distance (displayed in diagnotsic window)	Correction Data (entered in calibration file)
1000	1001	+1
2000	2001	+1
3000	3001	+1
4000	4002	+2
5000	5004	+4
6000	6002	+2
7000	7005	+5
8000	8010	+10
9000	9008	+8
10,000	10,008	+8
11,000	11,007	+7
12,000	11,994	-6
13,000	12,994	-6
14,000	13,998	-2
15,000	14,999	-1
16,000	15,999	-1

G.3. Orthogonality Correction

Orthogonality correction can be used to correct the straightness of a given axis. This is done by moving a separate (usually perpendicular) axis to correct the path of the first axis as it moves. The correction table values are usually generated with a precision reference such as a laser interferometer system.

Orthogonality correction can be used to generate a simple angular correction by placing several correction points in the table using the following formula,

error(n) = 4.85E-6 * angleerror * sampledistance * n

where:

error(n): correction value to enter into table at location “n”

n : calibration point number 1,2,3…

angleerror: angular error between axes in arc seconds

sampledistance : arbitrary calculation distance (should be greater than 256)

The correction does not have to be a simple angular correction however since a table is used. This can be used to correct arbitrary errors in the mechanical system.

Orthogonality correction will be active when:

- The orthogonality correction parameter (x71) is set to “yes”
- The ASCII calibration file is present in the U500 project files and U500 has been initialized
- The position dependent axis has been homed (the axis being corrected does not have to complete a home cycle)

A maximum of 1024 data points of correction for each axis is available. Refer to Figure G-2.

```
;*****************************************************************
;  Axis orthogonality correction data can also be entered in the .CAL table.
;  The axis number must be a two digit number AB, where A represents the
;  axis number to be corrected, and B represents the position dependent axis
;  number.  Correction does not begin until the B axis is homed.  The A axis
;  does not need to be homed.  A maximum of 1024 points are allowed for
;  orthogonality correction.  The sample distance must be greater than
;  256 encoder counts.   An example follows.
;*****************************************************************
:START                              ;start axis calibration block
21                                  ;ortho correction of axis 2 based on
                                        ;position of axis 1
1000                                ;sample distance in machine steps
                                        ;(>256)
1 2 3 5 7 9 12 14 16 14 10 8 5 2 1 0    ; absolute machine step corr. data
:END                                ;end axis call block
```

Figure G-2. Sample Single-Row Orthogonality Correction File

Multi row orthogonality correction data can also be entered. This is similar to the single row correction format described above but allows multiple straightness corrections to be made depending on the location of the axis being corrected. Linear interpolation is performed between elements in a given row but not between adjacent rows.

```
:ORTHO                              ; start of multi row ortho correction
                                        ; block
21                                  ; ortho correction of axis 2 based on
                                        ; position of axis 1 (axis 2 gets
                                         ; corrected based on axis 1's position)
200000                              ; sample distance of axis 1 (between
                                        ; row elements)
1250000                             ; sample distance of axis 2 (between
                                        ; rows)
10                                  ; number of points per row
1 2 3 5 7 9 12 14 16 14             ; row 1 data
1 -2 -3 -5 -3 1 5 7 5 2             ; row 2 data
( … )
:END                                ;end axis call block
```

Figure G-3. Sample Multi-Row Orthogonality Correction File

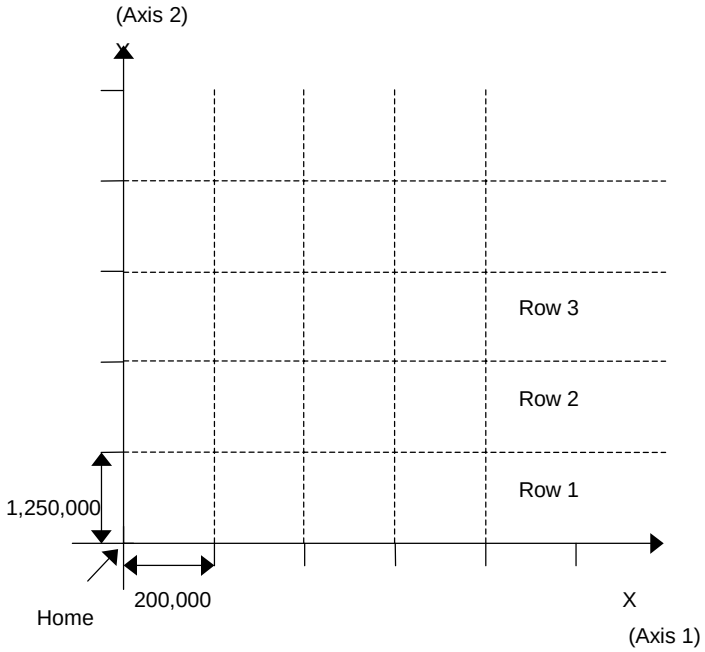


Figure G-4. Multi-Row Orthogonality Correction Grid

G.4. 2D Error Mapping

In this mode, calibration for a pair of axes is based on a two-dimensional grid of correction points. A total of 4096 correction values may be entered into the table. This represents 1365 (x,y,z) correction pairs. The correction table is usually generated with a reference plate and vision system

2D error mapping will be active when:

- The orthogonality correction parameter (x72) is set to "yes"
- The ASCII calibration file is present in the U500 project files and U500 has been initialized
- Both axes have been homed

Each correction point contains an absolute correction value in machine steps for 3 axes. The polarity of this correction is always the same as the encoder polarity as observed in the diagnostic window. See Figure G-5 through Figure G-7.

```
:MULTI                              ; beginning of table
1 2                                 ; row, column input axis numbers
1 2 3                               ; output (corrected) axis numbers
10000 10000                         ; row, column sample distance for input axes
                                        ; (machine steps)
0 0                                 ; row, column offset to table start
                                        ; (signed, machine steps)
5                                   ; number of points for the row axis
1 1 0   2 2 0   3 3 0   4 4 0   5 5 0   ; corrections for output axes in groups of three
2 1 –1   3 -3 2   3 2 1   2 2 1   0 0 0   ; [a_n b_n c_n]  [a_n+1 b_n+1 c_n+1] …
…
:END                                ; end of this block of data
```

Figure G-5. Calibration File with Correction Point Data

1). Row, column input axis numbers — These are the axis numbers that are used to index through the correction table. Allowed values are 1-4. The first number is the 'row' axis, the second is the 'column' axis.

2). Output axis numbers — These are the axes which will have the correction applied to them. Allowed values are 1-4. Correction values are generated for the specified axis if axis parameter x72 is set to yes, the row and column axes have been homed, and a valid calibration file has been selected in the project file.

3). Row, column sample distance — This is the sample distance in machine steps for the row and column axes respectively. These numbers are not necessarily the same but must be positive. They must also be greater than 4097. The first number (row sample distance) is the distance between a_n and $a_{n+1.}$ The second number (column sample distance) is the distance between b_n and b_{n+1}.

4). Row, column offset

This is the offset from the axis home positions to the start of the correction table in machine steps. The first number is the row axis offset, the second is the column axis offset. These numbers are signed and their magnitude must be less than 2,147,483,647. The polarity of these numbers is the same as the polarity displayed in the diagnostic window. These numbers can be used to calibrate a window that can be arbitrarily placed within the stages travel.

5). Number of points per row

This tells the software how many rows are in the correction data and, therefore, how many columns.

6). Correction data

The correction data is organized in groups of three corresponding to the three output axes specified in 2). In the example above, a_n is axis 1, b_n is axis 2, and c_n is axis 3. Correction data is absolute and in the same polarity as the feedback device (as displayed in the diagnostic window). The correction table must contain an even multiple of 3 points. The total number of points must also be evenly divisible by the number of points per row (x3). If either of these conditions are violated, the U500 will respond with a "Calibration file format error" message. All text following a ';' , to the end of the line, is a comment and is ignored.

7). Home direction

Usually, if axis parameter x02 is set to no, the stage travel is in the negative machine direction, therefore, all feedback as displayed in the diagnostic window is negative. In this case, the first correction point in the table will correspond to the home position. The next point in the table corresponds to one sample distance from the home position, etc.

8). Home position in center of travel

In this case, the row or column offsets should be set to a negative number (assuming that axis parameter x02 is set to yes). This will offset the table making the home position within the table.

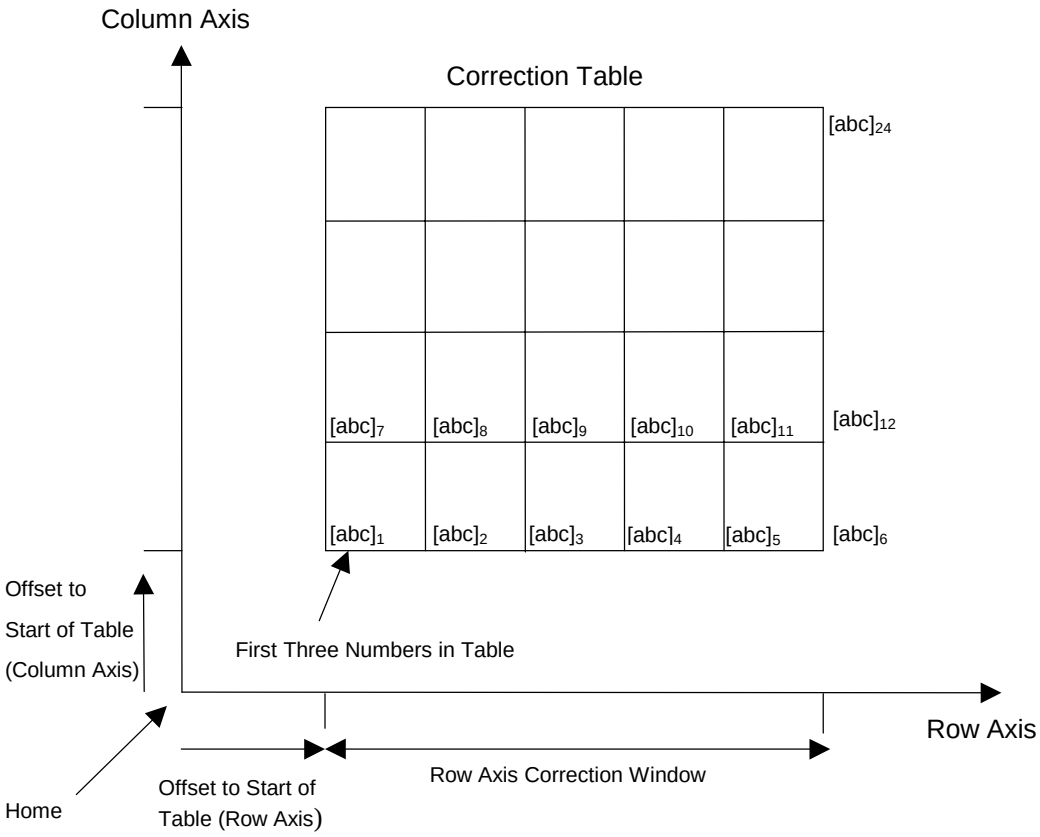


Figure G-6. 2D Correction Grid

Notes:

1. Table width is 5 x sample distance but there are 6 points per row
2. Table height is 4 x sample distance but there are 5 points along the column axis
3. When the axis is outside the correction table area, the correction added to the axis will be the last correction added or 0 if the axis has not entered the table.
4. Errors are linearly interpolated between points as follows…

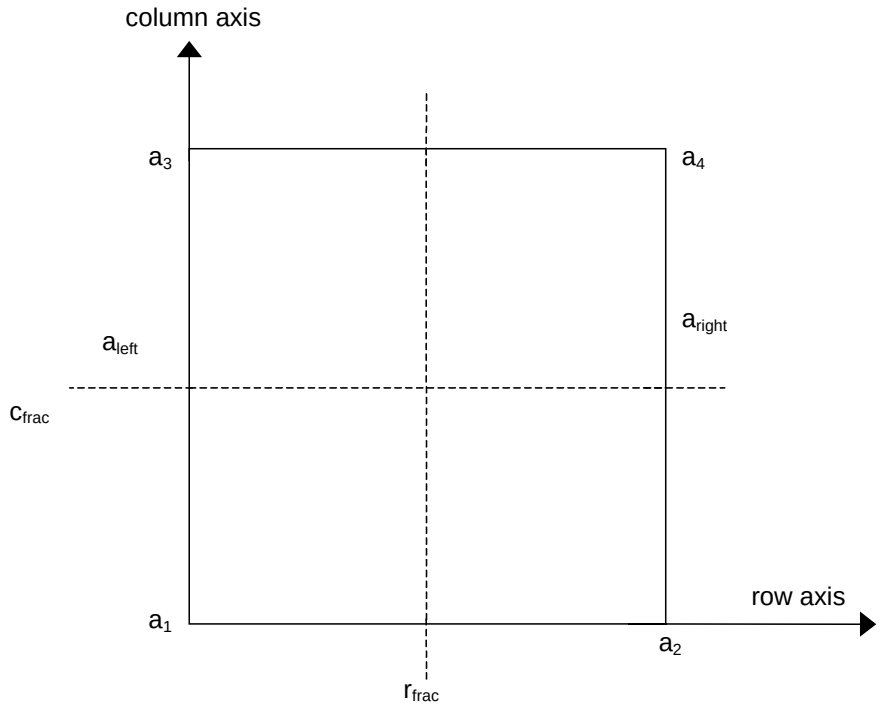


Figure G-7. 2D Correction Interpolation

Interpolation:

1. a_1,a_2,a_3,a_4 are row axis corrections about a square formed by adjacent points within the correction table.
2. c_{frac} is the fraction distance that the column axis is in between correction points (0<1).
3. r_{frac} is the fraction distance that the row axis is in between correction points (0<1).

Row axis:

1. a_1 to a_3 is interpolated based on c_{frac} to form a left side row axis correction a_{left}, a_2 to a_4 is interpolated based on c_{frac} to form a right side row axis correction a_{right}.
2. a_{left} and a_{right} are interpolated by r_{frac} to get the desired correction for the row axis.

Column / third axis :

1. Opposite of Row axis.

∇ ∇ ∇

SYMBOLS

$INn Input Command, 7-10, 9-13
$INP Input Command, 7-10
$UAP Register, 7-6
$URP Register, 7-6
$XAP Register, 7-6
$XRP Register, 7-6
$YAP Register, 7-6
$YRP Register, 7-6
$ZAP Register, 7-6
$ZRP Register, 7-6
( ) Operators, 7-5
\UTILITY Subdirectory, 11-1
^ Operator, 7-5
+ Operator, 7-5
< Operator, 7-76
<= Operator, 7-76
<> Operator, 7-76
= Operator, 7-5, 7-76
> Operator, 7-76
>= Operator, 7-76
7404-type Open Collector Driver, D-11
7440-type Axis Driver, 12-25

A

A/D channel registers, 7-8
A/D Channel Test Points, 12-22, D-3
Abort button, 4-38
ABORT Command, 7-22
Absolute Motion, 9-5
Absolute Position Registers, 7-6
Absolute Positioning, 1-2
AC brushless motor setup, E-1
ACCELERATION Command, 7-22
Acceleration feedforward gain - Aff, 6-21
Accessories
  BB500, 1-5
  DIO500 option, 1-5
  handwheel, 1-5
  JBV, 1-5
  OP500, 1-5
  Opto-22 interface board, 1-5
  PSO-PC, 1-5
  RDP-PC, 1-5
  RMX-PC, 12-24
  SHUNT500, 1-5
  vertical axis position brake, 1-5
Active Polarity Jumpers for Axes, 12-25
AD0 Register, 7-8, 7-9
AD1 Register, 7-8, 7-9
AD2 Register, 7-8, 7-9
AD3 Register, 7-8, 7-9
Addition Function, 7-5
Advanced motion tab, 5-13
Aff Acceleration feedforward gain, 6-3, 6-21
Aff Acceleration feedforward loop, 6-17
AGAIN Command, 7-27, 9-7
Amplifier Chassis (DR500) Part Numbers, 1-4
Amplifier Enable Jumpers, D-4
  default, 12-25
Amplifier Enable Output Locations, D-11
Amplifier Enable Outputs, D-11
Amplifier Fault, 7-64, 12-7
Amplifier Outputs, 12-25
  impedance states based on axis state, 12-25
Amplifiers
  7404-type open collector drivers, D-11
  state of, with respect to LED, 3-8
AND Function
  use with marker pulse qualification, 12-24
Antistatic Bag, 2-2
Antistatic Precautions, 3-4
ASCPRM.EXE Utility, 11-1
ASINE Function, 7-4
Assignment Function, 7-5
ATN Function, 7-4
AUTOEXEC.BAT File
  customizing, 3-1, 3-2
Autotuning, 6-4
AUX OUTPUT, 7-64
Axes
  accel/decel rates, 7-22
  active polarity jumpers, 12-25
  command output test points, 12-21, D-2
  disabling, 7-58
  enable/disable state at reset or initialization, 12-25
  enabling, 7-63
  marker pulse signals, 12-21, D-1
  ramping, 7-110
Axis, 5-19
  angular position, measuring, 5-7
Axis Calibration, 1-3
Axis display options, 4-47
Axis Movement
  during a reset, 5-6
Axis number radio buttons, 5-4
Axis Parameters, C-1, C-5
Axis position diagnostics, 4-17
Axis Position Display, 3-12
  clearing, 5-6
  inaccuracies after a reset, 5-6
  number of decimal digits, 5-11, 5-12
  using rotary (degree) units, 5-7
Axis scope screen, 4-20
Axis Scope toolbars, 6-2
Axis Scope window, 6-9, 6-22
Axis submenu - Axis scope screen, 4-24
Axis tab, 5-2

B

BACKUP Subdirectory, 3-2
Balance potentiometer, 6-29
Base Address
  test point, 12-22, D-2
Base Address Jumpers
  default, 3-4, 12-31
Base Addresses
  default setting, 3-4
Basic motion tab, 5-33
BOARD Command, 7-31
Board Jumpers, D-5
Board Level Execution, 7-2
Board Types, 7-2
Boolean Operators in PLC, 7-103
BRAKE Command, 7-32
Braking, 7-32
  output signal test point, 12-22, D-3
Breakout Module, 1-5
BRK/BPS Vertical Axis Brake Option, 1-5

C

Cable Interlock Send Test Point, 12-22, D-3
Cables
  U500 to chassis, 1-5
CAL File, 3-2, 5-100
Calibration, 1-3
Calibration File, 3-2, 5-100
CALLDLL Command, 7-34
Cam Tables, 9-12
Capacitors
  used to replace termination resistors, 12-3, D-7
CCW Rotation Limit Switch, 12-7
CCW_CIRCLE Command, 9-3, 9-6, 9-8
Cell descriptions of the status bar, 4-44
CFG File, 3-2
Checksum Failure, 13-2
Circular Interpolation, 1-3, 9-7
CLOCKWISE Circular Rotation Command, 7-37
Closed loop operation, 5-66
CLRSCR (Clear Screen) Command, 7-42
CM (Contouring Mode) Command, 7-140
CMOS (PC's), Reconfiguring, 5-8
CNC Example Program, 9-7, 9-13
Collect submenu - Axis scope screen, 4-23
COM Port
  data bits, 7-45
  initializing, 7-45
  stop bits, 7-45
COM Ports
  baud rate, 7-45
  parity, 7-45
Commands, 7-15
Comments in Programs, 7-82
Communications Failure, 13-2
Communications Problems, 13-2
Commutation, 7-88, E-1
Commutation Factors, 5-64, 5-78
Commutation Factors for 4, 6 and 8 Poles, 5-68
Comparison Operators, 7-76
Concept of programming steps, 7-8
Conditional Statement IF, 7-75
CONFIG.SYS File
  customizing, 3-1, 3-2
Configuration
  amplifiers, 12-25
  flowchart summary of installation process, 3-14
  servo loop setup, 3-15
Configuring Key Parameters, 3-9
Constant Velocity Motions, 1-2
Contour Planes
  stopping all motion, 7-73
  Using English or Metric units, 5-10
Contour Planes, Interlocking, 9-11
Control option, 4-27
Conversion Factor, 5-95
  machine steps, 5-95
  machine steps per unit, 5-95
  program steps, 5-95
  program units, 5-95
Conversion Factor Parameters
  examples using, 5-95
Corner Rounding, 7-113, 9-8, 9-9
COS Function, 7-4
Current limit potentiometer, 6-29
Cursor option, 4-26
Custom Software
  PC bus interrupt jumper, 12-27
Customer Order Number, 13-1
Customizing AUTOEXEC.BAT and CONFIG.SYS
  Files, 3-1, 3-2
Cutter Compensation Commands, 7-52
CVI Command, 7-55
CW Rotation Limit Switch, 12-7
CW_CIRCLE Command, 9-6, 9-8
Cycle button, 4-38
CYCLE Command, 7-56

D

DAC Command, 7-57
Default password, 4-39
Definitions of Terms, 0-1
Demo command - Tools menu, 4-19
Diagnostic command - Tools menu, 4-15
Diagnostic Screen, 3-11, 3-12
  accessing from the main Startup screen, 3-11
DIO500 Option, 1-5
Direct Variables, 7-3
DISABLE Command, 7-58
Disabling Axes, 7-58

Display Servo Command, 7-59
Display submenu - Axis scope screen, 4-24
Displaying Messages, 7-89
DR500 Amplifier Chassis Part Numbers, 1-4
DR500 Amplifier Chassis Styles, 1-4
DR500 Drive Rack, 1-5
Drive Faults, 7-64
DSP 56001 Related Test Points, D-2
Dual Loop Operation, 5-64
DWELL Command, 7-60, 9-3, 9-6, 9-13

E

Edit menu, 4-4, 4-39
Edit menu commands, 4-4
Edit parameters window, 5-1
Electrical Characteristics of a Single Ended Encoder Interface, D-6
Electrical Characteristics of the UNIDEX 500 Amplifier Enable Ou, D-11
Emergency Stop, 7-64
ENABLE Command, 7-63, 9-7
Enabling Axes
  functional verification prior to, 3-11
Encoder
  multiplier box, use with, 12-24
  'times 4' multiplication, 12-1, D-6
Encoder Gantry, 5-21
Encoder Sampling Frequency Jumper, D-4
Encoder Signal Pinouts, D-10
Encoder Signal Specifications, D-6
Endless Looping Program, 7-102
ENDWHILE statement, 7-145
English Measurement System, 5-5
English System, 5-10
Equate Function, 7-5
ERROR Command, 7-64
Error Mapping, 1-3
Error Mask, System, 7-64
Errors, 13-4
Evaluation Hierarchy, 7-5
Exit command, 4-2, 4-4
EXIT Command, 7-65, 9-3, 9-7
Expansion Slot, 3-4
Exponentiation Function, 7-5

F

F2 abort button, 4-36
F3 pause button, 4-36
F4 load button, 4-36
F5 Jog button, 4-38
F6 Password function key, 4-39
F7 Diagnostic button, 4-42
F8 Fault ack button, 4-42
F9 Reset button, 4-42
Factory Configuration for UNIDEX 500 RDP, 5-68
Fail Safe Brake, 12-12
FAULT ACKNOWLEDGE Command, 7-66
Fault Conditions, 13-5
Fault Mask Bit Descriptions, 5-107
Fault Mask Parameters, 5-108
Faults, 3-13
Faults tab, 5-106
Feedback
  phasing and verification, 3-15
Feedback Channels, Types and Additional Hardware, 5-70
Feedback Channels, Types and Hardware for Secondary Feedback Channel, 5-75
Feedback Device Error, 7-64
Feedback Types and Hardware for Primary Feedback Channels, 5-72
Feedback Verification, 3-15
Field Service Information, 0-1
Field Service Policy, 0-1
File menu, 4-2
File menu commands, 4-2
File submenu - Axis scope screen, 4-20
File submenu - Edit parameters, 4-6
Fillets, 9-8
Firmware command, 4-2, 4-3
FL (Filter Time Constant) Command, 7-142
Floating Point Numeric Constants, 7-3
Flowchart Overall tuning process, 6-10
Freerun, 1-2
FREERUN Command, 7-68
Function command - Windows menu, 4-35
Functions, 7-3
Functions menu, 4-48

G

G Codes, 9-3, 9-4, 9-7
G0 Command, 9-4, 9-7, 9-13
G1 Command, 7-41, 9-4, 9-7
G2 Command, 7-37, 7-39, 7-41, 9-4, 9-7
G24 Command, 7-113
G3 Command, 7-41, 9-7
G4 Command, 9-4
G70 Command, 9-4
G8 Command, 9-4, 9-7
G9 Command, 9-4, 9-7
G91 Command, 9-4
G92 Command, 9-13
Gain adjust option - Windows menu, 4-45
GAIN Command, 7-69
Gains option, 4-27
GEAR Command, 7-70, 9-10
Gearing Function, 1-2
General Parameters, C-1
General tab, 5-2
Getting Started
  fine tuning the system, 3-9

important parameter configurations, 3-9
Glossary of Terms, 0-1
GOTO Command, 7-71, 9-9
Grouping Function, 7-5

H

Hall Effect Switches, 12-15
Hall States and Motor Phase Labels, 7-92
HALT Command, 7-73
Handling the U500 Card, 2-2
Handwheel Option, 1-5
Hardware
display of diagnostic information, 3-11
display of servo diagnostics, 3-11
Hardware Limits, 7-64
Hardware status diagnostics, 4-18
Helical Interpolation, 1-3
Helix Motions, 7-40
Hexadecimal Address Ranges, 3-4, 12-31
Hexadecimal Numeric Constants, 7-3
HOME Command, 7-74, 9-3, 9-6
Home Cycle, 5-38, 5-42, 12-24
use of marker pulse qualification jumper setting, 12-24
Home Cycle Following a Power-Up, 5-39
Home Limit Switch, 12-7
Home Marker Option, D-7
Homing and limits tab, 5-37
Homing Problems, 13-7
Host Enable Test Point, 12-22, D-2
Host Interface, 5-9
Host Read/Write Test Point, 12-22, D-2
HW500 Handwheel Option, 1-5

I

ICMD Signals, 12-9
IF Command, 7-75, 9-9
IMMEDIATE Command, 7-77
Incremental Motion, 9-2
Incremental Positioning, 1-2
INDEX Command, 7-78, 9-3, 9-6
Inductosyn Feedback, 1-5
Initial servo parameters - tach tuning, 6-25
Initialization
LED state during, 3-4
problems during, 3-4, 12-31
Initialization Failure, 13-2
In-Position Integrator - Ki, 6-20
Input potentiometer, 6-31
Inputs, 9-13
limits, 12-7
Inspecting the UNIDEX 500 Control Board, 2-2
Installation, 3-3
flowchart of process, 3-14
Installation Problems, 13-2
INT Command, 7-80
Integral Error, 7-64
Integral Error Trap, 6-11, 6-24
Integral gain - Ki, 6-21
Integral Trap Error, 13-4
Interpolation
circular, 9-7
Interrupt the PC, 12-27
jumper setup for custom applications, 12-27
INTERUPT Command, 7-80
Inverse sine trajectory, 5-24
IRQ10, 12-28, D-4
IRQ11, 12-28, D-4
IRQ12, 12-28, D-4
IRQ3, 12-28, D-4
IRQ4, 12-28, D-4
IRQ7, 12-28, D-4
IRQA, D-2
IRQB, D-2
iSBX Expansion Port, 1-5
iSBX Interrupt Jumper Settings, D-4
iSBX Option Jumper Settings, D-4

J

JBV Joystick Option, 1-5
Job Number Files, 3-2
JOG Command, 7-82
Joystick, 7-124
Joystick and Digitizer, 1-5
Joystick center position, 5-30
Joystick deadband, 5-29
Joystick digitizing, 4-31
Jumper Configurations, D-3
Jumpers
PC bus interrupt, 12-3, D-8
Jumpers JP4-JP9, 3-6

K

Ki Integral gain, 6-3, 6-15, 6-21
Kp Proportional gain, 6-3, 6-13, 6-21
Kpos Position gain, 6-3, 6-16, 6-21, 6-27

L

Label Markers, 7-82, 9-9
Laser Firing Control Card, 1-5
LED, D-3
location, 3-4
operation states, 3-8
LED Problems, 13-2
Limit Inputs, 12-7
Limit Switch Input, 12-7
Limit Verification, 3-15
Linear and circular motion, 5-35
LINEAR Command, 7-83, 9-3, 9-6

Linear trajectory, 5-24
Load command, 4-3
Load program command, 4-2
LOOP Command, 7-85, 7-95
Looping Program, 7-102
LVDT Command, 7-86

M

M0 Command, 7-87, 9-7
M2 Command, 9-4
Machine position Data, 3-12
Machine Step, 5-95
Machine Steps per Unit, 5-95
Main Connector Pinout for the UNIDEX 500, D-12, D-13
Map Axis, 5-19
Master and Slave Axes, 1-2, 5-20
Mathematical Function Commands, 7-3
Mathematical Operators, 7-5
MCOMM Command, 7-88, E-2
MDI option - Windows menu, 4-45
Memory
  ranges for base addresses, 3-4, 12-31
Menu bar - Edit parameters window, 5-2
MESSAGE Command, 7-89, 9-3, 9-6, 9-10
Metric Measuring System, 5-5
MFO option - Windows menu, 4-46
Motion
  non-synchronized using INDEX command, 7-78
motor and encoder rotation, 5-64
Motor and feedback configuration tab, 5-62
Motor Buzzes, 13-4
MOTOR COMMUTATION Command, 7-88
Motor Drive Types, 5-77
Motor Drivers, 1-4
  AM16015C, 1-4
  AS16015C, 1-4
  brushless, 1-4
  DC brush, 1-4
  DS16020C, 1-4
  DS16030C, 1-4
  microstepping, 1-4
Motor Drivers, List of, 1-4
Motor feedback and configuration parameters, 5-71
Motor Has No Torque, 13-4
Motor Phase Labels and Hall States, 7-92
Motor Related Test Points, D-1, D-2
Motor Rotation, 5-40
MOTOR SETUP Command, 7-91
MSET Command, E-2
Multiplane configuration, 5-17
Multitasking, 1-2
  timing issues, 5-15

N

NEXT Command, 7-95
Non Ramp Time, 9-9
Notch filter, 5-87
Notch filter - example, 5-89
Number field, 4-6
Number of contour planes, 5-17

O

OP500 Cable Option, 1-5
Operators and Evaluation Hierarchy, 7-5
Opto-22 Interface Board, 1-5
OUTPUT Command, 7-97, 12-11
Output Commands
  linked to cam table points, 1-3
  linked to spline table points, 1-3
Outputs
  brake control, 7-32
  Open-collector type, 12-11
Overriding Scale Factor, 7-117, 9-16
Overview of the UNIDEX 500 System, 1-1

P

Parameter 000, 5-17
Parameter 001, 5-6
Parameter 002, 5-34
Parameter 015, 5-8
Parameter 098, 5-30
Parameter 099, 5-31
Parameter 500, 5-32
Parameter 501, 5-32
Parameter command, 4-3
Parameter command - Edit menu, 4-5
Parameter display window, 5-2
Parameter number field, 5-3
Parameter tabs, 5-2
Parameter value field, 5-3
Parameter x00, 5-95
Parameter x01, 5-95
Parameter x02, 5-40
Parameter x03, 5-41
Parameter x04, 5-42
Parameter x05, 5-43
Parameter x06, 5-43
Parameter x07, 5-44
Parameter x09, 5-45
Parameter x10, 5-46
Parameter x11, 5-98
Parameter x12, 5-99
Parameter x13, 5-99
Parameter x14, 5-100
Parameter x16, 5-51
Parameter x17, 5-52
Parameter x18, 5-53

Parameter x19, 5-54
Parameter x20, 5-55
Parameter x22, 5-46
Parameter x23, 5-46
Parameter x24, 5-85
Parameter x25, 5-85
Parameter x26, 5-85
Parameter x27, 5-86
Parameter x28, 5-86
Parameter x29, 5-86
Parameter x30, 5-87
Parameter x31, 5-87
Parameter x32, 5-87
Parameter x33, 5-87
Parameter x34, 5-87
Parameter x35, 5-101
Parameter x37, 5-102
Parameter x38, 5-72
Parameter x39, 5-75
Parameter x40, 5-76
Parameter x41, 5-76
Parameter x43, 5-78
Parameter x44, 5-79
Parameter x45, 5-79
Parameter x46, 5-80
Parameter x47, 5-80
Parameter x48, 5-56
Parameter x49, 5-58
Parameter x50, 5-103
Parameter x51, 5-103
Parameter x52, 5-104
Parameter x53, 5-59
Parameter x54, 5-60
Parameter x55, 5-108
Parameter x56, 5-108
Parameter x57, 5-108
Parameter x58, 5-109
Parameter x59, 5-109
Parameter x60, 5-109
Parameter x61, 5-110
Parameter x62, 5-92
Parameter x63, 5-80
Parameter x64, 5-81
Parameter x65, 5-81
Parameter x66, 5-82
Parameter x67, 5-83
Parameter x68, 5-83
Parameter x69, 5-83
Parameter x70, 5-61
Parameter x71, 5-104
Parameter x73, 5-47
Parameter x74, 5-48
Parameter x75, 5-48
Parameter x76, 5-48
Parameter x77, 5-49
Parameter x78, 5-92
Parameter x79, 5-83
Parameter x80, 5-83
Parameter x81, 5-49
Parameter x82, 5-83
Parameter x83, 5-93
Parameter x84, 5-61
Parameter x85, 5-105
Parameters
  2-D Error Mapping Enabled, 5-105
  A/D Channel n Joystick Center Position, 5-30
  A/D Channel n Joystick Deadband, 5-29
  Abort Motion, 5-109
  Abort on input high, 5-32
  Absolute Mode Scale, 5-104
  AC Brushless Motor Base Speed Advance, 5-83
  AC Brushless Motor Commutation Factor, 5-78
  AC Brushless Motor Phase Advance Base Speed, 5-82
  AC Brushless Motor Phase Offset, 5-79
  AC Brushless Motor Phase Speed, 5-83
  AC Brushless Motor Phase Speed Advance, 5-83
  Aff (Acceleration Feed Forward), 5-86
  AUX OUTPUT, 5-109
  Aux Output Active High, 5-61
  axis, C-5
  Axis n Gantry {Y/None}, Slave {1, 2, 3, 4}, 5-20
  Axis n Map to Plane {0, 1-4} as {X, Y, Z, U}, 5-19
  Axis n Rollover, 5-7
  Backlash, 5-102
  CCW Software Limit, 5-46
  Clamp Current Output, 5-59
  Clamp Feedrate, 5-25
  configuring, 3-9
  Contour Feedrate, 5-36
  Contouring Mode, 5-28
  Corner Rounding (028, 046, 064, 082), 9-9
  Corner Rounding Non-ramp Time, 5-26
  CW Software Limit, 5-46
  Disable Axis, 5-108
  Drive (Motor) Type, 5-77
  Drive Fault Signal Active Low, 5-61
  Enable Brake, 5-110
  Enable Vel/Accel Feedforward During Home, 5-47
  Encoder Feedback, 5-79
  Encoder Multiplication, 5-83
  English Conversion Factor, 5-95
  English Mode Number of Decimal Digits, 5-12
  Error Mask Fault, 5-108
  Filter Time Constant, 5-93
  general, C-1
  Halt Queue, 5-109
  Home Direction is CCW, 5-40
  Home Marker Search Speed, 5-49
  Home Offset, 5-43
  Home Switch Normally Open, 5-41
  Home Switch to Marker, 5-44
  Home/Limit Switch Debounce, 5-45, 5-49
  In Position Deadband, 5-101

Interrupt PC, 5-108
Joystick
  High Speed, 5-103
  Lowspeed, 5-103
Keep Axis Position after Reset, 5-6
Ki (Velocity Loop Integrator), 5-85
Kp (Velocity Loop Proportional Gain), 5-86
Kpos (Position Loop Gain), 5-85
Limit Switch Normally Open, 5-45
Linear Type Acceleration/Deceleration, 5-24
list of, 0-1
Max Ac/De, 5-51
Max Integral Error, 5-55
Max Position Error, 5-54
Max Velocity Error, 5-53
Metric Conversion Factor, 5-95
Metric Mode Number of Decimal Digits, 5-11
MFO Enable in Freerun, 5-100
MFO, 0 No MFO-pot, 1-255 Pot Offset, 5-34
Normal Home Feedrate, 5-43
Notch Filter, 5-85
Notch Filter D1, 5-87
Notch Filter D2, 5-87
Notch Filter N0, 5-87
Notch Filter N1, 5-87
Notch Filter N2, 5-87
Number of Contour Planes, 5-17
Option Board Setup Code, 5-31
Orthogonality Correction Table Enabled, 5-104
Pause Enable in Freerun, 5-99
plane, C-4
Plane n Contour Ramping Time, 5-35
Plane n Indexing Segment Time, 5-23
Plane n Metric System, 5-10
Positive (+) Jog Same Direction as + Move, 5-99
Positive (+) Move is CW, 5-98
Power-on Home Feedrate, 5-42
Primary Current Command Offset, 5-83
Primary Feedback Setup Code, 5-76
Primary/Position Feedback Channel, 5-72
PSO Mailbox Dual_port RAM Base Address, 5-8
Reverse Joystick Direction, 5-105
RMS Current Sample Time, 5-58
RMS Current Trap, 5-56
Safe Zone Limits, 5-48
Safe Zone Output Bit, 5-30
Secondary Current Command Offset, 5-83
Secondary Feedback Setup Code, 5-76
Secondary/Velocity Feedback Channel, 5-75
Servo Loop Type, 5-92
Servo Loop Update Rate, 5-92
Stepper Encoder Speed, 5-81
Stepper Encoder Verification, 5-81
Stepper High Current, 5-80
Stepper Low Current, 5-80
Stepper Microstepping Resolution, 5-80
summary of, 0-1
  Switch to Mechanical Stop, 5-46
  Top Feedrate, 5-52
  Use Home Limit During Home Cycle, 5-48
  User Interrupt Setup Code, 5-32
  Vff (Velocity Feed Forward), 5-86
  Which Output Bit for AUX.OUTPUT, 5-60
  X, Y, Z, and U Axes' Point-to-point Feedrates for Plane n, 5-36
Parameters 003, 004, 005, and 006, 5-19
Parameters 007, 008, 009, and 010, 5-20
Parameters 018, 036, 054, and 072, 5-23
Parameters 019, 037, 055, and 073, 5-35
Parameters 020, 038, 056, and 074, 5-10
Parameters 021, 039, 057, and 075, 5-24
Parameters 022, 040, 058, and 076, 5-36
Parameters 023 through 026, 041 through 044, 059 through 062, and 077 through 080, 5-36
Parameters 027, 045, 063, and 081, 5-25
Parameters 028, 046, 064, and 082, 5-26
Parameters 029, 047, 065, and 083, 5-11
Parameters 030, 048, 066, and 084, 5-12
Parameters 090, 092, 094, and 096, 5-29
Parameters 091, 093, 095, and 097, 5-30
Parameters 11, 12, 13, and 14, 5-7
Parameters 31, 49, 67, and 85, 5-28
Parameters command, 4-2
Parameters window, 6-11, 6-24
Part Rotation Command, 7-112, 9-14
Password protection window, 4-39
PAUSE Command, 7-100
PB16 Opto-22 Interface Board, 1-5
PB24 Opto-22 Interface Board, 1-5
PB8 Opto-22 Interface Board, 1-5
PC BUS Interrupt Jumpers, 12-3, D-8
PC Expansion Slot, 3-4
PC Interface to the UNIDEX 500, D-2
PC Power Problems, 13-2
PC Reset
  clearing machine positions after, 5-6
PLANE Command, 7-101
Plane Parameters, C-1, C-4
Planes, 5-14
  timing issues when programming, 5-15
PLC Command, 7-102
PLC Program Operators
  =, <, >, 7-103
  AND, 7-103
  ELSE, 7-103
  ENDIF, 7-103
  ENDWHILE, 7-103
  IF, 7-103
  OR, 7-103
  WHILE, 7-103
Plot submenu - Axis scope screen, 4-21
Point-to-point Motion, 1-2
Polarity Jumpers for Axes, 12-25
Polarity of Amplifier Outputs, 12-25

Position Error, 6-15, 7-64, 13-4
Position Error Trap, 6-11, 6-24
Position Gain - Kpos, 6-21
Position Loop, 6-15, 6-16
Position Registers
  range, 5-7
  size, 5-7
Position tracking tab, 5-5
Positioning
  absolute, 1-2
  incremental, 1-2
Power On Home Cycle, 5-38
Precautions, 1-6
Primary Feedback Setup Codes and Meanings, 5-76
Print option, 4-6
Printing Messages, 7-89
PRM File, 3-2
Program
  terminating flow, 7-65
PROGRAM Command, 7-106, 9-3, 9-7
Program command - Edit menu, 4-14
Program display field, 4-37
Program Execution Levels, 7-2
Program Unit, 5-95
Programming
  command Summary, 7-15
  commands, 7-15
  delay time, 7-60
  evaluation Hierarchy, 7-5
  functions, 7-4
  operators, 7-5
  planes, 5-15
  repeating a program, 7-27
  timing issues, 5-15
Project command, 4-2, 4-3
Project command - Edit menu, 4-4
Project Files, 3-2
Proportional gain - Kp, 6-21
Proportional gain parameter Kp, 6-20
PSO-PC Card, 1-5

Q

Quadrature Signals, 12-1, D-6
QUEUE Command, 7-108
QUEUE INPUT Command, 7-108
Quit button, 4-38

R

RAMP Command, 7-110
RDP Resolution and Setup Codes, 5-70
READ.ME File, 11-1
Real Time Axis position Display, 3-12
Real time commanded position registers, 7-7
Real time feedback position registers, 7-7
REFERENCE Command, 7-111
Relative Motion, 9-2
Relative Position Registers, 7-6
Reset command, 4-2, 4-4
Resolver Feedback, 1-5
Resolver Gantry, 5-21
Resolvers, 5-44, 5-68
Resolver-to-digital Converter Card, 1-5, 5-64
Restart button, 4-38
RETURN Command, 7-111
RMS Current Error, 7-64
Root Directory
  project files, 3-2
ROT Command, 7-112, 9-14
Rotary Applications, 5-7
  modulo distance, 5-7
  rollover point, 5-7
ROUNDING Command, 7-113, 9-3, 9-6
RS-232, 7-43, 7-44

S

Safety Procedures, 1-6
SBX-ENC1 Option, 1-5
Scaling and Resolution, Defining, 5-5
Scanning Program, 7-102
SCF Command, 7-117, 9-16
Second Order Low Pass Filter, 5-91
Secondary Feedback Setup Code, 5-76
Select axis field, 4-6
Select axis radio buttons, 4-7
Servo gain potentiometer, 6-27
Servo Loop Gains, 7-69
Servo Loop Setup, 3-15
Servo Loop tab, 6-11, 6-24
Servo Loop Update Rate parameter, 6-11, 6-24
Servo Loops
  displaying real-time data, 7-59
  tracking when axis is disabled, 7-58
Servo Motors
  setting a fixed vector, 7-91
Servo parameters, 6-12, 6-25
Servo Problems, 13-4
SHUNT500 Option, 1-5
SIN Function, 7-4
Single button, 6-13, 6-27
Single Ended Encoders, D-6
Single plane configuration, 5-17
Sinusoidal commutation, 5-67
SKEY Command, 7-118
SLEW Command, 7-124
Software
  base address, 3-4, 12-31
  initialization and LED state, 3-4
Software Base Address Configuration, 3-5
SOFTWARE Command, 7-126
Software Configurations, 3-9
Software Limits, 7-64

Software setup - Single-ended encoders, 12-2, D-7
Software Startup, 3-4
Software status diagnostics, 4-16
SOFWARE POSITION Command, 7-68
Spherical Interpolation, 1-3
SPLINE Command, 7-128
Spline Table Motion, 1-3
Splining, 9-12
SQR Function, 7-4
Square Root Function, 7-4
START Command, 7-129
Startup, 3-4
Startup Problems, 13-2
Static Charge Buildup, Removing, 2-2
Status bar, 4-43
Status option, 4-26
Stepper Motor Problems, 13-3
Stepper Motors, D-7
SUBROUTINE Command, 7-111, 7-130
Summary of Parameters, 0-1
Supported Programming Functions, 7-4
System Error Mask, 7-64
System Password prompt window, 4-40
System Password window, 4-40
System Registers, 7-6
System Scaling, 5-95

T

Tachometer based velocity loop, 6-20
Target Tracking Commands, 7-131
Technical Support Questions, 13-1
Termination Resistor Configuration, D-8
Test Points, D-1, D-3
  DSP 56001 chip, D-1
  DSP 56002 chip, 12-21
  miscellaneous, 12-21, D-1
  motor related, 12-21, D-1
  PC interface, 12-21, D-1
Time-based Motions, 1-2
Timing issues when multitasking, 5-15
Tools menu, 4-15, 4-31
Tools menu commands, 4-15
Tools submenu - Axis scope screen, 4-25
Tracking Display, 3-12
TRAJECTORY Command, 7-132
Traps, 3-13
Traps tab, 5-50
TRIGGER Command, 7-133
Trigger submenu - Axis scope screen, 4-22
Trigger submenu options, 4-22
Troubleshooting
  using parameter 001, 5-6
Tuning option, 6-1
Tuning procedures for Servo Loops, 6-9
Tuning procedures for Tachometer Loops, 6-22
Tuning tips, 6-19
Tuning with tachometer feedback, 6-20

U

U500, 1-3
U500DIAG.EXE Utility, 11-2
U500PLUS, 1-3
U500ULTRA, 1-3
UCP Register, 7-7
UFP Register, 7-7
UMFO Command, 7-133
UNIDEX 500
  base model, 1-1
  board installation, 3-3
  conroller options, 1-5
  enhancing operation with accessories, 1-5
  functions, 1-2
UNIDEX 500 PLUS (U500PLUS)
  functions, 1-3
UNIDEX 500 ULTRA (U500ULTRA)
  features, 1-3
UNIDEX 511
  Joystick, 7-124
UNIDEX 550
  I/O channel memory, 3-4, 12-31
Units submenu - Axis scope screen, 4-25
Unpacking the UNIDEX 500, 2-1
Unused Inputs, 7-10
Utilities
  ASCPRM.EXE, 11-1
  U500DIAG.EXE, 11-2

V

Value field, 4-6, 4-7
VAR Command, 7-134
Variables, 7-43, 7-89, 9-10
Velocity Command, 6-16
VELOCITY Command, 7-135, 9-3, 9-6
Velocity Error, 6-13, 7-64
Velocity feedforward gain - Vff, 6-20, 6-21
Velocity loop adjustment, 6-13
Velocity Profiled Motion, 1-2, 9-2
Velocity Profiling, 9-2, 9-5, 9-7
Version Number, Software, 13-1
Vertical Axis Position Brake, 1-5
Vff, 6-21
Vff Velocity feedforward gain, 6-3

W

WAIT Command, 7-144, 9-3, 9-6
Warnings, 1-6
Warranty Information, 0-1
Warranty Policy, 0-1
Watchdog Timer, D-4
WHILE/ENDWHILE Command, 7-145

Windows menu, 4-35
Windows menu commands, 4-35
Writing Messages to a File, 7-89
Writing Messages to a Serial Port, 7-89

X

X Command, 9-12
XCP Register, 7-7
XFP Register, 7-7

Y

YCP Register, 7-7
YFP Register, 7-7

Z

ZCP Register, 7-7
ZFP Register, 7-7

∇ ∇ ∇

READER'S COMMENTS

The UNIDEX 500 Motion Controller and Windows Software Operation and Technical Manual
P/N EDU 150, July, 2000

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) 967-6870

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP