Motion Engineering, Inc.

33 South La Patera Lane
Santa Barbara, CA 93117
ph (805) 681-3300
fax (805) 681-3311
info@motioneng.com
www.motioneng.com

DSPpro Series Motion Developer's User Manual

March 2002

DSPpro Series
Motion Developer's Manual

Mar 2002
Part # M001-0024 rev. B

Motion Engineering, Inc.
33 South La Patera Lane
Santa Barbara, CA 93117-3214
ph 805-681-3300
fax 805-681-3311
e-mail: technical@motioneng.com
website: www.motioneng.com ftp site: ftp.motioneng.com

CONTENTS

INTRODUCTION 1

1.1 Troubleshooting & Technical Support 1

1.2 Quick Start 2

1.3 Software Updates 2

1.4 A Note to OEMs 2

QUICK START 3

2.1 Installing DSPpro Controllers 3

2.1.1 Hardware Installation 3
2.1.2 Software Installation 4
2.1.3 Connections & Wiring 4

2.2 Tuning your system and verifying basic operation 5

DSPPRO CONTROLLER DESIGN 9

3.1 Architecture 9

3.1.1 Program Storage 10

3.2 Firmware Execution 10

3.2.1 Read All Encoders 10
3.2.2 Read Analog and Parallel Input 10
3.2.3 Calculate Next Trajectory Point 10
3.2.4 Check for Event Triggers 11
3.2.5 Perform Event Actions 11
3.2.6 Calculate and Set DAC Output 11

3.3 Hardware Features 12

3.3.1 Step Motor Control via VFCs 12
3.3.2 Communication via three words in I/O Space 12

HARDWARE INSTALLATION 14

4.0 Memory Addressing (DSPpro-VME) 14

4.1 Installing the DSPpro-VME 14

4.1.1 DSPpro-VME Dip Switch Locations 14
4.1.2 DSPpro-VME Base Memory Address Switch Settings 15
4.1.3 DSPpro-VME Interrupt (IRQ) Switch Settings. 15
4.1.4 DSPpro-VME Board Installation Procedures 15

4.2 Installing the DSPpro-Serial 16

MEI SOFTWARE 17

5.1 The SETUP Program (DSETUP.EXE & SSETUP.EXE).....17
5.1.1 Overview..... 17
5.1.2 Using SETUP with the DSPpro..... 17
5.1.3 SETUP Screens..... 21
5.2 The VERSION Program (VERSION.EXE) 34
5.3 The CONFIG Program (CONFIG.EXE).....35

MOTOR WIRING 37
6.1 Connection Accessories 37
6.2 DSPpro-VME Motor Signal Header Locations 39
6.2.1 DSPpro-VME Motor Signal Pinouts (P4-P7)..... 39
6.3 Motor Signal Header Locations - DSPpro-Serial 40
6.3.1 DSPpro-Serial Motor Signal Pinouts (P1-P4) 40
6.4 Wiring Servo Motors.....41
6.4.1 Velocity/Torque Mode..... 41
6.4.2 Encoder Input 41
6.4.3 Brush Servo Motors..... 41
6.4.4 Brushless Servo Motors..... 43
6.4.5 Step-and-Direction Controlled Servo Motors..... 44
6.5 Wiring Step Motors.....44
6.5.1 Open-Loop Step Motors 44
6.5.2 Direction Pulse Synchronization 46
6.5.3 Closed-loop Step Motors 47
6.6 Wiring for Dual-Loop Control 48
6.7 Interferometer Input Wiring (DSPpro-VME only) 48
6.7.1 Excel Precision 1000A Interferometer/Axis Board..... 49
6.7.2 Excel 1032B Interferometer Wiring Pin-Out Listing 50

I/O WIRING 53
7.1 General 53
7.2 I/O Wiring - DSPpro-VME.....53
7.3 I/O Wiring - DSPpro-Serial.....56
7.3.1 Dedicated I/O Wiring..... 56
7.3.2 User I/O Wiring 56
7.4 Home and limit switch wiring (DSPpro-VME only).....57
7.5 Dedicated and user output wiring 58
7.6 Opto-isolation 59
7.7 Analog input wiring 59
7.7.1 DSPpro-VME 59
7.7.2 DSPpro-Serial..... 59
7.8 8254 Counter wiring - DSPpro-VME.....60

DEVELOPING YOUR APPLICATION 61

8.0 Overview 61

8.1 Write the application program 62

8.2 Testing the Application Program 62

8.2.1 Configuring Communications 63
8.2.2 Using RCONSOLE to Test Your Application (DSPpro-PC only) 64
8.2.3 Using REMSVR to Test Your Application (DSPpro-VME or DSPpro-Serial) 65
8.2.4 Configure DOS startup files 66

8.3 Committing your application to flash memory 66

8.3.1 Committing your application to the DSPpro-VME or DSPpro-Serial 67

APPENDIX A: CONNECTOR PINOUTS 68

A.1 DSPpro-VME Connector Layout 68

A.2 Connector pinouts (DSPpro-VME) 68

A.3 DSPpro-Serial Connector Layout 71

A.4 DSPpro-Serial Connector Pinouts 71

APPENDIX B: SPECIFICATIONS 77

B.1 DSPpro-VME specifications 77

B.2 DSPpro-Serial specifications 80

APPENDIX C: TUNING YOUR SYSTEM 83

C.1 General Description 83

C.1.1 The Digital Filter 83

C.2 Tuning Parameters 84

C.2.1 Proportional Gain (K_p) 84
C.2.2 Derivative Gain (K_d) 85
C.2.3 Integral Gain (K_i) 86
C.2.4 Velocity Feed Forward (K_v) 87
C.2.5 Acceleration Feed-Forward (K_a) 88
C.2.6 Integration Limit 88
C.2.7 Offset (K_o) 88
C.2.8 Shift 88
C.2.9 Friction Feed Forward 89

C.3 Tuning Closed-Loop Servos 89

C.3.1 Step 1: Set Proportional Gain (K_p) 89
C.3.2 Step 2: Set the Derivative Gain 89
C.3.3 Step 3: Iterate Steps 1 and 2 90
C.3.4 Step 4: Set Integral Gain (K_i) 90
C.3.5 Step 5: Set Velocity and Acceleration Feed Forward 90

C.4 Tuning Closed-Loop Steps 90

C.4.1 Step 1: Set Proportional Gain (K_p) 91
C.4.2 Step 2: Set Velocity and Acceleration Feed Forward 91

C.4.3 Step 3: Setting the Integral Gain (K_i) 91

INDEX **92**

Chapter 1

INTRODUCTION

This manual describes how to install, wire, and use available development tools for MEI DSPpro motion controllers. It is intended as a general reference guide for the DSPpro-Serial, DSPpro-PC and DSPpro-VME and should be used in conjunction with the MEI *DSP-Series Motion Controller C Programming Manual*, which documents the specific motion control functions provided by the MEI standard C function library.

1.1 Troubleshooting & Technical Support

MEI takes technical support seriously. We want your system to work! Our staff of application engineers are knowledgeable and dedicated to answering questions during the development of your system. MEI provides technical support 24 hours a day free of charge for **5 years** from the date of your last purchase. Please feel free to call, fax, or e-mail us. We only ask that you take the time to read the manuals and release notes to make sure that information has not been overlooked.

Before calling Technical Support, make sure you know which product and what version of software you are using. The software version is printed on the distribution disks and is displayed in the SETUP program provided in the DOS, Windows 3.x, and Windows95 software support disks. If you FAX or e-mail us your program (or code fragments) be sure to include specific notes about where you are having problems. Also, don't forget to include your phone and FAX numbers and e-mail address.

The latest releases of software and firmware are available by anonymous FTP at our Internet site at `ftp://ftp.motioneng.com` and on our 24 hour BBS at (805) 681-3313. No special registration is necessary. The BBS supports baud rates up to 28.8 kbps, No Parity, 8 data bits, 1 stop bit. Files may be uploaded or downloaded. If you upload a file to our FTP site or BBS, be sure to let us know by phone, FAX, or e-mail.

In addition, product information, recent press releases, and registration forms for MEI-sponsored technical conferences can be found on our Web site at `http://www.motioneng.com`.

If you aren't sure about how to begin developing your application, write down your questions and/or a specification and FAX them to MEI at (805) 681-3311. We will answer your questions and/or write a small sample program and FAX or e-mail them to you. We cannot write your application code for you but we can save you development time by providing sample code.

Also, take a look at the *Sample Applications* distribution disk; it contains a variety of sample programs that address common questions. This disk is constantly being updated with new sample programs and creative solutions. The latest sample applications are available on our FTP site and the BBS in a file called APPS.ZIP.

Following is a list of our offices should you need to contact us.

Office	Phone	Fax	E-mail	BBS
West Coast	805-681-3300	805-681-3311	tech@motioneng.com	805-681-3313
East Coast	508-264-0051	508-264-0057		
Midwest	773-631-4992	773-631-4935		
Japan	(03) 5229-7007	(03) 3235-5655	yamada@motioneng.co.jp	

1.2 Quick Start

For developers familiar with motion systems who want to get their motors turning as soon as possible, we offer the "Quick Start" section. Also, to make it easy to test the hardware and tune servo systems, a DOS-based SETUP program is provided with the DOS, Windows 3.x, and Windows 95 software support disks. The SETUP program is designed to be self-explanatory for experienced developers who prefer to explore and learn on-the-fly.

1.3 Software Updates

MEI periodically releases new software/firmware versions. New features are implemented, performance enhanced and new applications developed. The latest firmware and software releases are available on our FTP site at `ftp://ftp.motioneng.com` as well as the MEI BBS.

1.4 A Note to OEMs

MEI always ships DSPpro controllers with the latest firmware. When building multiple machines we strongly recommend that you save a configured version of your firmware to a disk. Then, the next time you build a machine simply load that firmware (from disk) to the DSPpro controller using the CONFIG program. This method will give you the greatest control and ease in building future machines.

Chapter 2

QUICK START

This chapter offers a fast and easy installation procedure for those already familiar with MEI controllers, software libraries, and motion controller connections. For more information on MEI software libraries, please refer to the MEI *DSP-Series Motion Controller C Programming Manual*. Those unfamiliar with wiring controllers should see the "Hardware Installation" and "Motor Wiring" sections of this manual for wiring diagrams and a detailed discussion of installation procedures.

Depending on your operating system and configuration, you may be able to use one or more included DOS-based utilities to help you tune your system and verify initial operation. For more information, refer to section 2.2, "Tuning your system and verifying basic operation."

2.1 Installing DSPpro Controllers

2.1.1 Hardware Installation

DSPpro-PC

The DSPpro-PC is a board-level controller designed for PC compatible (ISA bus) computers.

1. Open the shipping container and carefully remove its contents. Be sure to observe proper ESD handling precautions.
2. Turn off the host computer.
3. Select an available ISA expansion slot and remove its expansion cover plate.
4. The default memory location of the DSPpro-PC is 0xD000:0000. If you need to change this, refer to section 4.1, "Installing the DSPpro-PC."
5. Remove the anti-static protective cover from the DSPpro-PC bus connectors.
6. Insert the DSPpro-PC in the ISA bus connector and secure the card in place with the screw from the cover plate.

For detailed hardware installation instructions, please see section 4.1.

DSPpro-VME

The DSPpro-VME is a single, 6U slot, board-level controller designed for VME-based computers.

1. Open the shipping container and carefully remove its contents. Be sure to observe proper ESD handling precautions.
2. Turn off the host computer.

3. Select an available VME expansion slot and remove its expansion cover plate.
4. The default memory location of the DSPpro-VME is 0xFFFF0300. If you need to change this, refer to section 4.2, "Installing the DSPpro-VME."
5. Insert the DSPpro-VME in the VME bus and firmly press the card into place until the card is properly seated and locked into place.
6. Connect the *Console* port on the DSPpro-VME to a serial port (e.g. COM1) of the host computer using a **null modem cable** (available from MEI as accessory cable CBL-D9 CONSOLE).

For detailed hardware installation instructions, please see section 4.2.

DSPpro-Serial

The DSPpro-Serial is a standalone controller that communicates with a host computer using an RS-232 serial port. The following instructions describe how to setup the DSPpro-Serial for application development. After application development, the DSPpro-Serial can operate independent of the host computer.

1. Open the shipping container and carefully remove its contents.
2. Turn off the host computer.
3. Connect the *Console* port on the DSPpro-Serial to a serial port (e.g. COM1) of the host computer using a **null modem cable** (available from MEI as accessory cable CBL-D9 CONSOLE).
4. Connect your 24 VDC power lines to the screw-terminal connector which plugs in to the power port of the DSPpro-Serial. Be sure to observe proper polarity.

For detailed hardware installation instructions, please see **s**ection 4.3, "Installing the DSPpro-Serial."

2.1.2 Software Installation

DSPpro motion controllers are available with software support packages for many popular operating systems including DOS, Windows 3.x, Windows95, and Windows NT. Limited support for other development environments is also available. Contact your MEI representative for more information.

Refer to the release notes which accompany your software support package for information on installing the software distribution.

2.1.3 Connections & Wiring

After installing the controller and software, you are ready to wire your system. Since your specific configuration may vary, below are general guidelines for wiring your system. Before beginning, make sure all drives are turned off.

1. Connect drives to motors (consult your drive documentation for connection details).
2. Connect the DSPpro to the drive.

3. Connect the encoders to the DSPpro.
4. Connect power cable to controller (DSPpro-Serial only).

For detailed connections and wiring instructions, please see chapter 6, “Motor Wiring,” and chapter 7, “I/O Wiring.”

2.2 Tuning your system and verifying basic operation

If you are using a PC running DOS, Windows 3.x, or Windows 95 or can run DOS applications using another type of computer, you can use MEI’s SETUP program to tune your motion system and to verify basic controller operation. From within this application, you can modify tuning, generate point to point motion, and test various other aspects of the controller.

Before following these instructions, be sure to properly wire your system as described in chapter 6, “Motor Wiring,” and chapter 7, “I/O Wiring.”

Verifying Operation - Servos

Follow these instructions if your application uses **SERVO** motors.

1. From the factory, the DSPpro is configured to connect to the SETUP program running on the host PC. If you have not modified the DSPpro flash memory, **skip to step 6**.
2. Insert the disk labeled *DSPpro Quick Start Disk* into the floppy disk drive of your PC.
3. If you are using the **DSPpro-Serial or DSPpro-VME**, execute the program by typing `REMSVR` from a DOS prompt on your host PC. If the DSPpro console cable is connected to COM2 of your PC, execute REMSVR by typing `REMSVR /2`. You will see the following output in the REMSVR window of your PC:

```
Motion Engineering Console/Disk Server V1.0
Copyright (C) 1996 Motion Engineering, Inc.
Press Alt-X to exit
Connecting Through COM1
```

Reset the controller by depressing the reset button. Use a paper-clip to reach the recessed reset button on the DSPpro-Serial.

If you are using the **DSPpro-PC**, execute the **RCONSOLE** program by typing `RCONSOLE /r` from a DOS prompt on your host PC. The `/r` switch reboots the controller before starting RCONSOLE. You will see the following output in the RCONSOLE window of your PC:

```
RCONSOLE v1.0  (c) 1996 Motion Engineering, Inc.
```

4. The DSPpro will then reboot, booting from the *DSPpro Quick Start Disk* in the floppy drive of your host PC. You will see the following output in the REMSVR or RCONSOLE window of your PC:

```
Motion Engineering 80C386EX BIOS V1.0
Copyright (C) 1992-1995 Motion Engineering, Inc.
Copyright (C) 1992-1995 General Software, Inc.
00896 KB OK
```

```
Embedded BIOS ROM Disk Enabled
Embedded BIOS REMOTE Disk [COM1] [CONNECTED]

Starting MS-DOS…

Server 1.00A Oct 11 1996 14:10:40
Listening on COM1.
```

5. Exit from RCONSOLE or REMSVR by typing ALT-X.

6. Execute the SETUP program by typing `ssetup` from a DOS prompt on your host PC if you are connecting to a DSPpro-VME or DSPpro-Serial, or `dsetup` if you are connecting to a DSPpro-PC. Note that if you are connecting to a DSPpro-VME or DSPpro-Serial using COM2 of your host PC, type `ssetup /2` to tell the SETUP program to connect through COM2.

7. Configure Axis 0 of the controller as a servo axis, closed loop, bipolar with the Configure/Axis Configuration window (Shortcut key F8).

8. Put the axis in Idle mode using the Status/Axis Status (F3) window.

9. Turn on the amplifier.

10. Test the encoder connections with the Status/Position Status (F2) window - turn the motor and verify it gives the correct number of encoder counts.

11. Apply an initial constant voltage of 100 using the offset register in the Configuration/Tuning Parameters (F7) window. Check to see that positive offsets produce increasing position value. If not, swap the A+ and B+ encoder leads, and swap the A- and B- encoder leads.

12. Clear the position with the Status/Position Status (F2) window.

13. Put the axis in Run mode with the Status/Axis Status (F3) window. The motor should now be servoed.

Verifying Operation - Steppers

Follow these instructions if your application uses **STEP** motors.

1. From the factory, the DSPpro is configured to connect to the SETUP program running on the host PC. If you have not modified the DSPpro flash memory, **skip to step 6**.

2. Insert the disk labeled *DSPpro Quick Start Disk* into the floppy disk drive of your PC.

3. If you are using the **DSPpro-Serial or DSPpro-VME**, execute the **REMSVR** program by typing `REMSVR` from a DOS prompt of you host PC. If the DSPpro console cable is connected to COM2 of your PC, execute REMSVR by typing `REMSVR /2`. You will see the following output in the REMSVR window of your PC:

```
Motion Engineering Console/Disk Server V1.0
Copyright (C) 1996 Motion Engineering, Inc.
Press Alt-X to exit
Connecting Through COM1
```

Reset the controller by depressing the reset button. Use a paper-clip to reach the recessed reset button on the DSPpro-Serial.

If you are using the **DSPpro-PC**, execute the **RCONSOLE** program by typing `RCONSOLE /r` at a DOS prompt. The `/r` switch reboots the controller before starting RCONSOLE. You will see the following output in the RCONSOLE window of your PC:

```
RCONSOLE v1.0  (c) 1996 Motion Engineering, Inc.
```

4. The DSPpro will then reboot, booting from the *DSPpro Quick Start Disk* in the floppy drive of your host PC. You will see the following output in the REMSVR or RCONSOLE window of your PC:

```
Motion Engineering 80C386EX BIOS V1.0
Copyright (C) 1992-1995 Motion Engineering, Inc.
Copyright (C) 1992-1995 General Software, Inc.
00896 KB OK

Embedded BIOS ROM Disk Enabled
Embedded BIOS REMOTE Disk [COM1] [CONNECTED]

Starting MS-DOS…

Server 1.00A Oct 11 1996 14:10:40
Listening on COM1.
```

5. Exit from RCONSOLE or REMSVR by typing ALT-X.

6. Execute the SETUP program by typing `ssetup` from a DOS prompt on your host PC if you are connecting to a DSPpro-VME or DSPpro-Serial, or `dsetup` if you are connecting to a DSPpro-PC. Note that if you are connecting to a DSPpro-VME or DSPpro-Serial using COM2 of your host PC, type `ssetup /2` to tell the SETUP program to connect through COM2.

7. Configure Axis 0 as a step axis, open loop, unipolar with the Configure/Axis Configuration (F8) window.

8. Put the axis in Idle mode using the Status/Axis Status (F3) window.

9. Apply a constant pulse rate using the offset register in the Configuration/Tuning Parameters (F7) window. Note that only positive offset values will produce rotation.

10. Clear the position with the Status/Position Status (F2) window.

11. Put the axis in Run mode with the Status/Axis Status (F3) window.
12. Command a trapezoidal motion using the Motion/Two-Point Motion Window.
13. Verify that the motor turns one rotation when the appropriate number of steps are commanded.

Chapter 3

DSPpro CONTROLLER DESIGN

3.1 Architecture

The DSPpro design combines an Intel 386EX microprocessor with an Analog Devices DSP. High-level decisions and communications with the host processor occur on the 386EX, while the DSP is responsible for numerically intensive calculations.

Board-level DSPpro series controllers are mapped into the memory space of the host CPU, which allows fast, direct binary communication across the data bus.

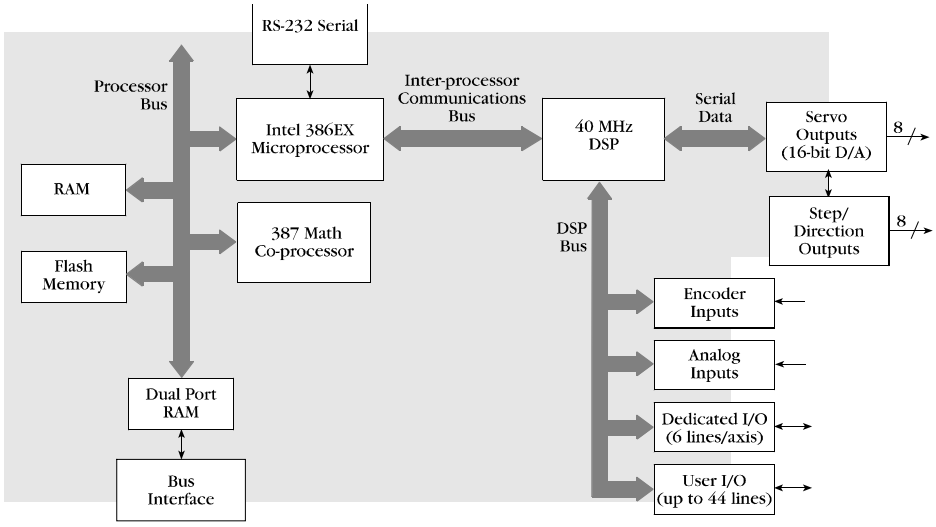

Figure 3-1. DSPpro architecture - dual-processor motion control

For maximum design flexibility, motion control programs can be created on a standard PC, using off-the-shelf C programming tools (Microsoft, Borland, etc.). Compiled programs can be downloaded to the DSPpro and stored in on-board flash memory.

With embedded DOS running on the Intel 386EX, standard programs can execute on the DSPpro, completely independent of the host processor.

The DSP section of the controller uses an Analog Devices 40MHz DSP for real-time calculations. The DSP handles all servo loop calculations, command position trajectory calculations, frame buffer execution, response to programmable software limits, hardware limits, plus many other functions.

A complete function library with source code is provided with the DSPpro controller. The library functions manipulate and conceal the internal details of the controller, allowing the programmer to concentrate on the application.

3.1.1 Program Storage

DSPpro controllers offer 1 Mbyte of DRAM and 1 Mbyte of non-volatile flash memory to store multiple motion control programs for execution as needed.

3.2 Firmware Execution

After the firmware is loaded, the DSP constantly executes the following series:

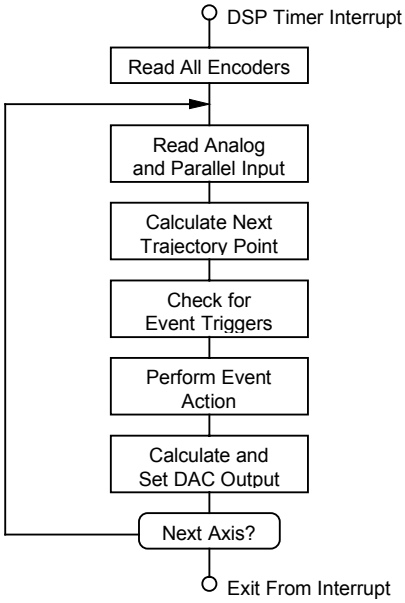

Figure 3-2. Main Loop

3.2.1 Read All Encoders

Immediately after the DSP's timer interrupt (time interval is determined by the sample rate), the DSP reads all eight encoder inputs and stores them into memory for future use. Then the DSP runs a series of events for each axis before the next timer interrupt.

3.2.2 Read Analog and Parallel Input

If an axis is configured to read an analog input, the DSP writes a control word to the A/D, waits for the conversion, reads the 12-bit digital value, and stores its value in memory. If an axis is configured to read the parallel input, the DSP reads the 32 user I/O bits and stores its value in memory.

3.2.3 Calculate Next Trajectory Point

The DSP calculates an axis' command position trajectory. It maintains a command jerk, command acceleration, command velocity, and command position:

Tn = Tn-1 + FS	Tn is the time at sample n
An = An-1 + FS * J	J is the jerk

Vn = Vn-1 + FS * An	An is the acceleration at sample n
Xn = Xn-1 + FS * Vn	Vn is the velocity at sample n Xn is the command position at sample n FS is the feed speed

The command velocity is the rate of change of the command position, acceleration is the rate of change of the command velocity, and jerk is the rate of change of the command acceleration.

3.2.4 Check for Event Triggers

The DSP checks an axis' positive limit input, negative limit input, home input, amp fault input, and software position limits. Also, the DSP checks the axis for time limits, position triggers, and I/O triggers to determine if a new frame is to be executed.

3.2.5 Perform Event Actions

If an event trigger exists for an axis, the DSP performs the associated event. The possible events are: Abort Event (highest priority), E-Stop Event, Stop Event, or New Frame Event (lowest priority).

3.2.6 Calculate and Set DAC Output

The DSP calculates an axis' output (analog voltage or pulse rate) based on a PID servo control algorithm. The input to the PID algorithm is the current position error. The current position error equals the difference between the command position and the actual position. The actual position is controlled by the feedback device, and command position is controlled by the trajectory calculator. The PID algorithm is based on the following formula:

$$On = K_R (Kp*En + Kd*(En - En-1) + Ki*Sn + Kv*Vn + 64*Ka*An + Kf * Mn) + Ko$$

The subscripted n represents the sample period. The terms are defined:

Sn = Sn-1 + En if -Smax < Sn < Smax
Smax if Sn > Smax
-Smax if Sn < -Smax

On = DAC output	K_R = overall scale factor
Kp = proportional gain	Kd = derivative gain
Ki = integral gain	Kv = velocity feed-forward
Ka = acceleration feed-forward	Ko = static DAC offset
Kf = friction feed-forward	En = position error
Mn = 0 or 1 based on the command velocity	Vn = command velocity
An = command acceleration	Sn = integrated error
Smax = maximum integrated error	

The error (En) is the basis for changes in control voltage (On). To review how each element of the PID algorithm affects motion, refer to appendix C, “Tuning Your System.” The same PID algorithm is used for open-loop and closed-loop servo and step motor control. This feature makes the programming the same for both motor types.

3.3 Hardware Features

3.3.1 Step Motor Control via VFCs

Step motors are controlled via the analog control voltage. The analog control voltage (DAC) is connected to a voltage to frequency converter (VFC). The VFC generates a pulse train directly proportional to the input voltage. The relationship between the analog control voltage and the pulse output rate is constant and linear over the entire frequency range.

This is a major performance enhancement over timer/divider pulse generators that have exponential pulse output rates and will cause large changes in step rate at high frequencies. Since a step motor's torque curve is inversely related to its speed, it is more susceptible to stalling at high speeds. Thus, a VFC at high pulse rates is less likely to stall a step motor. Another advantage of the VFC is the high pulse output resolution.

The step pulse output supports the following speed ranges:

Slow	0 to 23 kHz
Medium	0 to 94 kHz
Fast	0 to 375 kHz

The voltage level at which steps are produced by the VFCs is determined by the internal DAC offset set by the CONFIG program. This offset is usually required so that the DAC’s output zero agrees with the VFC’s input zero. If the analog zero volt output is above the VFC zero input, the VFC will begin to output step pulses.

The firmware guarantees the VFC will not output steps when command velocity is zero and the position error is zero. If the offset is set too high, the firmware will automatically “shut down” the step pulse output at the end of a move. If the offset was set too low, a small delay would occur between the time the command position changes and the step motor actually starts moving. The delay would be minimal, rarely exceeding two sample periods and would be based on the acceleration and velocity figures.

One great design advantage to using VFCs based on servo DAC outputs is the same programming is used to control either step or servo motors. If no motors are connected to the DSPpro-Series controller, you can simulate motors by configuring the axes as open-loop steppers.

3.3.2 Communication via three words in I/O Space

On a hardware level, the controller is a small microcomputer unto itself. It has its own address and data bus that the DSP, its data memory, and peripherals (dedicated and non-dedicated I/O, analog inputs and outputs, encoder inputs, and timer) use for communication. The 386EX has

access to the address and data busses of the DSP controller through three words in its I/O space. The 386EX and the DSP do not communicate directly. Instead, messages are passed through the DSP's external data memory. It is like a chalkboard, where we communicate by reading or writing messages, instead of directly modifying DSP registers.

External data memory is mostly dedicated to the frame buffer containing information about changes in an axis' current trajectory. For example, a constant velocity move will have an acceleration frame, followed by a slowing or constant velocity frame. These two frames are downloaded into the buffer for DSP execution.

The library's job is to simplify this interface. While we gain immense flexibility in axis trajectory manipulation, most applications only require simple trapezoidal-profile or similar motion. Rather than being concerned with the low-level details of frames, the programmer need only be concerned with library functions controlling trapezoidal profile or any other type of motion. These functions translate trapezoidal or other motion profiles into frames for you.

Chapter 4

HARDWARE INSTALLATION

4.0 Memory Addressing (DSPpro-VME)

Board-level DSPpro controllers communicate with applications running on the host by sharing certain memory locations in the host computer's memory. This technique is known as memory-mapping. A designated window of memory is set aside for this purpose. This window is set using on-board dip switches, which are described in detail in the following sections.

4.1 Installing the DSPpro-VME

The DSPpro-VME is a board-level controller designed for VME compatible computers running a variety of operating systems. It occupies a single 6U VME expansion slot. Before installing the DSPpro-VME, you must first set the memory location and the IRQ level.

4.1.1 DSPpro-VME Dip Switch Locations

The DSPpro-VME will respond to memory addresses in A24 space. Two banks of switches on the DSPpro-VME set the memory address of the controller. These are located next to the P1 and P2 bus connectors, and are labeled **SW1** and **SW2**.

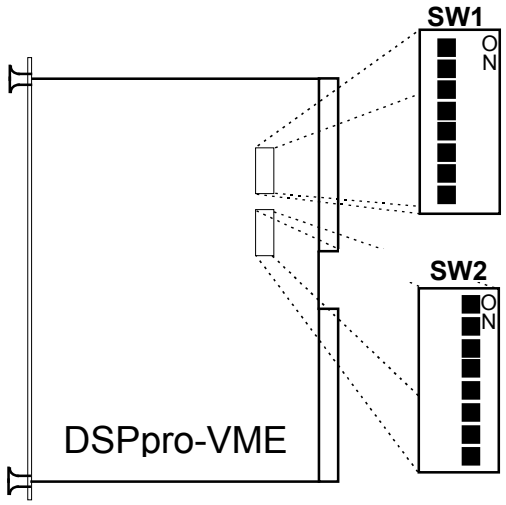

Figure 4-2. DSPpro-VME Dip Switch Locations

4.1.2 DSPpro-VME Base Memory Address Switch Settings

Memory Location	*SW1-1*	*SW1-2*	*SW1-3*	*SW1-4*	*SW1-5*	*SW1-6*	*SW1-7*	*SW1-8*	*SW2-1*
0x00000000	ON	ON	ON	ON	ON	ON	ON	ON	ON
0x00008000	ON	ON	ON	ON	ON	ON	ON	ON	OFF
0x00010000	ON	ON	ON	ON	ON	OFF	ON	OFF	ON
0x00018000	ON	ON	ON	ON	ON	ON	ON	OFF	OFF
0x00020000	ON	ON	ON	ON	OFF	ON	ON	ON	ON
0x00028000	ON	ON	ON	ON	OFF	ON	ON	ON	OFF

** default setting*

4.1.3 DSPpro-VME Interrupt (IRQ) Switch Settings.

Interrupts may be generated from the DSPpro-VME controller to the host CPU. To use one of the IRQ lines, interrupt switch SW2 must be configured. To select an IRQ line, turn ON the corresponding switch while leaving the other switches off.

Be sure to never turn more than one IRQ line ON at any one time, as this may result in erratic system behavior.

IRQ Switch SW2							
IRQ	8	7	6	5	4	3	2
None*	OFF	OFF	OFF	OFF	OFF	OFF	OFF
IRQ1	OFF	OFF	OFF	OFF	OFF	OFF	ON
IRQ2	OFF	OFF	OFF	OFF	OFF	ON	OFF
IRQ3	OFF	OFF	OFF	OFF	ON	OFF	OFF
IRQ4	OFF	OFF	OFF	ON	OFF	OFF	OFF
IRQ5	OFF	OFF	ON	OFF	OFF	OFF	OFF
IRQ6	OFF	ON	OFF	OFF	OFF	OFF	OFF
IRQ7	ON	OFF	OFF	OFF	OFF	OFF	OFF

(*) denotes default setting.

4.1.4 DSPpro-VME Board Installation Procedures

Use any available slot to install the controller board as follows:

1. Turn off the power to your computer.
2. Select an unused slot.

3. Install all required ribbon cables. Note that the non-strain-relieved end of the MEI-supplied cables should be inserted into the board. The strain-relieved end fits into the locks on the STC modules. Refer to Chapters 6 and 7 for more details.
4. Insert the card and press firmly until the board is seated in the backplane connector.
5. Fasten the mounting screws on both ends.

4.2 Installing the DSPpro-Serial

The DSPpro-Serial is a standalone motion controller that communicates with the host computer via an RS-232 interface. In order for the host computer to communicate with the controller, it must be properly connected to the Console and COM ports on the DSPpro. The Console connects to the host for booting, etc., and the COM is for the application developer to use if desired.

To install:

1. Connect one end of the MEI cable CBL-D9 CONSOLE to the CONSOLE port on the DSPpro.
2. Connect the other end of the MEI cable CBL-D9 CONSOLE to an available serial port on the host computer (e.g. COM1).
3. Connect one end of MEI cable CBL-D9 COM to the COM port on the DSPpro.
4. Connect the other end of the MEI cable CBL-D9 COM to an available serial port on the host computer (e.g. COM2).

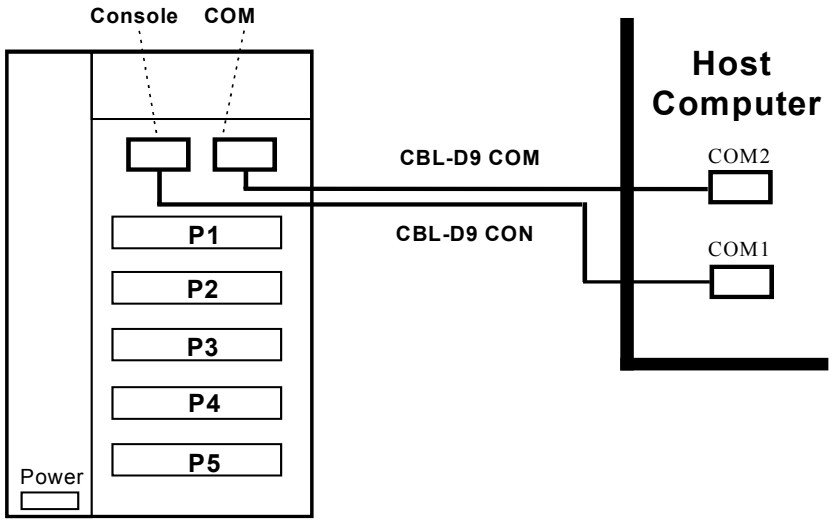

Figure 4-2. DSPpro-Serial external connections

Chapter 5

MEI SOFTWARE

DSPpro motion controllers are available with software support packages for many popular operating systems including DOS, Windows 3.x, Windows95, and Windows NT. In addition, limited support for other development environments is available. Contact your MEI representative for more information.

Refer to the release notes which accompany your software support package for information on installing the software distribution.

5.1 The SETUP Program (DSETUP.EXE & SSETUP.EXE)

5.1.1 Overview

The SETUP program is a powerful tool for installation, configuration, tuning and debugging for PC-based architectures running DOS, Windows 3.x, and Windows 95. The main screen has pull-down menus that are used to access different windows. Many windows can be accessed and arranged on the screen at one time. Each window will enable you to see and manipulate the command position, actual position, dedicated I/O, software limits, axis status, axis state, source of an event, etc. for each axis.

We recommend that you use the SETUP program to thoroughly test the hardware. Make sure you can perform two-point motion (using repeat) with all of your motors before you write any code. If you do not have motors connected to the board, you can simulate motors by configuring the axes as open-loop steps (uni-polar).

The SETUP program executes on the **host PC's CPU**, not the on-board 386EX. This is an important distinction. At the time that the controller is shipped from the factory, the flash memory contains a small program called SERVER and an AUTOEXEC.BAT file that calls this program at startup. SERVER executes on the DSPpro's 386EX, and connects to the SETUP program running on the host PC.

Follow these instructions to execute the SETUP program.

5.1.2 Using SETUP with the DSPpro

To use SETUP with the DSPpro-PC, make sure the SERVER program is running on the DSPpro-PC's CPU. If you haven't modified the contents of the DSPpro-PC flash memory since shipment from the factory, **skip to step 5 below**.

1. Insert the disk labeled *DSPpro Quick Start Disk* into the floppy disk drive of your PC.

2. If you are using the **DSPpro-Serial or DSPpro-VME**, execute the **REMSVR** program by typing `REMSVR` from a DOS prompt of your host PC. If the DSPpro console cable is connected to COM2 of your PC, execute REMSVR by typing `REMSVR /2`. You will see the following output in the REMSVR window of your PC:

```
Motion Engineering Console/Disk Server V1.0
Copyright (C) 1996 Motion Engineering, Inc.
Press Alt-X to exit
Connecting Through COM1
```

Reset the controller by depressing the reset button. Use a paper-clip to reach the recessed reset button on the DSPpro-Serial.

If you are using the **DSPpro-PC**, execute the **RCONSOLE** program by typing `RCONSOLE /r` from a DOS prompt of your host PC. The `/r` switch reboots the controller before starting RCONSOLE. You will see the following output in the RCONSOLE window of your PC:

```
RCONSOLE v1.0  (c) 1996 Motion Engineering, Inc.
```

3. The DSPpro will then reboot from the *DSPpro Quick Start Disk* in the floppy drive of your host PC. You will see the following output in the REMSVR or RCONSOLE window of your PC:

```
Motion Engineering 80C386EX BIOS V1.0
Copyright (C) 1992-1995 Motion Engineering, Inc.
Copyright (C) 1992-1995 General Software, Inc.
00896 KB OK

Embedded BIOS ROM Disk Enabled
Embedded BIOS REMOTE Disk [COM1] [CONNECTED]

Starting MS-DOS…

Server 1.00A Oct 11 1996 14:10:40
Listening on COM1.
```

4. Exit from RCONSOLE or REMSVR by typing ALT-X.

5. Execute the SETUP program by typing `ssetup` from a DOS prompt on your host PC if you are connecting to a DSPpro-VME or DSPpro-Serial, or `dsetup` if you are connecting to a DSPpro-PC. Note that if you are connecting to a DSPpro-VME or DSPpro-Serial using COM2 of your host PC, be sure to use `ssetup /2` to indicate to the SETUP program to connect through COM2.

Mouse and Trackball Support

A mouse or trackball makes the SETUP program much easier to use. A mouse can be used to click on the "hot keys" listed across the bottom of the screen, or can be used to select which window is active when several are displayed on the screen. It can also be used to move windows by positioning the mouse on the title of the window and holding down the left button.

Be sure to load your mouse driver before running the SETUP program.

Hot Keys

If you do not have a mouse or trackball, the keyboard can be used to perform the same tasks. The function keys are listed below.

Hot Key	*Description*
Space Bar or <Enter>	Select the highlighted button
F2	Open a Position Status Window
F3	Open an Axis Status Window
F5	Move the current window (with the cursor keys)
F6	Jump to the next open window
F7	Open Tuning Parameters Window
F8	Open Axis Configuration Window
F9	DSP hardware reset
<ESC>	Close the current window
Alt/F	Select the File menu
Alt/C	Select the Configure menu
Alt/S	Select the Status menu
Alt/M	Select the Motion menu
Alt/X	Exit the SETUP program
Cursor Keys	Move between fields and buttons

Buttons

In each window, there are buttons provided to send, read and save information stored in the DSP's data memory and boot memory. The buttons have the following functions:

Button	*Description*
Send	Same as the <ENTER> key. Write the values in the window into data (volatile) memory on the board.
Set axis	Same as the <ENTER> key. Set the axis to display the current values in data memory.
Save	Store the window values into DSPpro boot (non-volatile) memory.
Read	Bring values from boot (non-volatile) memory into data (volatile) memory.
Copy all	Copy the values in the window to data (volatile) memory all axes. Values displayed in other windows are not affected.
Save all	Store the values in the window to boot (non-volatile) memory for all axes. Values displayed in other windows are not affected.

Saving Default Parameters On-Board

Many of the configuration windows have **Read** and **Save** buttons. The **Read** button loads the default (power-up or reset) configuration parameters from boot memory into data memory The **Save** button stores the current parameters into boot memory.

The SETUP program can access the data memory (volatile) and boot memory (non-volatile). When a value is entered in any window, the value is automatically stored in the DSP data

memory. The values stored in data memory are lost when the DSPpro is reset (F9 key) or the power is turned off.

The reset function (F9) loads the firmware and configuration parameters from boot memory into data memory. Values stored in boot memory will be read by the SETUP program on initialization.

Selecting "**Save Defaults to File**" saves the current boot memory configuration to a disk file with the extension .ABS. Selecting "**Load Defaults From Disk**" will load the values from a disk file into the boot memory on board.

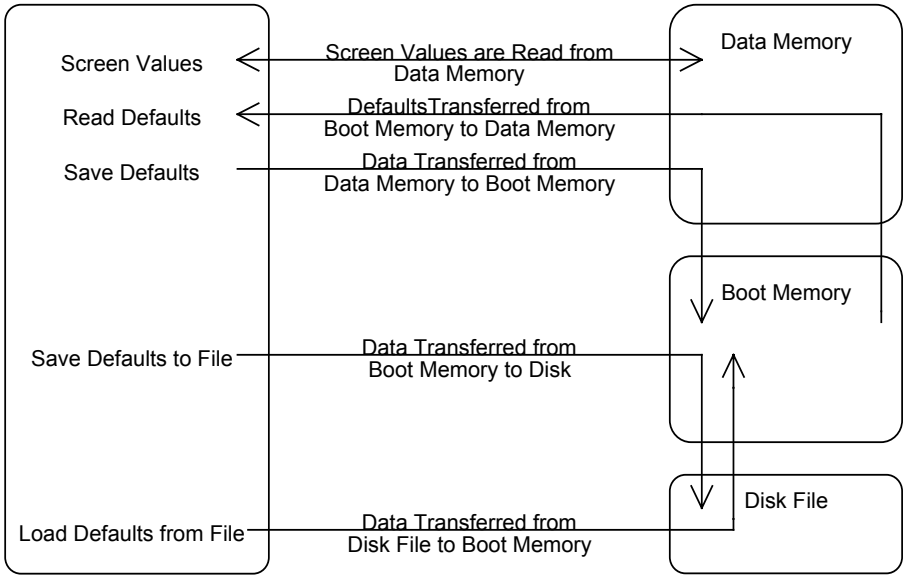

Figure 5-1. SETUP Parameter Storage

Functional Grouping by Axis

Some of the functions and parameters of the board must be the same across groups of axes:

Function	*Axes in Group*	*Example*
Step or Servo Motor	2	A 3 axis board is to be used for two steps and one servo, the servo must be axis 2 and the steps axes 0 and 1. When a pair of axes (2 and 3 in this case, even though axis 3 is not present) are configured as a servo axes, the step pulse output is turned off for both axes 2 and 3.
Open-Loop or Closed-Loop	2	A 3 axis board is to be used for two closed-loop steps and one open-loop step, the open-loop motor must be axis 2 and the closed-loop motors axes 0 and 1.
Home and Index functions	4	On a 4 axis board, all axes will be configured to use the Home and Index in the same fashion. On a 7 axis board, axes 0-3 will have a configuration that is independant of the configuration for axes 4-6.

5.1.3 SETUP Screens

The SETUP screens are organized under four main menu categories:

File:
- Load default parameters from a disk file
- Store default parameters to a disk file
- Shell out to DOS
- Display version number
- Move location of selected window
- Jump to another window
- Exit program

Configure:
- Set I/O address
- Set PID tuning parameters
- Set auxilliary tuning parameters
- Set axis configuration
  - Servo or step
  - Open/closed loop
  - Stepping speed
  - Home sensor configuration
  - Voltage output
  - Feedback device type
  - Integration active mode
- Set limit switch configuration
- Set software limits
- Reset controller with boot memory

Status:
- Monitor position status:
  - Position, Velocity, Acceleration and Error
- Monitor axis status:
  - Idle/Run Mode, In Motion, In-Position, Source and State
- Monitor Dedicated I/O Status
  - Enable/Disable Amplifier

Motion:
- Two-point motion
  - Endpoints, Delay, Velocity, Acceleration and Jerk
  - Motion Profile: Trapezoidal, Parabolic or S-Curve

5.1.3.1 File Menu

The **File** menu contains the options described in detail below:

Option	*Description*
Load Defaults from File	Loads values from disk into DSP boot memory (requests filename)
Save Defaults to File	Saves values in DSP boot memory to disk (prompts for filename)
DOS Shell	Shells to DOS
About	Displays the program version number
Exit	Terminates the program

Load Defaults from File

Selecting Load Defaults from **File** will read a disk file containing the firmware, which includes the parameters for the PID filter, limit switch configurations, software limit configurations, etc. The filename may be selected when the prompt appears.

Save Defaults to File

Selecting Save Defaults to File will write a disk file containing the firmware, PID parameters, limit switch configurations, software limit configurations, etc. The filename may be selected when the prompt appears. It is recommended that you store your configuration into a file after configuring the board . Once stored on disk, the parameters can be easily downloaded to the board in the future if necessary.

DOS Shell

This selection allows users to access the DOS command line without exiting the SETUP program. Typing EXIT at the DOS prompt will return control to the SETUP program.

About

Displays the SETUP version number and date.

Exit

Exits the SETUP program. Note that on exit, motion will stop, but all configuration parameters remain active.

5.1.3.2 Configure Menu

The **Configure** menu contains the options:

Option	*Description*
I/O Base Address	Sets the I/O address where SETUP communicates with the board
Tuning Parameters	Sets tuning parameters, DC offset and voltage/pulse rate limit
Aux. Tuning Parameters	Sets auxilliary tuning parameters: derivative sample rate, etc.
Axis Configuration	Allows axes to be configured as step/servo, etc.
Limit Switch Configuration	Sets the active level of limit switches and associated action
Software Limits	Sets the software limits and associated actions
Reset	Resets the DSP with parameters stored in battery backed RAM

I/O Base Address Window

This window is used to set the base address for the board. If SW1 is set for an address other than 300 hex, this window must be used to tell the SETUP program the location of the board.

The SETUP program also has the ability to read an environment variable called 'DSP' and automatically set the base address. The only understood parameter currently is 'BASE', which is used to specify the base I/O address of the controller. If BASE is specified, then the SETUP program will initialize the controller using the 'BASE' address. For example, if 'set DSP=base: 0x280' is executed at the DOS prompt then the SETUP program will use address 280 hex.

Tuning Parameters Window

This window is used to set the control loop tuning parameters for each axis. The DSP uses a 2nd order PID algorithm with velocity and acceleration feed-forward. A more detailed explanation of each parameter is contained in appendix C, “Tuning Your System.”

Note that multiple windows can be open simultaneously. For example, windows can be open for **Tuning Parameters** and **Motion Status** for one axis, or windows can be open **for Tuning Parameters** and **Motion Status** for both Axes 1 and 4. This makes it possible to change tuning parameters on-the-fly and the effect can be seen in real-time.

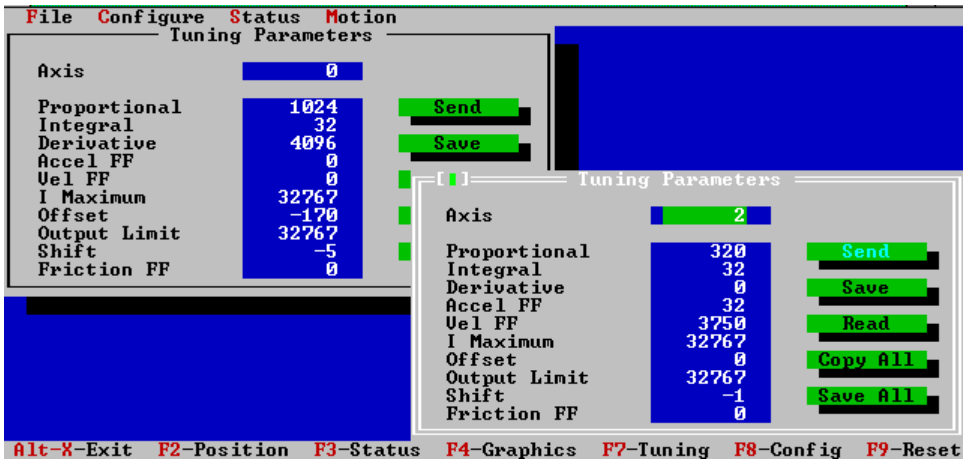

Figure 5-1. Configure/Tuning Parameter Windows for Axes 0 and 1

The PID algorithm is based on the following formula:

$$O_n = K^R (K_p*E_n + K_d*(E_n - E_{n-1}) + K_i*S_n + K_v*V_n + 64*K_a*A_n + K_f * M_n) + K_o$$

The subscripted $_n$ represents the sample period. The terms are defined as follows:

$S_n = S_{n-1} + E_n$ if $-S_{max} < S_n < S_{max}$
S_{max} if $S_n > S_{max}$
$-S_{max}$ if $Sn < -S_{max}$

O_n = DAC output
K_p = proportional gain
K_i = integral gain
K_a = acceleration feed-forward
K_f = friction feed-forward
M_n = 0 or 1 based on the command velocity
A_n = command acceleration * 2^{-6}
S_{max} = maximum integrated error
K_R = overall scale factor
K_d = derivative gain
K_v = velocity feed-forward
K_o = static DAC offset
E_n = position error
V_n = command velocity
S_n = integrated error

Proportional Gain

The proportional gain affects the analog command voltage or pulse rate based on the amount of position error. The higher the proportional gain, the "stiffer" the response. If the proportional gain is set too low, the response will be "mushy" - the motor will have trouble following the commanded trajectory.

If the proportional gain is set too high, the motor may oscillate or "buzz" at rest or during motion. The range of values for proportional gain is 0 to 32,767.

See appendix C, “Tuning Your System,” for more information on setting proportional gain.

Integral Gain

The integral gain parameter is used to integrate static errors and "fine tune" the position at rest. The motor command (analog voltage or pulse rate) will increase with increasing error and time. The maximum amount of gain due to integration is limited to prevent "windup".

With proper tuning, motor sizing and a low-friction mechanical system, 0-1 encoder count (step) error is possible. The range of values for integral gain is 0 to 32,767.

See appendix C, “Tuning Your System,” for more information on setting integral gain.

Derivative Gain

The derivative gain affects the analog command voltage or pulse rate based on the amount of position error change occurring in the last two samples. The derivative gain term acts as a damping factor. The range of values for the derivative gain is 0 to 32,767.

See appendix C, “Tuning Your System,” for more information on setting derivative gain.

Acceleration Feed Forward

The acceleration feed forward term is used to add extra output during acceleration to reduce following error. The range of values for the acceleration feed forward is 0 to 32,767.

See appendix C, “Tuning Your System,” for more information on setting acceleration feed forward.

Velocity Feed Forward

The velocity feed forward term is used to add extra output during constant velocity to reduce following error. The range of values for the velocity feed forward is 0 to 32,767.

See appendix C, “Tuning Your System,” for more information on setting velocity feed forward.

I Maximum

This parameter sets the maximum voltage output by the integration term of the PID algorithm. The limit is used to prevent "windup". Generally, "windup" occurs in systems where (very) high friction cannot be overcome without entering an oscillation mode. The range of values for the integration limit is 0 to 32767.

See appendix C, “Tuning Your System,” for more information on setting the integration limit.

Offset

This parameter sets the DAC output level. It can be used to compensate for other system offsets. The offset parameter should be set at 0 in most cases.

Note that each axis also has an internal offset, which is in series with the digital filter offset described here and visible on the SETUP program tuning screen. The offset is used to zero the DAC and Voltage-to-Frequency converter outputs to prevent motion when the axis is placed in idle mode.

The internal offset is set by the CONFIG.EXE program. Normal DAC offset after the CONFIG program is run should be under 3 millivolts (which will not produce step pulses). Temperature drift is approximately 1 millivolt per degree C.

Also, note that only positive values of offset will output steps, since the Voltage-to-Frequency converter can only react to positive voltages. The range of values for the offset is +/- 32,767.

See appendix C, "Tuning Your System," for more information on the offset.

Output Limit

This parameter is used to limit the controller output (analog voltage or pulse rate) during system tuning. For servo motors, this term limits the analog output voltage. For step motors, this term limits the step pulse output rate. The range for the output limit is 0 to 32,767. This range corresponds to -10 volts to +10 volts for servos (i.e. 0.00305 volts/unit) or 0 to full scale pulse rate for steps.

Shift Range

This parameter is used to shift the range of the tuning parameters. The shift factor multiplies or divides all the filter parameters by a user specified power of two. For example, if the shift factor = -3, then all the filter parameters will be divided by 8 ($2^{-3} = 1/8$). If the shift factor = 2, then all the parameters will be multiplied by 4. This is useful for unusual motors such as air-bearing motors, voice-coil actuators and hydraulics or other actuators. The default parameter is -5, i.e. a multiplier of 2^{-5} or 1/32.

Friction Feed Forward

The friction feed forward term is used to add extra output during any commanded velocity to reduce following error caused by friction. The range of values for the friction feed forward are 0 to 32,767.

See appendix C, "Tuning Your System," for more information on setting friction feed forward

Axis Configuration Window

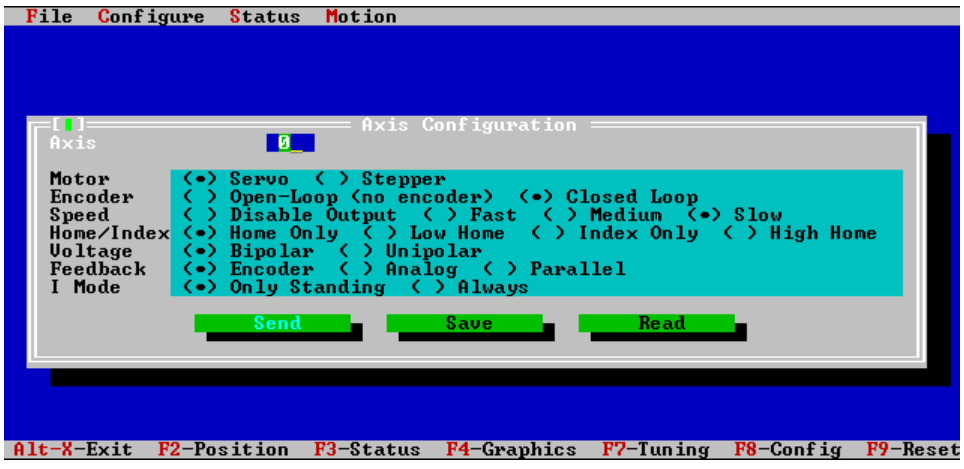

Figure 5-2. The Configure/Axis Configuration Window

Servo or Step Motor

This selection is used to enable or disable the step pulse output for a given pair of axes. Selecting "step" will enable the step output for the pair of axes (0 and 1, 2 and 3, etc.). The analog output is available regardless of the selection. When the motor type is changed, a set of default tuning parameters will be loaded into SETUP for that axis.

Open-loop or closed-loop

This selection allows users to indicate if the pair of axes is to be open-loop or closed-loop. If closed-loop is selected, the board will use feedback from an external device to close the loop. If step and open-loop are selected, the board will direct the step output back into the encoder input for the axis, in effect digitally closing the loop on-board:

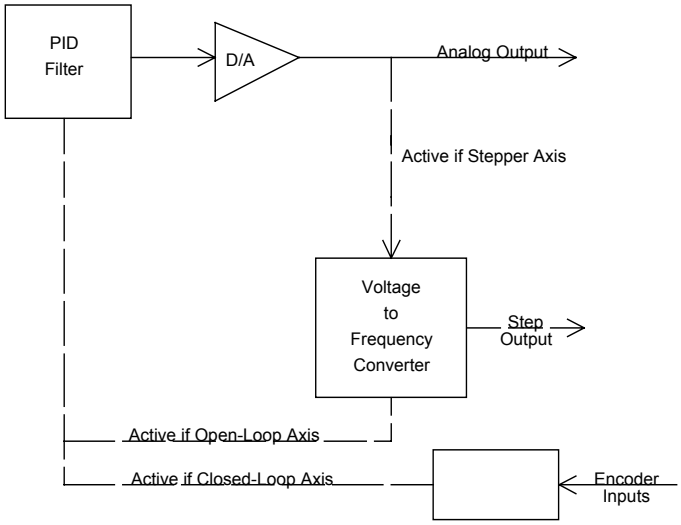

Figure 6-4. Internal Architecture to Control Step Motors

Speed

This selection sets the maximum pulse rate for the step output in either open-loop or closed-loop mode. Whenever step is selected, the step speed range must be set. The ranges are:

Slow	*Medium*	*Fast*
0 to 23 kHz	0 to 94 kHz	0 to 375 kHz

You must set the tuning parameters as follows for each axis configured for open-loop steps:

Parameter	*Setting*
Proportional (K_p)	320
Integral (K_i)	32
Derivative (K_d)	0
Accel FF (K_a)	32
Vel FF (K_v)	3750
I Maximum	32767
Offset	0
Ouput Limit	32767
Shift-	-1 (Slow), -3 (Medium), -5 (Fast)

We recommend choosing the slowest possible speed range that is adequate for your system. If an axis is configured for servo, the speed selection should be: **Disable Output**.

Home

This selection configures whether the index pulse is required for the home input to be active. Typically a rotary encoder has a single index pulse (per revolution). The index pulse can be used with the home signal input to produce accurate homing to within one encoder count. Standard boards have four possible types of homing:

Type	*Description*
Home Only	Home input only (active high or active low)
Low Home and Index	Home input ANDed with index (active low home and active high index)
Index Only	Index only (active high or active low)
High Home and Index	Home input ANDed with index (active high home and active high index)

Note that the home/index setting affects the axes in groups of 4. For example, on a 4-axis board, all the axes must be configured the same with respect to home/index. For more information, see the section on home switch wiring.

Bipolar/Unipolar

This selection configures whether the analog output is unipolar (0 to +10 volts) or bipolar(-10 volts to +10 volts). When using analog servo motors, the output should be configured for bipolar operation. When using steps, the output should be configured for unipolar operation.

Trouble Tip - Motors Turn Only One Direction:

***Steppers**: If a step motor turns only one direction, check the Configuration/Axis Configuration window to be sure the axis is set for UNIPOLAR. The voltage-to-frequency-converter only responds to positive voltages (unipolar) and will not output steps if the voltage is negative (bipolar).*

***Servos**: If a servo motor turns only one direction, check the Configuration/Axis configuration window to be sure the axis is set for BIPOLAR.*

Feedback Device

This selection allows each axis to be configured for the type of feedback device used. The choices are:

Selection	*Device Type*	*Pin Location*
Encoder	Incremental Encoder	Motor Signal Header
Analog	Unipolar LVDT	Analog Input Header
Parallel	Laser Interferometer	User I/O Headers

Note that each axis can be individually configured with any feedback device, but if any axis uses analog inputs, the remaining analog inputs cannot be used for any purpose other than analog feedback. For example, the analog inputs cannot be used for both a joystick and analog feedback on the same board.

Integration Mode

This selection allows the PID integration term for each axis to be configured as:

Only Standing	Only when the command velocity is zero
Always	During motion and when standing

Limit Switch Configuration Window

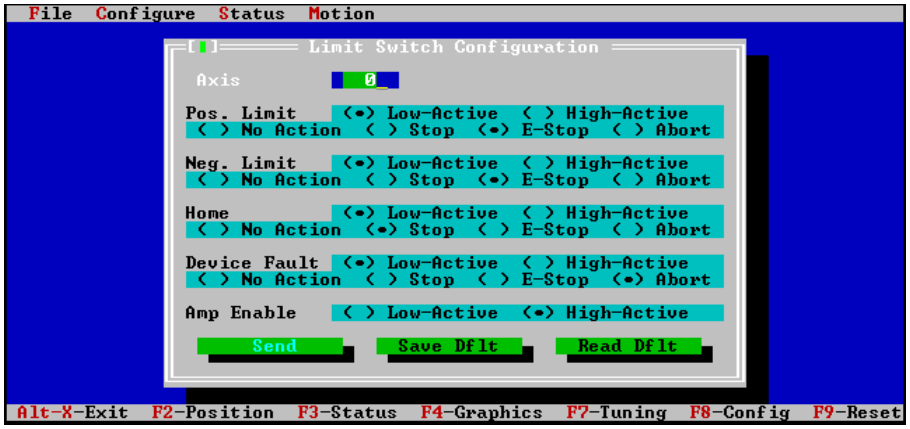

Figure 5-3. Configure/Limit Switch Configuration Window

This window defines the active state of the home switch, limit switches, device fault and the amp enable output. It also specifies which event is triggered when each sensor becomes active. The events are:

Event	*Description*
NO_EVENT	Ignore a condition
STOP_EVENT	Decelerate to a stop (at specified stop rate).
E_STOP_EVENT	Decelerate to a stop (at specified E-stop rate).
ABORT_EVENT	Disable PID control and the amplifier for this axis.

Software Limits Window

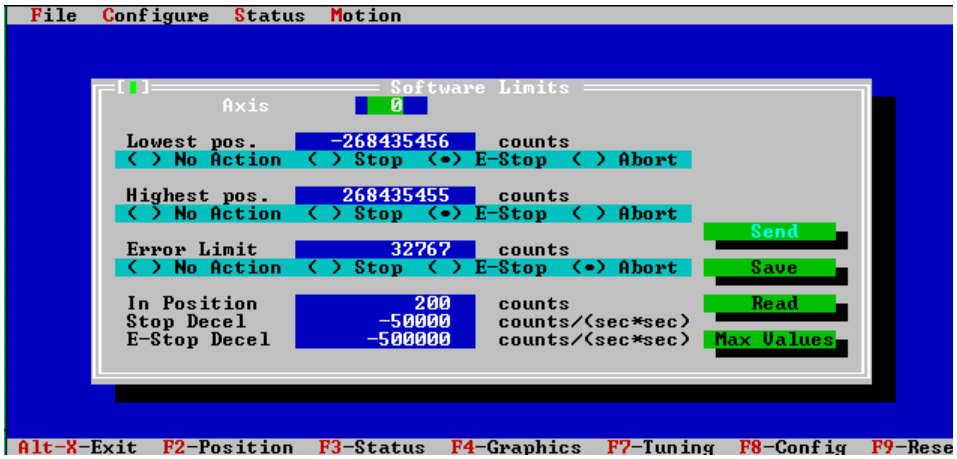

Figure 5-4. Configure/Software Limit Configuration Window

This menu is used to set the software limits (lowest position, highest position and error limit) for each axis. The values for each of these limits and the event to be performed when the limit is exceeded can be specified. The events are:

Event	*Description*
NO_EVENT	Ignore a condition
STOP_EVENT	Decelerate to a stop (at specified stop rate).
E_STOP_EVENT	Decelerate to a stop (at specified E-stop rate).
ABORT_EVENT	Disable PID control and the amplifier for this axis.

Reset

This selection will perform a power-up reset of the DSP controller. The software and hardware configurations are re-read from boot memory, the command and actual positions are reset, the amp enable output is disabled, User I/O is reconfigured, etc. A hardware reset causes the DSP to release control of the axes and I/O for a few milliseconds which may cause motors to jump.

5.1.3.3 Status Menu

The **Status** menu contains the options:

Option	*Description*
Position Status	Displays the position, velocity and acceleration of each axis
Axis Status	Displays the status of each axis: Motion, E-Stop, Run/Idle, etc.
Dedicated I/O	Display the status of dedicated I/O lines

Position Status Window

The **Position Status** window is a read-only window which provides an easy way to monitor the status of each axis. The Clear button will immediately zero actual and command positions.

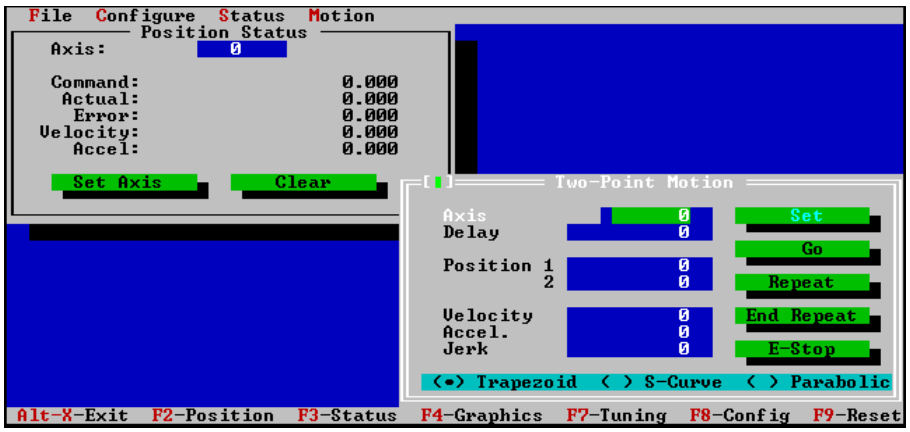

Figure 5-5. Status/Position Status Window

Axis Status Window

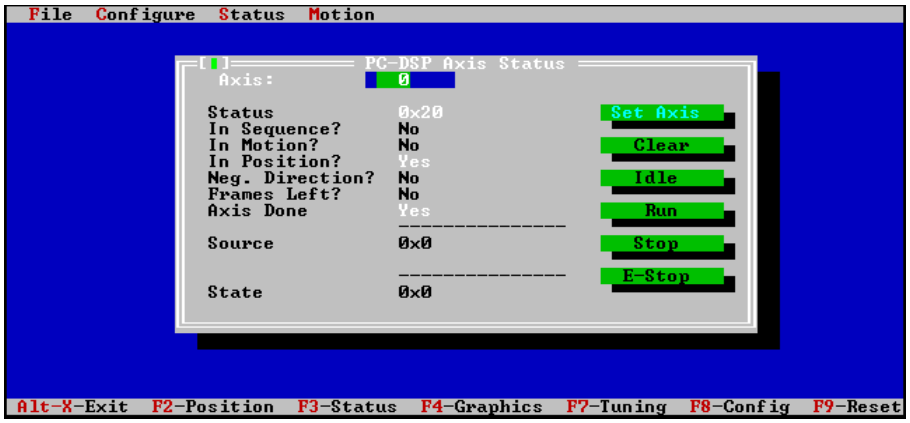

Figure 5-6. Status/Axis Status Window

This window displays the real-time status of the flags for the axis displayed. In the following description the term "Event" means Stop, E-Stop, or Abort. The status items reported are:

Item	*Description*
Status	Displays the current condition of an axis in hex.
In Sequence?	Displays "Yes" if a set of frames describing a move is executing.
In Motion?	Displays "Yes" if command velocity is non-zero.
In Position?	Displays "Yes" if the position is within the in-position window.
Negative Direction?	Displays "Yes" if the command velocity is negative.
Frames Left?	Displays "Yes" if additional, unexecuted frames are still in buffer.
Axis Done?	Displays "Yes" if In Motion? is "No" and In Position? is "Yes".
Source	Displays the source of a current event (host CPU, position limit, etc.).
State	Displays current event on an axis (Running, E-Stop, etc.).

The buttons on the right of the window perform the following functions:

Button	*Description*
Clear	Reset all flags, clear stops and E-stops.
Idle	Set analog and step/direction outputs to zero, disables amp enable output and disables PID filter.
Run	Closes the loop, enables PID filter. Note that loop is closed internally (on- board) for open-loop steps when Run mode is selected. Amp enable must be manually enabled.
Stop	Decelerate at Stop rate.
E-Stop	Decelerate at E-Stop rate.

Dedicated I/O Window

The dedicated I/O displays the status of the dedicated inputs for limits, home, device fault and in-position. The window also contains buttons to set the amp enable output to a high or low state.

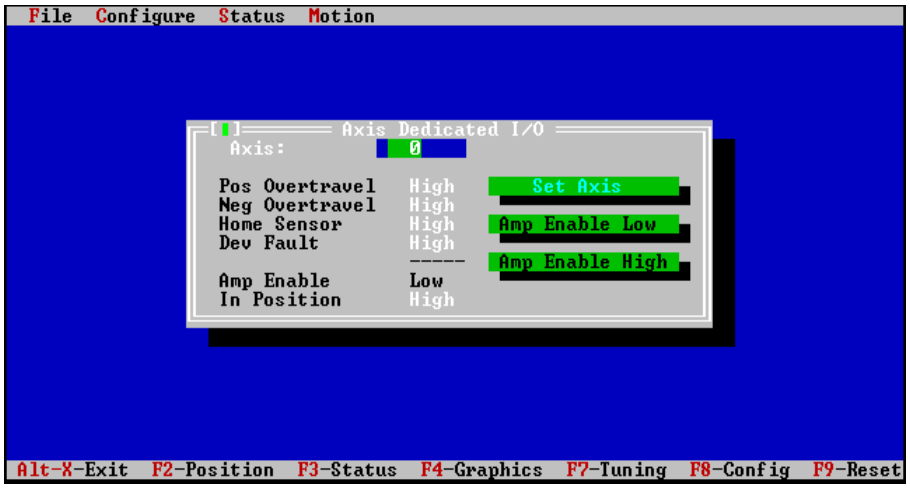

Figure 5-7. The Configure/Axis Dedicated I/O Window

5.1.3.4 Motion Menu

The **Motion** menu contains the options:

Option	*Description*
Point to Point Motion	Is used to command an axis to move between two points
Graphics Mode	Displays the command vs. actual and analog output for a move

Point to Point Motion Window

This window is used to command motion between two points. Point 1 and Point 2 specify the endpoints of the motion, Velocity specifies the maximum slew speed and Acceleration the acceleration rate. The jerk field is only used when performing non-constant acceleration profiles (S-curve and parabolic). Units are encoder counts (steps), counts (steps) per second, counts (steps) per second ^{2} and counts (steps) per second ^{3}.

The GO button is used to start the motion. The **Repeat** and **End Repeat** fields may be used to start or stop repetitive motion. The E-STOP field is used to trigger an E-Stop event. Use the cursor to move between fields and buttons, and the Space Bar or <Enter> to "push" a button.

Three motion profiles are available: trapezoidal, parabolic and S-curve. Generally, choose an acceleration that is 10 times the velocity and a jerk that is 100 times the acceleration. Remember, that the velocity is the rate of change of the position, acceleration is the rate of change of the velocity, and the jerk is the rate of change of the acceleration. Note that increasing the velocity and acceleration of parabolic and S-curve moves can actually increase the time to position.

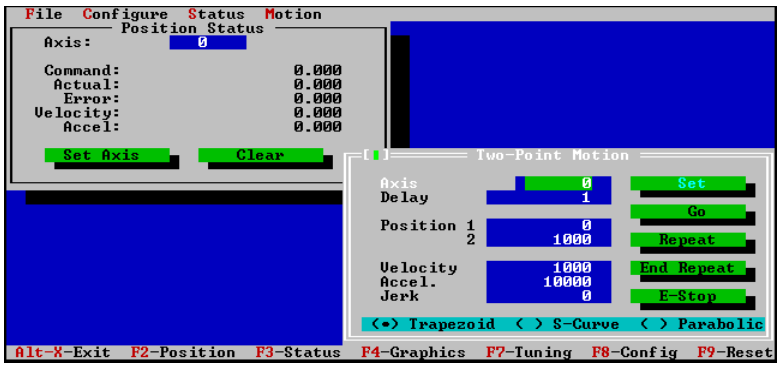

Figure 5-8. The Motion/Two-Point Motion Window (shown with Status/Position Status window)

The delay field allows motion to be paused at the endpoints. Units of delay are relative time and depend on the CPU speed of the computer.

5.2 The VERSION Program (VERSION.EXE)

The VERSION program provides information on the current revision level of the firmware that controls the motion functions of the DSP. The VERSION program must be executed on the DSPpro's CPU, and not on the host PC.

To execute VERSION on the DSPpro, follow these instructions:

1. Copy the VERSION program onto the *DSPpro Quick Start Disk*.

2. Edit the AUTOEXEC.BAT file on the *DSPpro Quick Start Disk* and place a `REM` command at the front of the line which calls SERVER.

3. If you are using the **DSPpro-Serial or DSPpro-VME**, execute the **REMSVR** program by typing `REMSVR` from a DOS prompt of your host PC. If the DSPpro console cable is connected to COM2 of your PC, execute REMSVR by typing `REMSVR /2`. You will see the following output in the REMSVR window of your PC:

```
Motion Engineering Console/Disk Server V1.0
Copyright (C) 1996 Motion Engineering, Inc.
Press Alt-X to exit
Connecting Through COM1
```

Reset the controller by depressing the reset button. You will have to use a paper-clip to reach the recessed reset button on the DSPpro-Serial.

If you are using the **DSPpro-PC**, execute the RCONSOLE program by typing `RCONSOLE /r` from a DOS prompt of your host PC. The `/r` switch reboots the controller before starting RCONSOLE. You will see the following output in the RCONSOLE window of your PC:

```
RCONSOLE v1.0  (c) 1996 Motion Engineering, Inc.
```

4. The DSPpro will then reboot, booting from the *DSPpro Quick Start Disk* in the floppy drive of your host PC. You will see the following output in the REMSVR or RCONSOLE window of your PC:

```
Motion Engineering 80C386EX BIOS V1.0
```

```
Copyright (C) 1992-1995 Motion Engineering, Inc.
Copyright (C) 1992-1995 General Software, Inc.
00896 KB OK

Embedded BIOS ROM Disk Enabled
Embedded BIOS REMOTE Disk [COM1] [CONNECTED]

Starting MS-DOS…

DSPpro >
```

5. Execute the VERSION program by tying `version` at the DSPpro command prompt. You should see output in your REMSVR or RCONSOLE window similar to the following:

```
DSP Firmware Production Version: 2.40, Revision: E1, Option:0
Board Type: 08 (PROPC), Rev. 1
FPGA Prom Version: 1.3.2
```

6. Exit from RCONSOLE or REMSVR by typing ALT-X.

7. Edit the AUTOEXEC.BAT file on the *DSPpro Quick Start Disk* to remove the `REM` statement at the front of the line which calls SERVER.

5.3 The CONFIG Program (CONFIG.EXE)

The CONFIG program is used to download new firmware to the DSPpro. The CONFIG program must be executed on the DSPpro's CPU, and not on the host PC.

To execute CONFIG on the DSPpro, follow these instructions:

1. Copy the CONFIG program and the firmware file 8AXIS.ABS onto the *DSPpro Quick Start Disk*.

2. Edit the AUTOEXEC.BAT file on the *DSPpro Quick Start Disk* and place a `REM` command at the front of the line which calls SERVER.

3. If you are using the **DSPpro-Serial or DSPpro-VME**, execute the **`REMSVR`** program by typing `REMSVR` from a DOS prompt of your host PC. If the DSPpro console cable is connected to COM2 of your PC, execute REMSVR by typing `REMSVR /2`. You will see the following output in the REMSVR window of your PC:

```
Motion Engineering Console/Disk Server V1.0
Copyright (C) 1996 Motion Engineering, Inc.
Press Alt-X to exit
Connecting Through COM1
```

Reset the controller by depressing the reset button. Use a paper-clip to reach the recessed reset button on the DSPpro-Serial.

4. The DSPpro will then reboot, booting from the *DSPpro Quick Start Disk* in the floppy drive of the host PC. You will see this output in the REMSVR or RCONSOLE window of the PC:

```
Motion Engineering 80C386EX BIOS V1.0
Copyright (C) 1992-1995 Motion Engineering, Inc.
Copyright (C) 1992-1995 General Software, Inc.
00896 KB OK

Embedded BIOS ROM Disk Enabled
```

```
Embedded BIOS REMOTE Disk [COM1] [CONNECTED]

Starting MS-DOS…

DSPpro >
```

5. Execute the `CONFIG` program by tying `config` at the DSPpro command prompt. You should see output in your REMSVR or RCONSOLE window similar to the following:

```
DSPpro >config

CONFIG version 2.0b, PC/DSP library version 2.40E.
Testing data memory.

Part 1:  Testing data memory with sequential values.
Part 2:  Testing data memory with random values.
Part 3:  Testing boot memory with linear values.
Part 4:  Testing boot memory with random values.
Loaded firmware from 8AXIS.ABS.
0 ++--: 276
::::::::::::::::::::::::
Motion Test:          -453
Motion Test Passed!
1 +--: 247
::::::::::::::::::::::::
Motion Test:          -456
Motion Test Passed!
2 +--: 241
::::::::::::::::::::::::
```

6. CONFIG will continue until all eight axes are tested. When you see the `DSPpro >` prompt, exit from RCONSOLE or REMSVR by typing ALT-X.

7. Edit the AUTOEXEC.BAT file on the *DSPpro Quick Start Disk* to remove the `REM` statement at the front of the line which calls SERVER.

Chapter 6

MOTOR WIRING

The following sections describe procedures to connect the controller to drives and motors and test the connections.

CAUTION

DO NOT CONNECT THE MOTOR SHAFT TO THE MECHANISM DURING TESTING AND INITIAL TUNING!

6.1 Connection Accessories

MEI offers screw terminal connection blocks such as the STC-20, STC-26, STC-50 and STC-D50 which can be used (and are recommended) to provide convenient screw-terminal connections to the signals. The STC modules convert the ribbon cable signals into screw terminal connections for quick and easy connections. STC's mount on a standard DIN rail.

In addition, MEI cables are available to provide a clean and reliable interface to the STC modules.

The following are suggested accessory configurations:

For DSPpro-VME:

STC-20 Connection module for analog input lines, one required per board

STC-26 Connection module for motor axes, one required for two motor axes

STC-50 Connection module for I/O lines, one required for each I/O header

CBL-20 Analog Input ribbon cable, one required per board

CBL-26 Motor axis ribbon cable, one required for every two axes

CBL-50 I/O ribbon cable, one required for each I/O header

For DSPpro-Serial:

STC-D50 Connection module for motor axes and I/O lines, one for each two axes, and one required for I/O

CBL-D50 Cable for motor axes and I/O lines, one for each two axes, and one required for I/O

All cables are electrically tested for continuity at the factory and are marked with a blue dot to indicate that they have passed.

Board configurations with connection accessories are shown below. Note that all terminal blocks include mounting feet for standard DIN rail.

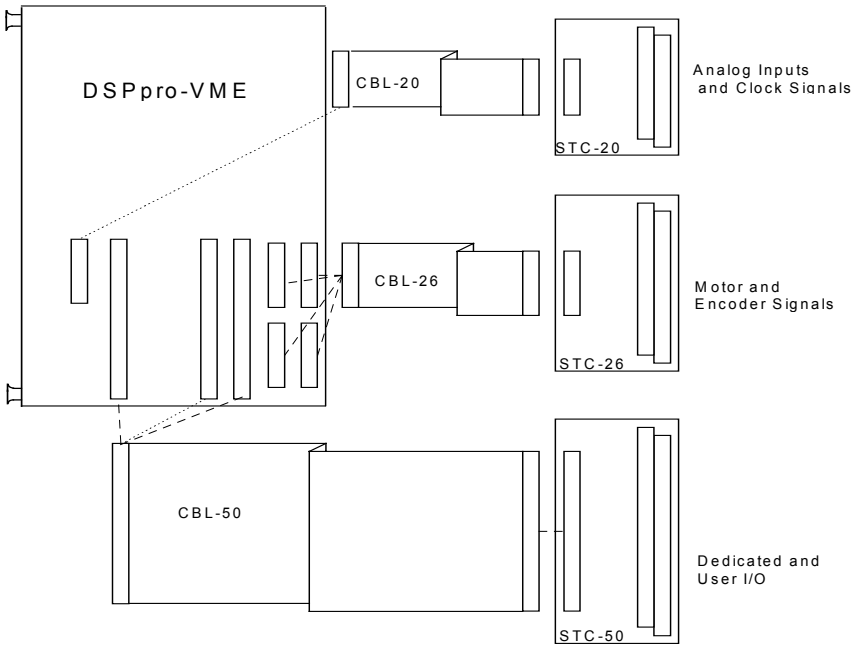

Figure 6-2. DSPpro-VME Configuration with Connection Accessories

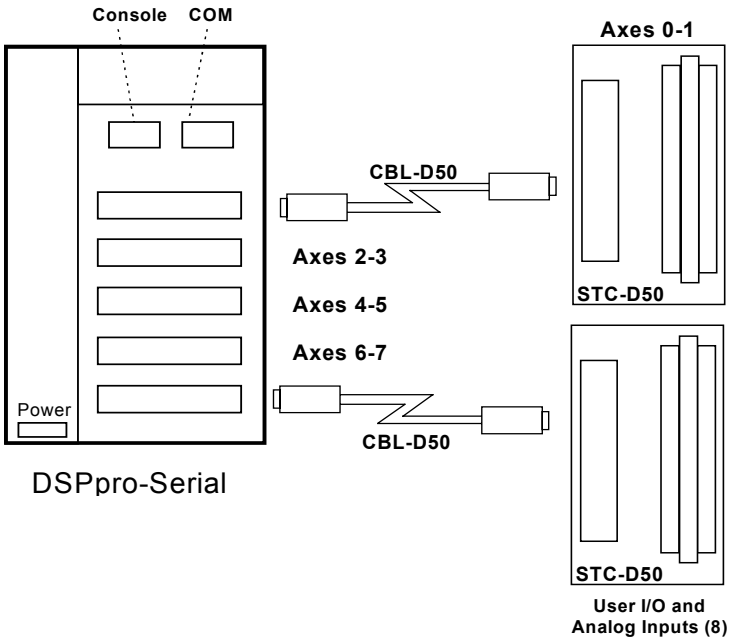

Figure 6-3. DSPpro-Serial Configuration with Connection Accessories

6.2 DSPpro-VME Motor Signal Header Locations

The motor signals for the DSPpro-VME are brought out through 26-pin box headers. There is one 26-pin box header for each pair of motor axes. The header locations and pin-outs are shown below and in appendix A at the end of this manual.

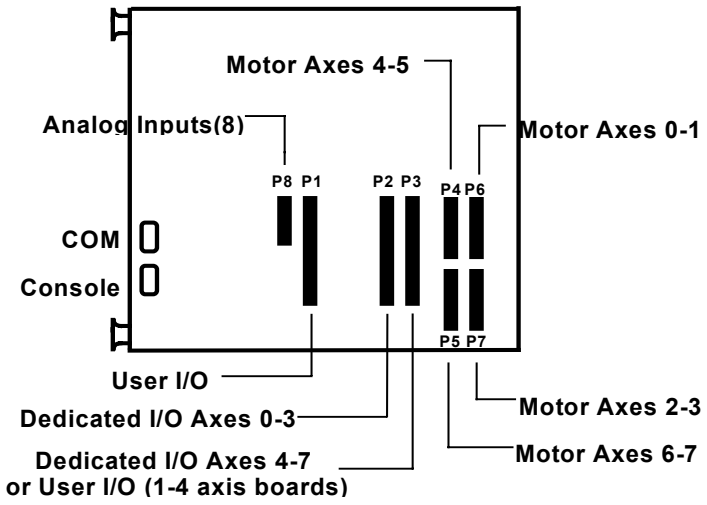

Figure 6-5. Motor Signal Header Locations - DSPpro-VME

6.2.1 DSPpro-VME Motor Signal Pinouts (P4-P7)

The table below summarizes the pinouts for the 26-pin motor signal headers (P4-P7) on the DSPpro-VME for each pair of motor axes.

Pin	Signal	Axis	Pin	Signal	Axis
1	GND	1st	14	GND	2nd
2	5V	1st	15	5V	2nd
3	Encoder A +	1st	16	Encoder A +	2nd
4	Encoder A -	1st	17	Encoder A -	2nd
5	Encoder B +	1st	18	Encoder B +	2nd
6	Encoder B -	1st	19	Encoder B -	2nd
7	Encoder Index +	1st	20	Encoder Index +	2nd
8	Encoder Index -	1st	21	Encoder Index -	2nd
9	+/- 10V Analog Out	1st	22	+/- 10V Analog Out	2nd
10	Step Pulse +	1st	23	Step Pulse +	2nd
11	Step Pulse -	1st	24	Step Pulse -	2nd
12	Step Direction +	1st	25	Step Direction +	2nd
13	Step Direction -	1st	26	Step Direction -	2nd

6.3 Motor Signal Header Locations - DSPpro-Serial

The motor signals for the DSPpro-Serial are brought out through four 50-pin DB-50 connectors. There is one DB-50 connector for each pair of motor axes. These DB-50 connectors contain both motor signal pinouts as well as dedicated I/O for each pair of axes. The header locations and motor signal pin-outs are shown below. The pinouts for the dedicated I/O are shown in chapter 7, "I/O Wiring."

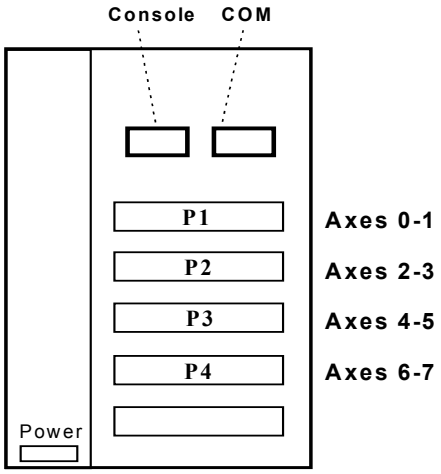

Figure 6-6. Motor Signal Header Locations - DSPpro-Serial

6.3.1 DSPpro-Serial Motor Signal Pinouts (P1-P4)

Pin	Signal	Axis	Pin	Signal	Axis
1	5V	1st	26	5V	2nd
2	GND	1st	27	GND	2nd
3	Encoder A +	1st	28	Encoder A +	2nd
4	Encoder A -	1st	29	Encoder A -	2nd
5	Encoder B +	1st	30	Encoder B +	2nd
6	Encoder B -	1st	31	Encoder B -	2nd
7	Encoder Index +	1st	32	Encoder Index +	2nd
8	Encoder Index -	1st	33	Encoder Index -	2nd
9	+/- 10V Analog Out	1st	34	+/- 10V Analog Out	2nd
10	Step Pulse +	1st	35	Step Pulse +	2nd
11	Step Pulse -	1st	36	Step Pulse -	2nd
12	Step Direction +	1st	37	Step Direction +	2nd
13	Step Direction -	1st	38	Step Direction -	2nd

6.4 Wiring Servo Motors

DSPpro controllers can control brush servo motors, brushless servo motors, or linear brushless motors. Basic connections require an analog output signal (from the board to the amplifier) and an encoder input (from the motor to the board).

6.4.1 Velocity/Torque Mode

Most amplifiers support either Velocity mode (voltage control), Torque mode (current control), or both. The DSPpro motion controllers can be used with either Velocity or Torque controlled servo motor/amplifier packages.

When the amplifier is in Velocity mode, the velocity of the motor is proportional to the analog input voltage (-10 volts to +10 volts). When the amplifier is in Torque mode the current applied to the motor is proportional to the analog input voltage (-10 volts to +10 volts). Generally, Velocity mode is more stable than Torque mode.

6.4.2 Encoder Input

DSPpro motion controllers accept TTL level (0v to 5v, 40mA max) encoder input from either differential or single-ended encoders. Differential encoders are preferred due to their excellent noise immunity. When used with differential encoders, the differential line receiver (LM26LS32) on the controller reads the difference between A+ and A- and between B+ and B-. By reading the difference between the square wave inputs any significant noise is canceled out.

The connections for a single ended encoder are identical to a differential encoder except channel A- and channel B- should be left unconnected (floating). The A- and B- lines are pulled up internally to 2.5 volts.

The controller also reads the index pulse (either single ended or differential). Typically, there is one index pulse per revolution of the encoder (rotary type), which can be used for homing.

Encoder signals are read in quadrature. This means every line on the encoder will produce a rising edge and a falling edge on channels A+ and B+ which are interpreted by the DSPpro as four encoder counts.

The amp enable and amp fault connections are discussed in chapter 7, “Dedicated I/O”. Any unused lines should be left unconnected.

6.4.3 Brush Servo Motors

The minimum required connections for brush type servo motors are:

Analog signal (+/- 10 V)

Signal Ground

Encoder Channel A +

Encoder Channel B +

+5 volts

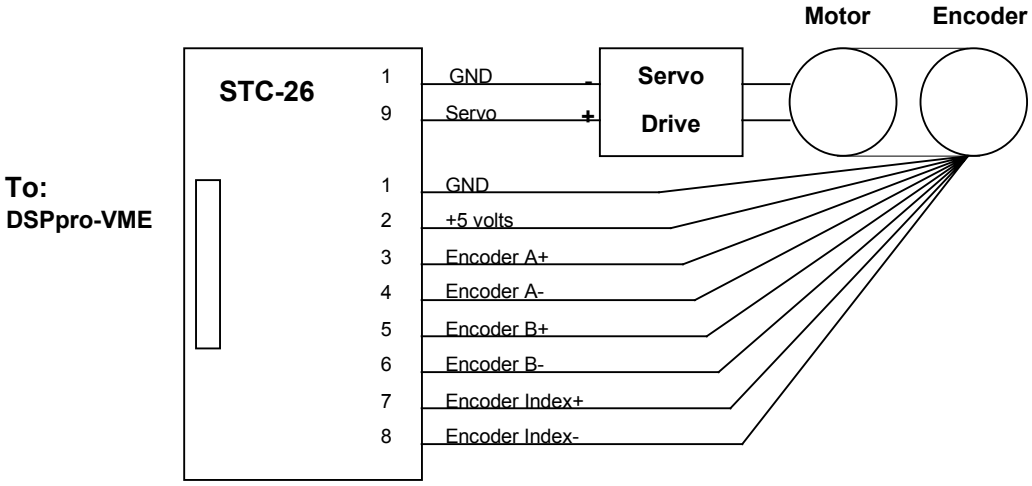

Figure 6-7. Typical Brush Servo Wiring with Differential Encoder
Axis 0 - DSPpro-VME Models

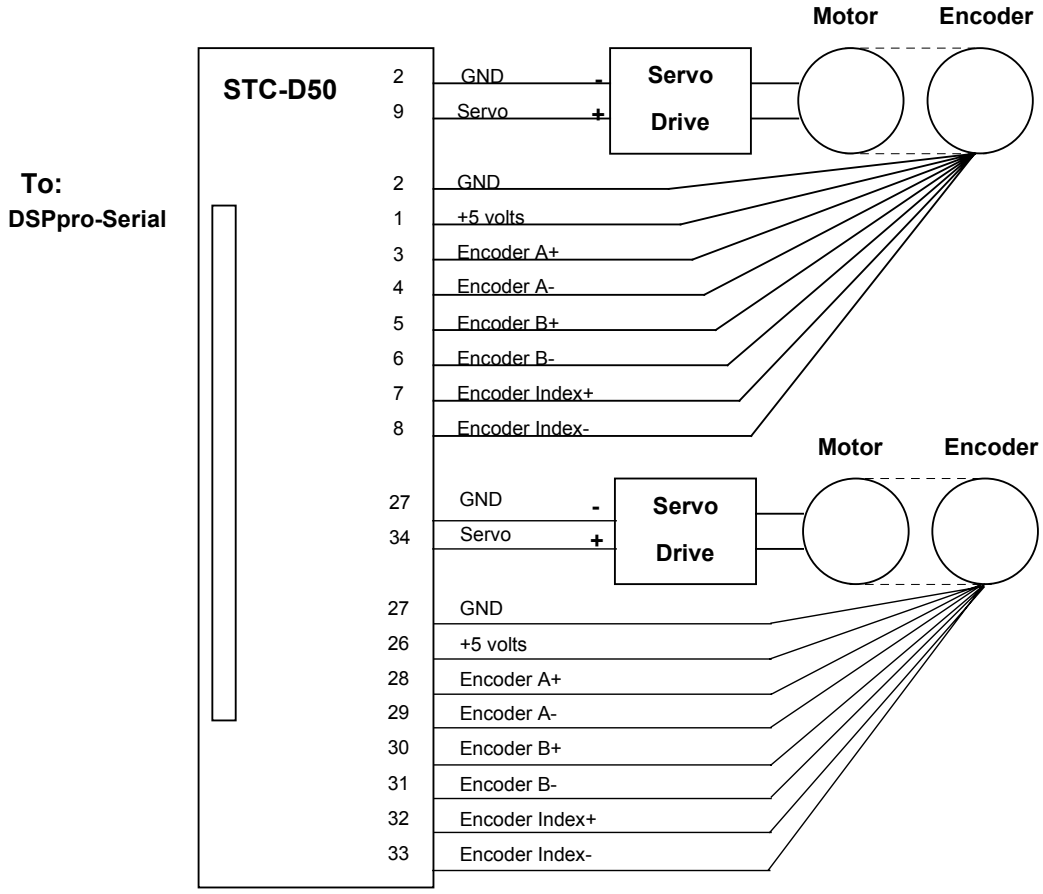

Figure 6-8. Typical Brush Servo Wiring with Differential Encoder
Axis 0 and 1 - DSPpro-Serial

6.4.4 Brushless Servo Motors

Wiring diagrams for typical brushless servo motors with differential encoders:

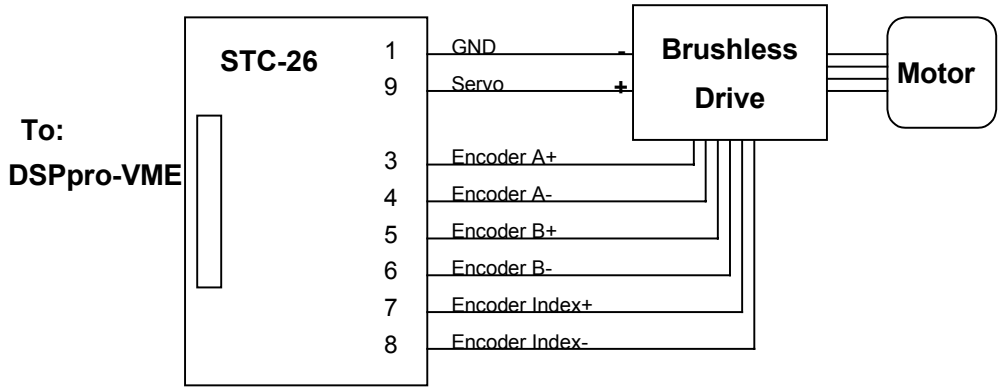

Figure 6-9. Typical Brushless Servo Wiring with Differential Encoder Axis 0 - DSPpro-PC and DSPpro-VME Models

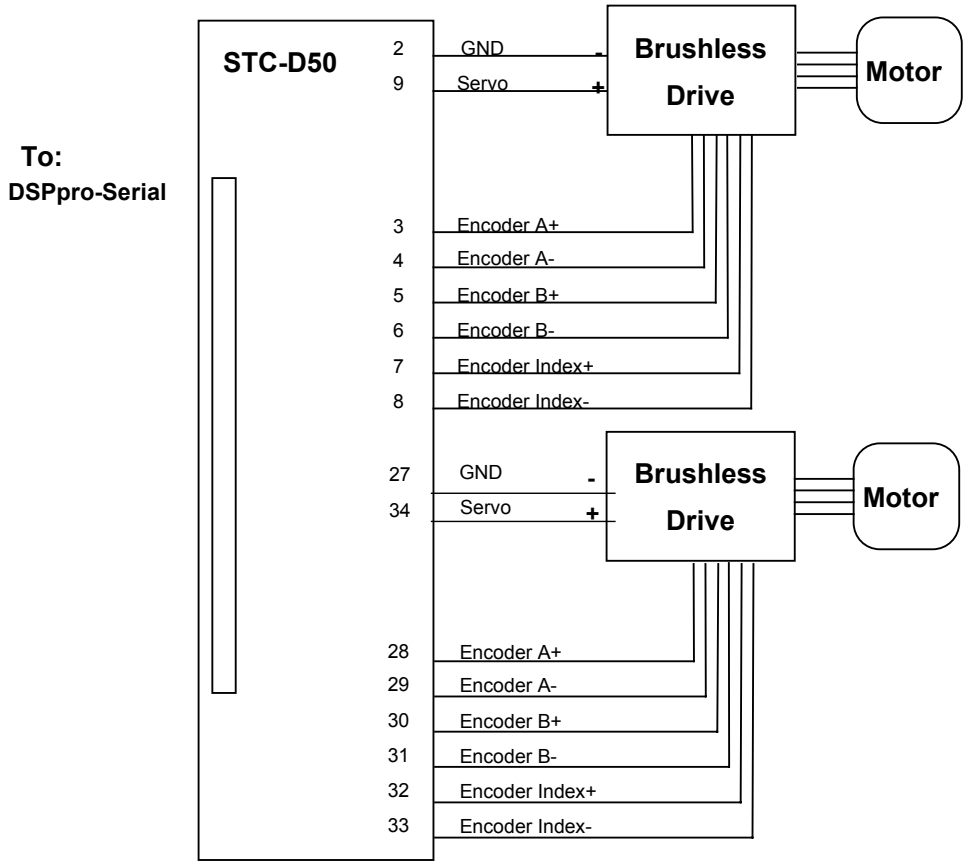

Figure 6-10. Typical Brushless Servo Wiring with Differential Encoders Axis 0 and 1 - DSPpro-Serial

The amp enable and amp fault connections are discussed in the chapter 7, "I/O Wiring." Any unused lines should be left unconnected.

6.4.5 Step-and-Direction Controlled Servo Motors

Some brushless servos are controlled by step-and-direction pulses. With this type of motor, the position information is communicated by step pulses, and the PID loop is handled internally by the drive itself.

To avoid possible instability caused by conflict between the drive PID loop and the controller's PID loop, it is best to operate step-and-direction servos as open-loop step motors. The controller will send step pulses and a direction pulse to the drive, which will handle the PID servo control internally.

6.5 Wiring Step Motors

6.5.1 Open-Loop Step Motors

DSPpro controllers can control step motors in both open-loop (no encoder) and closed-loop configurations. In the open-loop configuration the step pulse output (connected to the driver) is fed back into the controller and used to keep track of the "actual position." Thus the DSP closes the loop internally on a pair of axes when the open-loop step configuration is selected. Full/half and micro stepping drives are compatible with the boards.

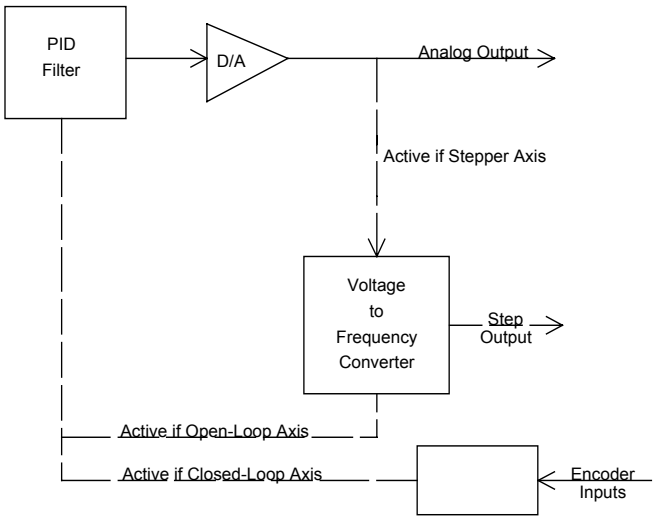

Figure 6-11. Internal Architecture to Control Step Motors

Most step drives require three wires for operation: step, direction and ground (or + 5 volts). The board provides a TTL level step pulse(+) output and direction(+) output for each axis. In addition, the complements of the step and direction signals are also provided (step(-) and dir(-)). Some drives allow differential inputs in which both step + and step - lines are connected for higher noise immunity. If in doubt, fax the driver data sheets or driver pin-outs to MEI Technical Support along with any questions.

Note that when only step + or step - is used, it may be necessary to jumper unused terminals on the step drive. Consult the manual on your step driver prior to connection.

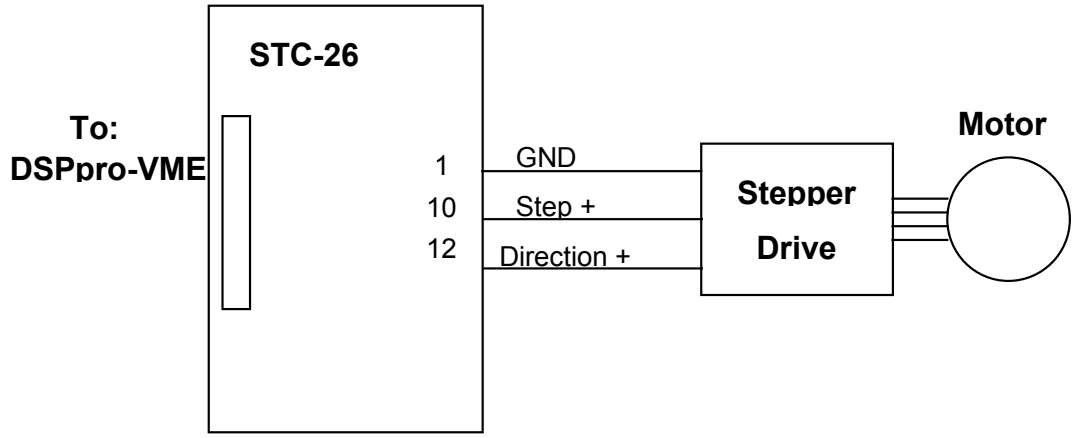

Figure 6-12. Typical Open-loop Step Wiring Axis 0 - DSPpro-VME*
*****For drives that trigger on the falling edge of the pulse input, use Step- instead of Step+.***

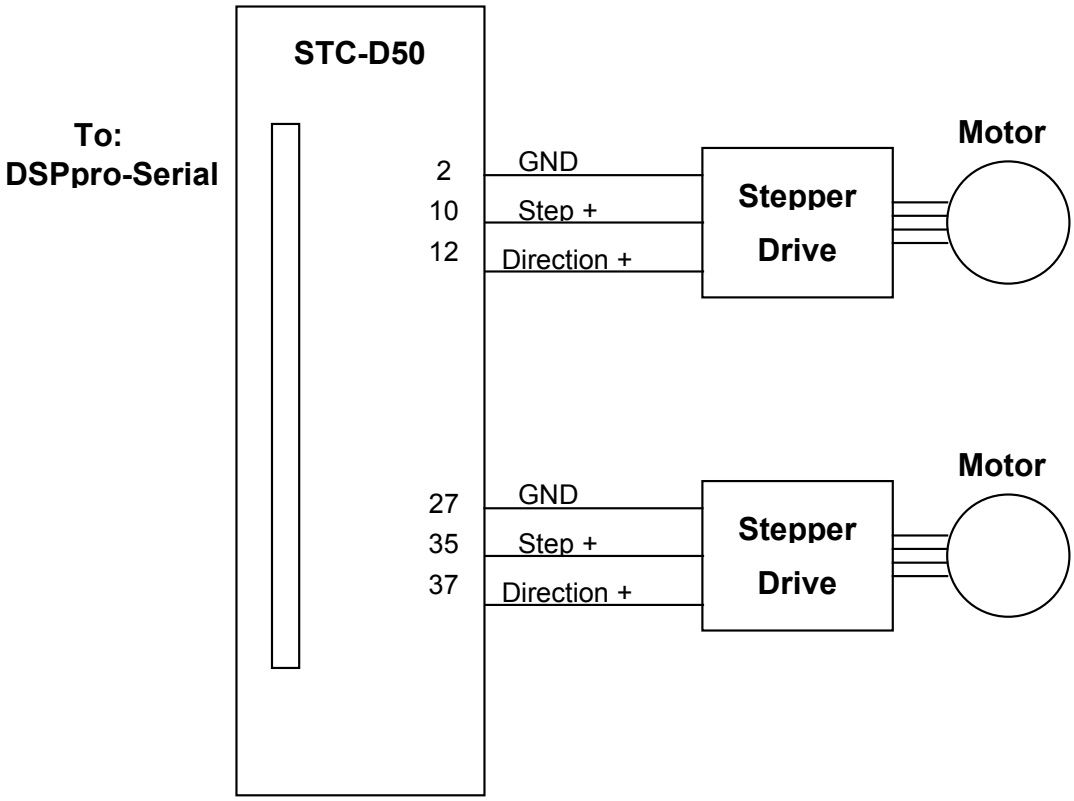

Figure 6-13. Typical Open-loop Step Wiring Axis 0 and 1- DSPpro-Serial*
*****For drives that trigger on the falling edge of the pulse input, use Step- instead of Step+.***

6.5.2 Direction Pulse Synchronization

DSPpro controllers synchronize the direction pulse with the falling edge of the positive step pulse output. When connected to the step drive properly, this ensures that a step pulse and direction change will never occur at the same time.

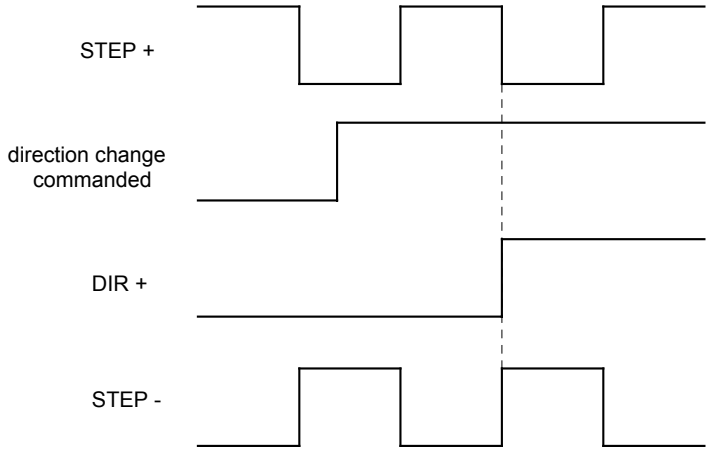

Figure 6-14. Direction Pulse Synchronization

Most step drives count pulses on either the rising edge or falling edge of the step pulse input. If the driver triggers on the falling edge, then the Step Pulse (-) from the board should be connected to the pulse input on the drive. If the driver triggers on the rising edge, then the Step Pulse(+) from the board should be connected to the pulse input on the drive. The Direction(+) line should be connected to the direction input of the drive. This will guarantee that the drive will never receive a direction change during a step pulse.

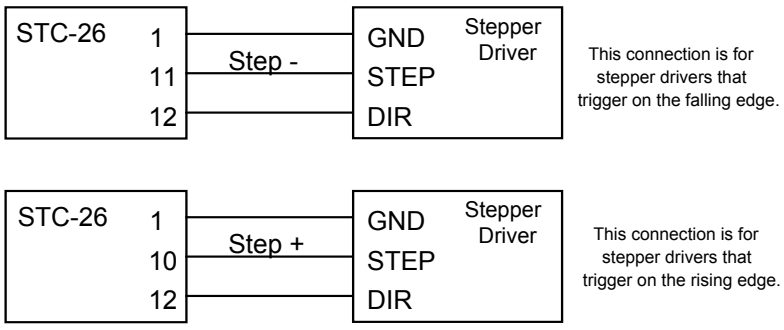

Figure 6-15. Wiring for Step Drives by Type for DSPpro-VME

6.5.3 Closed-loop Step Motors

DSPpro controllers can control step motors with encoder feedback. Closed-loop steppers are controlled by a PID algorithm running on the DSP in real time. The boards accept TTL level (0v to 5v, 40mA max) encoder input from either differential or single-ended encoders. Differential encoders are preferred due to their excellent noise immunity. The connections for a single ended encoder are identical to a differential encoder except nothing should be connected to channel A- and channel B-. The A- and B- lines are pulled up internally to 2.5 volts.

Encoder signals are read in quadrature. This means every line on the encoder will produce a rising edge and a falling edge on channels A+ and B+ which is interpreted by the DSPpro as four encoder counts.

Connecting closed-loop step motors to the board is similar to servo motors, except that the step and direction lines are connected instead of the analog signal. The minimum connections are:

- Step + (or Step -)
- Direction + (or Direction -)
- Signal Ground
- Encoder A + and B + lines
- + 5 Volts

Note that when only Step+ or Step- is used, it is often necessary to jumper unused terminals on the step driver. Consult the manual on your step drive prior to connection.

In general, use Step+ for drives with active high logic, and use Step- for drives with active low logic. Also, note that both Step+ and Step- lines can be connected to drives with differential inputs. If in doubt, fax the drive pin-outs to MEI along with any questions.

A sample wiring diagram for a step motor with differential encoder follows:

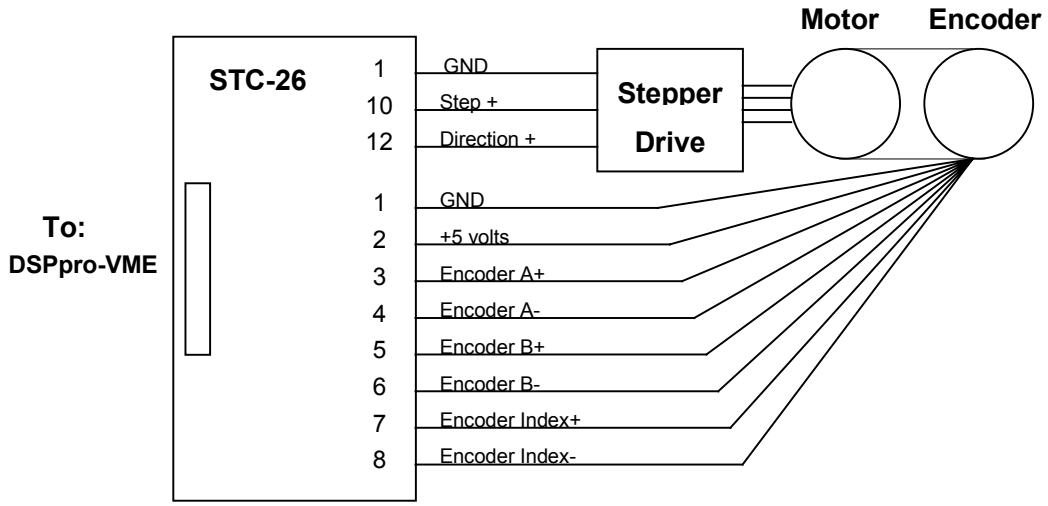

Figure 6-16. Typical Closed-loop Step Wiring with Differential Encoder, Axis 0 - DSPpro-VME

Note:

For drives that trigger on the falling edge of the pulse input use Step- instead of Step+.

6.6 Wiring for Dual-Loop Control

DSPpro controllers can be configured for dual-loop control. In dual-loop control, the velocity information for the PID derivative term (Kd) is derived from a rotary encoder on the motor shaft, and the position information for the PID proportional and integral terms (Kp and Ki) are derived from an encoder on the load itself.

The axis that will be used for the rotary encoder (velocity feedback) is configurable through software and can be any axis that is not controlling a motor. For example, if axis 0 is configured for velocity feedback and axis 1 is configured for positional feedback, the system would be connected as follows:

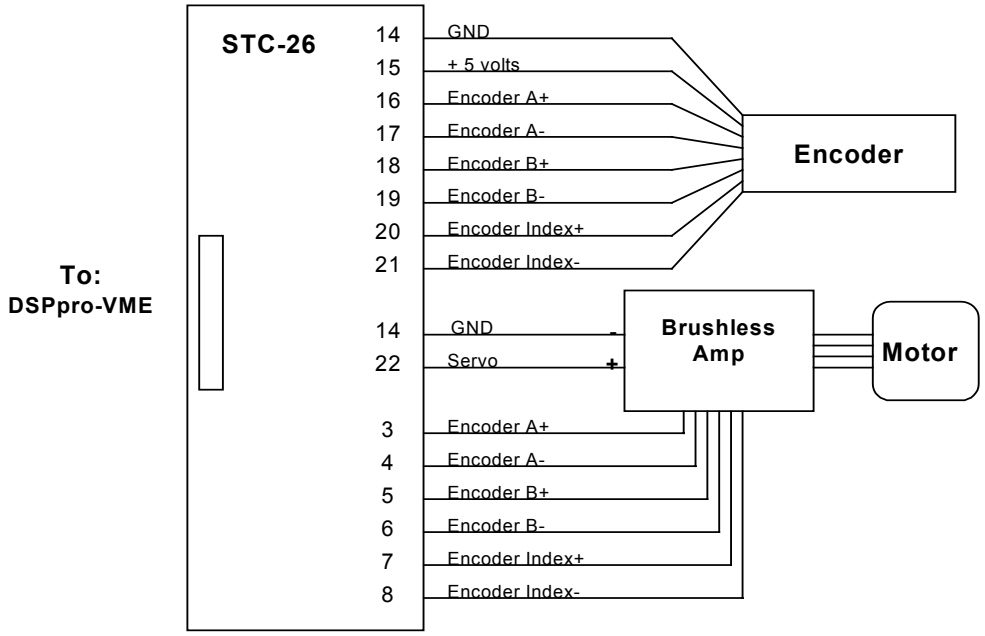

Figure 6-17. Typical Dual-loop Encoder Wiring for DSPpro-VME with Differential Encoders

6.7 Interferometer Input Wiring (DSPpro-VME only)

DSPpro-VME controllers can accept position feedback information from laser interferometers with 32-bit parallel output. User I/O lines are used to receive the 32-bit parallel input. Wiring diagrams for several popular interferometers are shown in the following sections.

6.7.1 Excel Precision 1000A Interferometer/Axis Board

Connection to the Excel Precision 1000A interferometer is done through an optional MEI interface board (bus buffer board). Standard CBL-50 ribbon cables are used to connect to the buffer board.

6.7.2 Excel 1032B Interferometer Wiring Pin-Out Listing

Laser Axis	Laser Name	MEI Pin #
0	Ouput Enable	P3-13
0	Laser Error	P2-41
0	Laser Reset	P2-15
0	Data Hold	P3-15
1	Ouput Enable	P3-11
1	Laser Error	P2-33
1	Laser Reset	P2-11
1	Data Hold	P3-15
2	Ouput Enable	P3-9
2	Laser Error	P2-25
2	Laser Reset	P2-7
2	Data Hold	P3-15
3	Ouput Enable	P3-7
3	Laser Error	P2-17
3	Laser Reset	P2-3
3	Data Hold	P3-15
ALL	Position Data Bit 0 (LSB)	P1-47
ALL	Position Data Bit 1	P1-45
ALL	Position Data Bit 2	P1-43
ALL	Position Data Bit 3	P1-41
ALL	Position Data Bit 4	P1-39
ALL	Position Data Bit 5	P1-37
ALL	Position Data Bit 6	P1-35
ALL	Position Data Bit 7	P1-33
ALL	Position Data Bit 8	P1-31
ALL	Position Data Bit 9	P1-29
ALL	Position Data Bit 10	P1-27
ALL	Position Data Bit 11	P1-25
ALL	Position Data Bit 12	P1-23
ALL	Position Data Bit 13	P1-21
ALL	Position Data Bit 14	P1-19
ALL	Position Data Bit 15	P1-17
ALL	Position Data Bit 16	P1-15
ALL	Position Data Bit 17	P1-13
ALL	Position Data Bit 18	P1-11
ALL	Position Data Bit 19	P1-9
ALL	Position Data Bit 20	P1-7
ALL	Position Data Bit 21	P1-5
ALL	Position Data Bit 22	P1-3
ALL	Position Data Bit 23	P1-1
ALL	Position Data Bit 24	P3-47
ALL	Position Data Bit 25	P3-45
ALL	Position Data Bit 26	P3-43
ALL	Position Data Bit 27	P3-41
ALL	Position Data Bit 28	P3-39
ALL	Position Data Bit 29	P3-37
ALL	Position Data Bit 30	P3-35
ALL	Position Data Bit 31 (MSB)	P3-33

Contact MEI for more information about using DSPpro controllers with Excel Precision laser interferometers.

6.7.3 Hewlett-Packard 10885A Interferometer/Axis Board

Signal Name		*Pin Number*	
HP	***MEI***	***HP****	***MEI***
GND	GND	2	P1-46
		3	
GND	GND	4	P1-44
~BP_error	Device Fault	5	P2-41(axis 0)
~BP_error	Device Fault	5	P2-33(axis 1)
~BP_error	Device Fault	5	P2-25(axis 2)
~BP_error	Device Fault	5	P2-17(axis 3)
GND	GND	6	P1-42
BP_meas_lol		7	
BP_ref_lol		8	
~BP_wnd0		9	
BP_overflow		10	
~BP_force_zero		11	
GND	GND	12	
~BP_pos_reset		13	
~BP_sample		14	P1-38
~BP_output_en	GND**	15	P1-40
~BP_output_hold	1C0	16	P3-15
~BP_clock		17	
GND	GND	18	P1-34
		19	
GND	GND	20	P1-32
		21	
		22	
P1	2A1	23	P1-45
P0	2A0	24	P1-47
P3	2A3	25	P1-41
P2	2A2	26	P1-43
GND	GND	27	P1-36
P4	2A4	28	P1-39
P6	2A6	29	P1-35
P5	2A5	30	P1-37
P8	2B0	31	P1-31
P7	2A7	32	P1-33
P10	2B2	33	P1-27
P9	2B1	34	P1-29
P12	2B4	35	P1-23
P11	2B3	36	P1-25
P13	2B5	37	P1-21
GND	GND	38	P1-28
P15	2B7	39	P1-17
P14	2B6	40	P1-19
P17	2C1	41	P1-13
P16	2C0	42	P1-15

**Pins as counted on the H-P header. Note that the H-P manual counts the pins differently.*

*** Connection of the output enable depends on the number of axes. For 1-axis systems, connect the output enable to ground (P1-40). For multi-axis systems, connect the pins as follows:*

	HP	***MEI***
Axis 0	*15*	*P3-13*
Axis 1	*15*	*P3-11*
Axis 2	*15*	*P3-9*
Axis 3	*15*	*P3-7*

Hewlett-Packard 10885A Interferometer/Axis Board, continued

Signal Name		*Pin #*	
HP	***MEI***	***HP****	***MEI***
P19	2C3	43	P1-09
P18	2C2	44	P1-11
GND	GND	45	P1-8
P20	2C4	46	P1-07
P22	2C6	47	P1-03
P21	2C5	48	P1-05
P24	1A0	49	P3-47
P23	2C7	50	P1-01
P26	1A2	51	P3-43
P25	1A1	52	P3-45
P28	1A4	53	P3-39
P27	1A3	54	P3-41
P29	1A5	55	P3-37
GND	GND	56	P1-04
P31	1A7	57	P3-33
P30	1A6	58	P3-35
		59	
		60	
		61	
		62	
GND	GND	63	P1-48
GND	GND	64	P1-02

Chapter 7

I/O WIRING

7.1 General

DSPpro controllers have I/O lines divided into two groups: Dedicated I/O and User I/O. There are six Dedicated I/O signals for each axis of the controller -- four inputs and two outputs:

Inputs	Positive Limit Switch
	Negative Limit Switch
	Home Sensor
	Amplifier Fault
Outputs	In Position
	Amplifier Enable

The DSPpro-VME controllers support 24 or 44 bits of general purpose I/O for user-defined functions. The DSPpro-Serial controller supports 24 bits of optically isolated general purpose I/O. These signals can be configured as inputs or outputs in groups of 8.

On DSPpro-VME controllers, the unused Dedicated I/O for axes 4-7 is available for use as User I/O, except for the Home Sensor Inputs.

7.2 I/O Wiring - DSPpro-VME

Connectors P1, P2, and P3 contain all user and dedicated I/O lines. These connectors are located at the far left-hand side of the controller, as shown in Figure 7-1. Each connector is divided into three sections -- A, B, and C -- as illustrated in Figures 7-1 and 7-2.

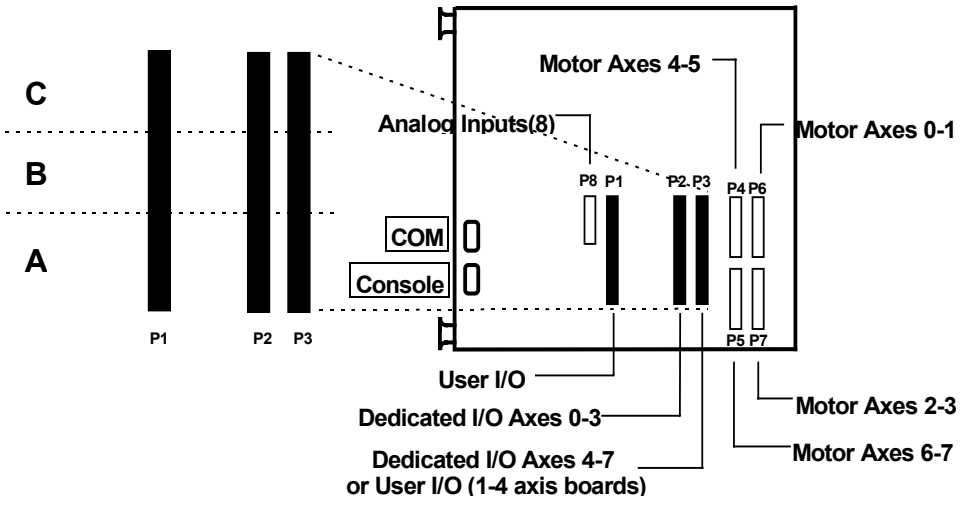

Figure 7-2. User and Dedicated I/O Headers - DSPpro-VME

The following table illustrates the pinouts for the headers P1 P2, and P3. These pinouts are identical for the DSPpro-VME.

I/O	*Description*
P1A	User Port 0 (8 bits input or output)
P1B	User Port 1 (8 bits input or output)
P1C	User Port 2 (8 bits input or output)
P2A	Dedicated Inputs for Axes 0 and 1
P2B	Dedicated Inputs for Axes 2 and 3
P2C	Dedicated Outputs for Axes 0-3
P3A	Dedicated Inputs for Axes 4 and 5 or User Port 3 (6 bits in or 6 bits out)*
P3B	Dedicated Inputs for Axes 6 and 7 or User Port 4 (6 bits in or 6 bits out)*
P3C	Dedicated Outputs for Axes 4-7 or User Port 5 (8 bits input or output)*

* The function of the signals on P3 depends on the number of axes. Boards with 5 or more axes use P3 for Dedicated I/O. On boards with 4 axes or less, P3 is available for user I/0. If P3 is used for User I/O, User Ports 3, 4 or 5 can be configured for inputs or outputs

The User I/O connections use 50 pin box headers and are Opto-22/Grayhill/Gordos compatible. Even numbered pins are grounds and pin 49 is +5 volts.

User I/O available on DSPpro-VME (all models):

Bit	Port	Header	Pin	Bit	Port	Header	Pin	Bit	Port	Header	Pin
0	0	P1	47	8	1	P1	31	16	2	P1	15
1	0	P1	45	9	1	P1	29	17	2	P1	13
2	0	P1	43	10	1	P1	27	18	2	P1	11
3	0	P1	41	11	1	P1	25	19	2	P1	9
4	0	P1	39	12	1	P1	23	20	2	P1	7
5	0	P1	37	13	1	P1	21	21	2	P1	5
6	0	P1	35	14	1	P1	19	22	2	P1	3
7	0	P1	33	15	1	P1	17	23	2	P1	1

User I/O available on +DSPpro-VME (2 or 4-axis only):

Bit	Port	Header	Pin	Bit	Port	Header	Pin	Bit	Port	Header	Pin
24	3	P3	47	32	4	P3	31	40	5	P3	15
25	3	P3	45	33	4	P3	29	41	5	P3	13
26	3	P3	N/A*	34	4	P3	N/A*	42	5	P3	11
27	3	P3	41	35	4	P3	25	43	5	P3	9
28	3	P3	39	36	4	P3	23	44	5	P3	7
29	3	P3	37	37	4	P3	21	45	5	P3	5
30	3	P3	N/A*	38	4	P3	N/A*	46	5	P3	3
31	3	P3	33	39	4	P3	17	47	5	P3	1

* bits 26, 30, 34, and 38 are not available

Both Dedicated and User I/O signals come directly from 82C55 programmable I/O controllers. These signals can be programmed in groups of 8 as inputs or outputs. The input state is a high impedance TTL level input. The output state has TTL logic levels with +/- 2.5 mA drive current (4.0 mA max). The power up state of all User I/O is high impedance (Input state).

7.3 I/O Wiring - DSPpro-Serial

All connections for I/O on the DSPpro-Serial are optically isolated to provide electrical isolation between the controller and the equipment to which it is connected.

7.3.1 Dedicated I/O Wiring

The connections for dedicated I/O on the DSPpro-Serial are located on the same 50-pin DB-50 connectors which contain the motor signal pinouts. There are two axes of dedicated I/O pinouts on each connector, just as there are two axes of motor signal pinouts on each connector. The pinouts for dedicated I/O for each connector are shown below:

Pin	*Signal*	*Axis*	*Pin*	*Signal*	*Axis*
15	Amp Enable	1st	40	Amp Enable	2nd
16	In Position	1st	41	In Position	2nd
17	Amp Fault	1st	42	Amp Fault	2nd
18	Home	1st	43	Home	2nd
19	Positive Limit	1st	44	Positive Limit	2nd
20	Negative Limit	1st	45	Negative Limit	2nd
21	+V User	1st	46	GND User	2nd

All +V User pins are tied together. All GND User pins are tied together.

7.3.2 User I/O Wiring

The connections for user I/O on the DSPpro-Serial are located on the 50-pin DB-50 connector labeled “I/O” as shown in figure 7-3. This connector also contains the pinouts for the analog inputs, as described in section 7.7.2.

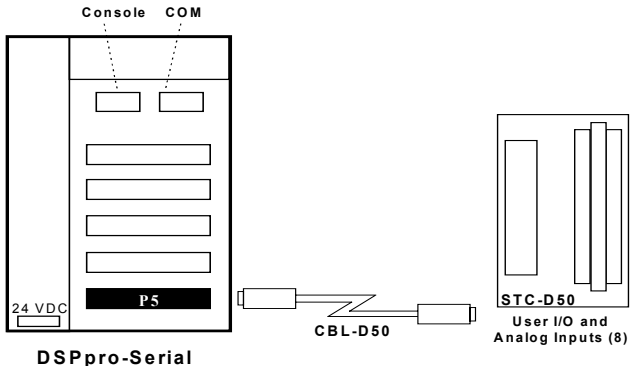

Figure 7-3. User I/O and Analog Inputs Connector- DSPpro-Serial

The pinouts for user I/O are shown below:

Outputs				*Inputs*				*Inputs*			
Bit	***Port***	***Header***	***Pin***	***Bit***	***Port***	***Header***	***Pin***	***Bit***	***Port***	***Header***	***Pin***
0	0	P5	2	8	1	P5	10	16	2	P5	18
1	0	P5	3	9	1	P5	11	17	2	P5	19
2	0	P5	4	10	1	P5	12	18	2	P5	20
3	0	P5	5	11	1	P5	13	19	2	P5	21
4	0	P5	6	12	1	P5	14	20	2	P5	22
5	0	P5	7	13	1	P5	15	21	2	P5	23
6	0	P5	8	14	1	P5	16	22	2	P5	24
7	0	P5	9	15	1	P5	17	23	2	P5	25

7.4 Home and limit switch wiring (DSPpro-VME only)

For small, electrically quiet machines the home and limit switches can be wired directly to the dedicated inputs. For larger, noisier machines optical isolation is recommended. The following diagrams show the wiring for both types of machines.

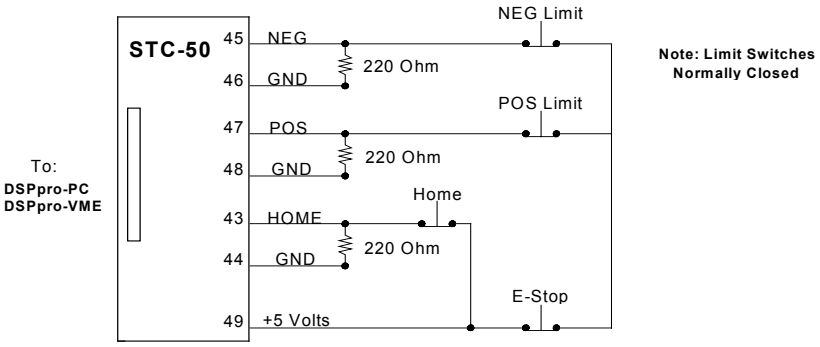

Figure 7-4. Sample Wiring Diagram for Axis 0 Limit Switch - Non-Opto-Isolated

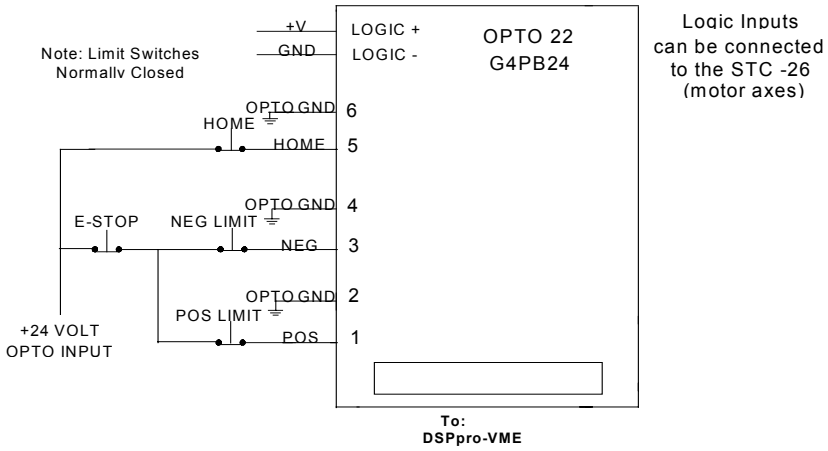

Figure 7-5. Sample Wiring Diagram for Axis 0 Limit Switch - Opto-Isolated

Fail-safe limit operation is provided for both the optically isolated and non-isolated limit circuits. If a wire breaks in the limit circuit the associated limit is activated and the motion is stopped until the problem is corrected. Since the controller can be configured for either active high or active low inputs, other limit and home sensor circuits can be used.

7.5 Dedicated and user output wiring

User outputs on the DSPpro-VME are driven by an Intel 82C55 Programmable Peripheral Interface Controller. When power is supplied to the 82C55 these outputs have three possible output states: High Impedance (High-Z) (1 micro amp leakage current), High (>3.0V at 2.5 milliamp source current) , and Low (< 0.4V at 2.5 milliamperes sink current). If there is no power to the 82C55 the output state is held low by input protection diodes.

The figure below shows the power on and power off timing of the board output states. Approximately .3 to .5 seconds after power is supplied to the computer the User outputs will go to the Power-On state. This state can be any one of the three output states of the 82C55. The Power-On state is configured at the factory to be the Low state.

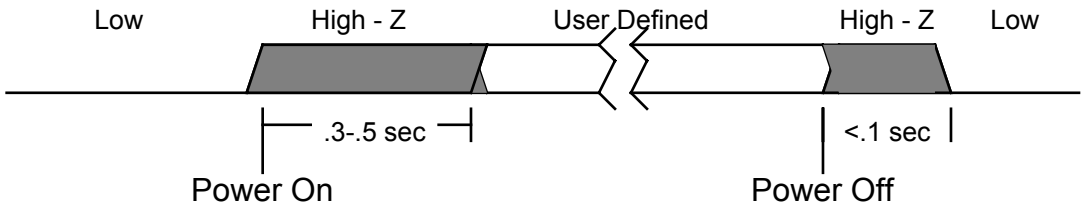

Figure 7-6. Power On/Off Timing

Critical control signals such as amplifier enable/disable which must always be in a defined state should be designed so that the default state of the 82C55 output is Low. A pull-down resistor should be used to insure that the output does not float high when the output is in the High-Z state. The following figure shows the correct wiring for amplifiers with Low Enable and High Enable inputs:

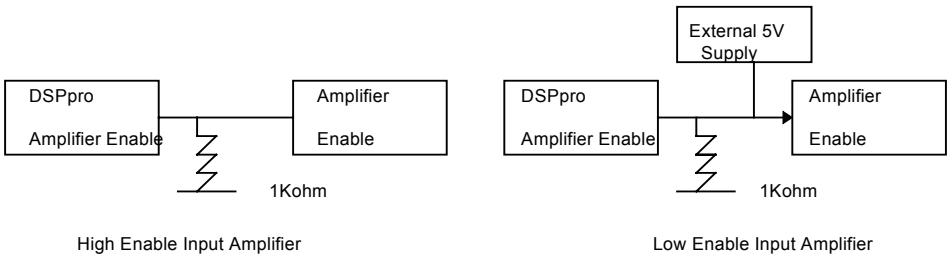

Figure 7-7. Amplifier Enable Wiring

7.6 Opto-isolation

On the DSPpro-VME, dedicated and user I/O headers conform to Opto-22/Grayhill/Gordos standard pin arrangement, and may be connected directly. Some Opto-22 racks do not use the +5V logic power on pin 49 of the I/O connector. +5 volts must be provided from an external source. Grayhill racks can be configured to take the logic power from pin 49 so that no external source is necessary.

When power is first applied to the DSPpro boards, the User I/O signals come up in a high-impedance state and the `amp_enable` line in the Dedicated outputs comes up low. Most opto-isolation modules invert the I/O signals, which means that I/O signals may come up high. The active level of the Dedicated I/O signals can be configured in the SETUP program and the boot configurations of the User I/O signals can be set with the function libraries. Refer to the MEI *DSP-Series Motion Controller C Programming Manual* for detailed information.

DSPpro-Serial controllers have internal opto-isolation so no external I/O isolation is required.

7.7 Analog input wiring

7.7.1 DSPpro-VME

Analog inputs for the DSPpro-VME are connected to the 20 pin connector P8. Pins 2 and 20 (Analog GND) are connected to the logic ground of the A/D chip and to a separate ground plane beneath the A/D chip. The logic ground of the A/D chip is also connected to the bus ground (with all of the other GND signals). When connecting analog inputs it is often better to use the separated analog grounds to improve noise immunity.

There are 8 channels, each with 12-bit resolution. Each channel can be configured as either Unipolar (0 to 5V) or Bipolar (-2.5V to +2.5V). Since there is no buffer between the P8 connector and the A/D chip, the input voltages must not exceed +5V or fall below -2.5V.

7.7.2 DSPpro-Serial

Analog inputs for the DSPpro-Serial are connected to the 50-pin connector P5 marked User I/O. The MEI accessory STC-D50 can be used to simplify wiring. The STC-D50 includes screw-

terminal connectors for the I/O wiring, and a 50-pin header for connecting to MEI accessory cable CBL-D50.

The following diagram illustrates these connections.

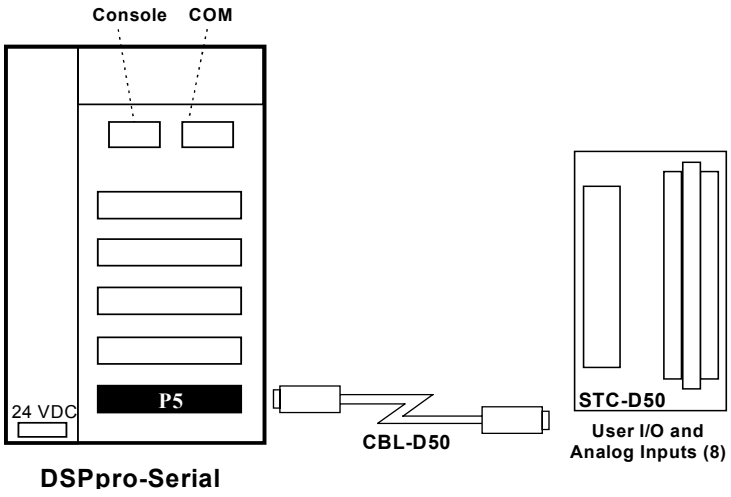

Figure 7-8 DSPpro-Serial Analog Input Connections

7.8 8254 Counter wiring - DSPpro-VME

There are (3) 16-bit counters available for user functions. Counter 0 can take an external clock input (pin 3 on P8) and Counters 1 and 2 have fixed frequency inputs of 1.25 and 10 MHz respectively. The gate signal for Counter 0 (used in some modes) is on pin 11 of P8. All counter outputs are available on P8.

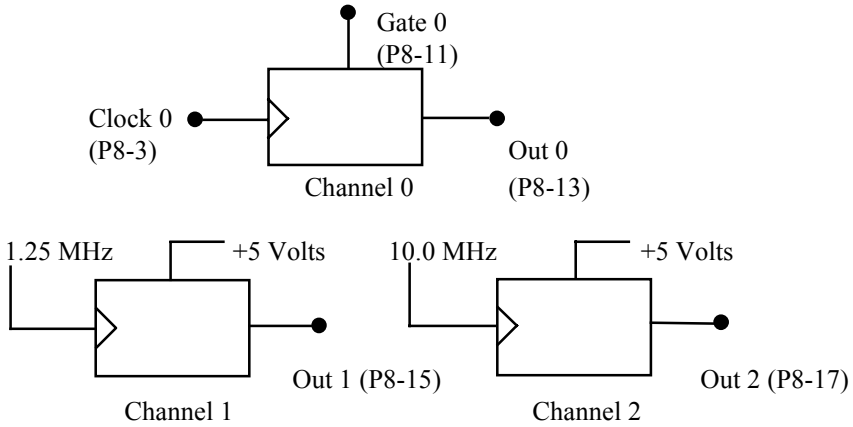

Figure 7-9 Counter Wiring Diagram

Chapter 8

DEVELOPING YOUR APPLICATION

8.0 Overview

DSPpro controllers are designed to execute motion programs autonomously using their on-board CPU. The DSPpro executes these programs from DRAM, which boots from a boot disk image stored in flash memory, or a boot disk on the host computer via a serial cable connection or directly through dual-port memory (DSPpro-PC only). This process is similar to a PC booting from its hard drive.

During application development, we recommend that you test and debug your program using the host computer's processor along with an MEI LC/DSP or PCX/DSP controller. With this method, you benefit from the development and debugging tools supplied with your application development environment (e.g. Microsoft Visual C++, Borland C, etc.). Because all DSP and DSPpro Series controllers share the same C function motion library, software developed with a DSP Series controller can be executed on a DSPpro Series controller with little or no changes.

After you are finished debugging and testing your program on the host with a PCX/DSP or other DSP Series controller, you can use the supplied utilities to test your application on the DSPpro's CPU. When you're satisfied with the operation of your program on the DSPpro, simply download the application using the flash memory utilities to commit it to the non-volatile memory on the DSPpro. You can use either the on-board dual-port memory (DSPpro-PC only) or a serial connection to connect the two systems.

Because the DSPpro does not have a video output port, you should be sure that your application does not to make use of the video display for troubleshooting or other functions. Instead, communications with the host computer should take place over serial connections using the serial toolkit, or across the bus interface using the dual-port toolkit. Other than these display issues, your program should require no changes to execute on the DSPpro's CPU.

When developing an application on the DSPpro, system developers will typically take three steps (illustrated below). These steps are described in more detail in the following sections.

1) Develop the application using a PC and a DSP-Series controller.
2) Verify the application with the DSPpro.
3) Burn the application into flash memory.

Figure 8-1. DSPpro application development process

8.1 Write the application program

The first step in developing your motion application is to write and compile a program on a host PC using your standard PC development tools. This process involves integrating the MEI C function libraries with your compiler of choice, developing your motion application, and compiling the application along with the appropriate libraries.

Because the DSPpro executes DOS on-board, it is important to be sure to compile **16-bit targets** from your development environment. You can do this by calling your compiler from a DOS prompt, or by selecting a 16-bit build from within your Windows development environment.

Consider the following example which assumes a DOS development environment and the Borland Turbo C compiler. If your application is called YOURAPP.C, you would type the following at the command prompt:

```
C:\TEST> bcc -ml yourapp.c medbc45l.lib
```

This calls the Borland command-line C compiler and supplies it with the source code of the program under development (`yourapp.c`) along with the appropriate MEI function libraries (in this case `medbc451.lib`).

If you are using Microsoft Visual C++, you would use the following command to compile `yourapp.c`:

```
C:\TEST> cl /AL yourapp.c medcl80l.lib
```

MEI supplies pre-compiled C function libraries for many popular compilers, and source code libraries to use with unsupported development environments.

8.2 Testing the Application Program

If you have access to a computer with a standard MEI bus-based DSP Series motion controller (e.g. PCX/DSP, LC/DSP, etc.), we strongly recommend using it as an application development platform—the software should require no changes to run in the DSPpro.

When you reach the final stages of your development, you will want to execute the application directly on the DSPpro CPU. MEI has developed utilities that enable the DSPpro to use the keyboard and screen capabilities of the host system to provide a console for the DOS shell running on the DSPpro. Using these utilities, your program can send diagnostic and other information from the DSPpro to the host computer across the serial connection.

The utility to communicate with the DSPpro-Serial and DSPpro-VME is called REMSVR. The only difference between the two utilities is that RCONSOLE communicates with the DSPpro directly through the PC's ISA bus, while REMSVR communicates with the DSPpro-Serial and DSPpro-VME using the console port and a null-modem cable.

The following diagram illustrates this connection:

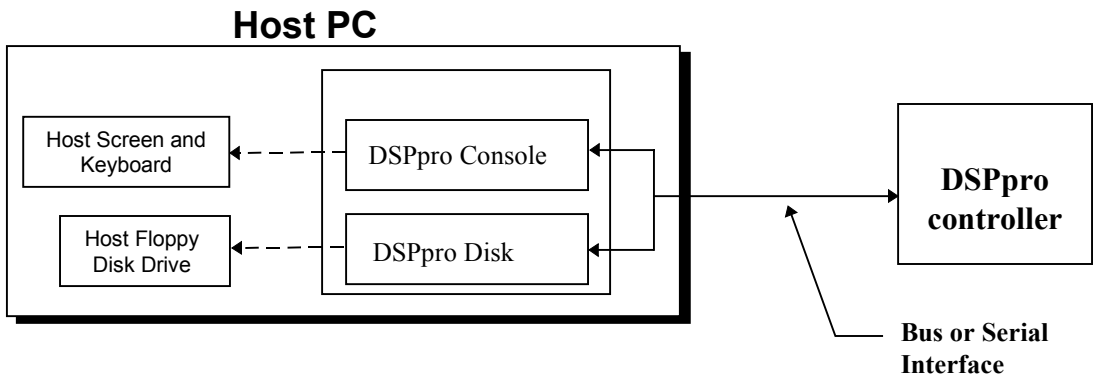

Figure 8-2 DSPpro to Host Communications

In addition to providing console for the DSPpro, these utilities enable the DSPpro to use the host computer's floppy disk drive as its file system. Both RCONSOLE and REMSVR use the *DSPpro Boot Disk* included in your distribution to provide the DOS shell to the DSPpro controller.

When the controller is reset, it first attempts to establish a communications channel with RCONSOLE or REMSVR running on the host computer directly through the CONSOLE port on the DSPpro-VME and DSPpro-Serial.

If neither RCONSOLE nor REMSVR is running, the card is not installed in the host computer, or the CONSOLE connection has not been made, then the DSPpro will boot out of its flash memory.

In order to use RCONSOLE or REMSVR, your application must make use of special-purpose C functions provided to aid host-controller communications. These functions are provided in the Serial and Dual-port toolkits. The following sections describe these tools.

8.2.1 Configuring Communications

All DSPpro Series controllers have the capability to communicate with the host system using two RS-232 serial ports. The two ports, CONSOLE and COM, are used to communicate with the host for debugging and testing purposes and for input and output communications of application programs. Applications developed for the DSPpro can make use of these ports using the Serial Toolkit, which provides a suite of C functions to communicate via either of the two serial ports on the DSPpro.

In addition, board-level controllers such as the DSPpro-VME can communicate with the host system directly over the host bus interface. Applications can use the bus interface to access the shared memory regions of the host system using the Dual-port Toolkit. The dual-port toolkit provides a suite of C functions to communicate with the host system using this region of shared memory.

Both the Serial Toolkit and Dual-port Toolkit are explained in detail in the *DSPpro Release Notes*.

When DSPpro controllers are shipped from the factory, the flash memory is preconfigured to enable communication to the RCONSOLE or REMSVR application, as described below. Data

written to the standard output will appear on the host computer's RCONSOLE or REMSVR window. No programming is necessary to establish this communication channel.

DSPpro-VME and DSPpro-Serial controllers are pre-configured to use the serial ports for communication with the host computer. DSPpro-Serial controllers are pre-configured at the factory to use the dual-port memory interface for communication with the host computer.

The DSPpro-VME also has a dual-port memory interface, but comes configured for serial port support only. Contact MEI's Application Engineering department for more information on using the dual-port memory interface with the DSPpro-VME.

8.2.2 Using RCONSOLE to Test Your Application (DSPpro-PC only)

Before using RCONSOLE, be sure that the DSPpro-PC is properly installed and seated firmly in the host PC. See chapter 4, "Hardware Installation" for more information. In addition, you may want to verify basic installation using the techniques described in chapter 2, "Quick Start."

RCONSOLE may be used to test your application program or to monitor your application running on the DSPpro controller. This section describes the process necessary to use RCONSOLE to test your application before committing it to flash memory.

RCONSOLE uses the *DSPpro Boot Disk* to provide the DOS shell to the DSPpro controller when testing your application. Before using RCONSOLE you will need to copy your application to the *DSPpro Boot Disk*. Be sure to make a backup of the *DSPpro Boot Disk* before you copy your application. Copy your application from the host computer's hard drive to the *DSPpro Boot Disk*.

Because the DSPpro uses the floppy disk drive of the host computer as its file system when operating under RCONSOLE, you will not have access to your application if you do not copy it to the *DSPpro Boot Disk*, or if it is not already committed to the flash memory on the DSPpro.

Follow these instructions to use RCONSOLE:

1. Insert the *DSPpro Boot Disk* in the floppy drive of the host computer.
2. Execute the RCONSOLE application typing the following command at the DOS prompt:

```
A:\ rconsole /r
```

3. The `/r` flag tells RCONSOLE to reset the controller, which will then attempt to connect to the RCONSOLE application running on the host computer.
4. Upon successful connection, the DSPpro controller will attach to the RCONSOLE program running on the host computer and download DOS from the *DSPpro Boot Disk* in the host computer's floppy drive. The output on the screen of the host should look similar to the following:

```
MOTION ENGINEERING 80386EX BIOS V1.0
Copyright © 1992-1996 Motion Engineering, Inc.
Copyright © 1992-1995 General Software, Inc.

Embedded BIOS ROM Disk Enabled
```

```
Embedded BIOS Remote Disk [COM1] [CONNECTED]
Current date is …

Microsoft® MS-DOS® Version 6.22
Copyright Microsoft Corp 1981-1993

A:\>
```

4. When you see the A:\> prompt, you are ready to execute your program. Enter its name at the prompt and press <RETURN>.

5. Your program should execute on the DSPpro, and you will see the console output (if any) on the monitor of your host computer.

6. To exit RCONSOLE, type Alt-X.

8.2.3 Using REMSVR to Test Your Application (DSPpro-VME or DSPpro-Serial)

If you are using the DSPpro-VME or DSPpro-Serial, make sure that the CONSOLE cable is properly connected before using REMSVR. If you are using the DSPpro-PC, make sure that the card is properly installed in the host computer. See chapter 4, "Hardware Installation" for more information. In addition, you may want to verify basic installation using the techniques described in chapter 2, "Quick Start."

REMSVR may be used to test your application program, or to monitor your application running on the DSPpro controller. This section describes the process necessary to use REMSVR to test your application before committing it to flash memory.

REMSVR uses the *DSPpro Boot Disk* to provide the DOS shell to the DSPpro controller. Because the DSPpro uses the floppy disk drive of the host computer as its file system when operating under REMSVR, you will not have access to your application if you do not copy it to the *DSPpro Boot Disk*.

Before using REMSVR for the first time, you will need to copy your application to the *DSPpro Boot Disk*. Be sure to make a backup of the *DSPpro Boot Disk* before you copy your application. Copy your application from the host computer's hard drive to the *DSPpro Boot Disk*.

Follow these instructions to use REMSVR:

1. Insert the *DSPpro Utility Disk* in the floppy drive of the host PC.

2. Execute the REMSVR application. Use the following syntax to execute REMSVR:

`remsvr`	If the console cable is connected to COM1 on the host
`remsvr /2`	If the console cable is connected to COM2 on the host

3. The output on the screen of the host should look similar to the following:

```
Motion Engineering Console/Disk Server V1.0
Copyright (C) 1996 Motion Engineering, Inc.
Press Alt-X to exit
Connecting Through COM1
```

4. Remove the *DSPpro Utility Disk* from the floppy drive of the host PC and insert the *DSPpro Boot Disk*. Make sure it has your application program copied onto it.

5. Reset the controller using the reset button or by toggling power to the controller. The DSPpro will then attempt to connect to the REMSVR application running on the host computer.

6. Upon successful connection, the DSPpro controller will attach to the REMSVR program running on the host computer and download DOS from the *DSPpro Boot Disk* in the host computer's floppy drive. The output on the screen of the host should look similar to the following:

```
MOTION ENGINEERING 80386EX BIOS V1.0
Copyright © 1992-1996 Motion Engineering, Inc.
Copyright © 1992-1995 General Software, Inc.

Embedded BIOS ROM Disk Enabled
Embedded BIOS Remote Disk [COM1] [CONNECTED]
Current date is …

Microsoft® MS-DOS® Version 6.22
Copyright Microsoft Corp 1981-1993

A:\>
```

4. When you see the A:\> prompt, you are ready to execute your program. Enter its name at the prompt and press <RETURN>.

5. Your program should execute on the DSPpro, and you will see the console output (if any) on the monitor of your host computer.

6. To exit REMSVR, type Alt-X. Your program will continue to run on the DSPpro uninterrupted until it terminates.

8.2.4 Configure DOS startup files

The *DSPpro Boot Disk* can contain standard DOS configuration files to provide additional functionality to your application. These files, `AUTOEXEC.BAT` and `CONFIG.SYS`, can be used to set system variables, and to control the loading of your application. For example, you could have the DSPpro automatically execute your application (or series of applications) upon boot.

Because the DSPpro runs MS-DOS 6.22, you must be sure to follow the appropriate syntax for your AUTOEXEC.BAT and CONFIG.SYS files. This syntax is described in detail in the Microsoft publication *Microsoft MS-DOS Concise User's Guide*. To order a copy of this publication, contact the Microsoft Sales Information Center at (800) 426-9400.

8.3 Committing your application to flash memory

Once you have completed developing and debugging your application, you are ready to copy the contents of the boot disk to flash memory on the DSPpro. After loading the program to flash memory, the DSPpro controller can execute it completely independent of the host.

8.3.1 Committing your application to the DSPpro-VME or DSPpro-Serial

Follow these instructions to commit your program to flash memory on the DSPpro-VME or DSPpro-Serial:

1. If you haven't done so already, copy your application files to the *DSPpro Boot Disk*. Be sure that you have a backup of the original *DSPpro Boot Disk*.
2. If necessary, create `AUTOEXEC.BAT` and `CONFIG.SYS` files for use on the DSPpro. You may want your `AUTOEXEC.BAT` file to automatically invoke your application. Save these files on the *DSPpro Boot Disk*.
3. Connect the console cable between the host computer (use either COM1 or COM2) and the DSPpro Console port. Be sure to use a DB-9F to DB-9F null-modem cable, available from MEI as accessory cable CBL-D9 CONSOLE.
4. Execute the `REMSVR` program on the *DSPpro Utility Disk*, using the following options to select the appropriate serial ports:

`remsvr`	If the console cable is connected to COM1 on the host
`remsvr /2`	If the console cable is connected to COM2 on the host

3. Reset the controller by pressing the reset button or by toggling the power supply. The DSPpro will then attempt to connect to the REMSVR application running on the host computer.
4. After the DSPpro boots, you will see a DOS prompt from the DOS shell running on the DSPpro. At the prompt, type the following command:

```
procook /p
```

This executes the `PROCOOK` utility with the `/p` option, which pauses the utility to allow you to insert the *DSPpro Boot Disk*. You should see the following message:

```
Paused:  Insert your boot disk and press any key
```

5. Remove the *DSPpro Utility Disk* from the host computer's disk drive, replace it with the *DSPpro Boot Disk*, and press any key. `PROCOOK` will read the files on the disk and copy them into the flash memory on the DSPpro.
6. When `PROCOOK` finishes, exit `REMSVR` by typing Alt-X.
7. Restart the DSPpro by pushing the reset button or power-cycling the controller.
8. The DSPpro should boot from flash memory and begin execution of your program (as specified in the `AUTOEXEC.BAT` file).

Appendix A: CONNECTOR PINOUTS

A.1 DSPpro-VME Connector Layout

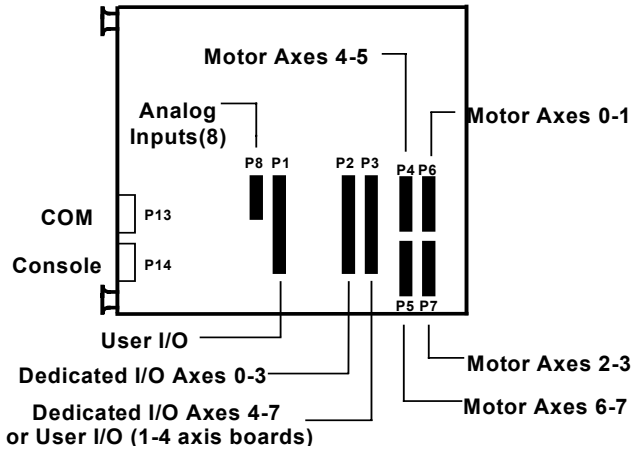

Figure A-2. Connector Locations - DSPpro-VME

A.2 Connector pinouts (DSPpro-VME)

Motor Axes Connections P4 - P7 26-pin box header

Pin	*Signal*	*Axis*
1	GND	1st
2	5V	1st
3	Encoder A +	1st
4	Encoder A -	1st
5	Encoder B +	1st
6	Encoder B -	1st
7	Encoder Index +	1st
8	Encoder Index -	1st
9	+/- 10V Analog Out	1st

Analog Input Connections P8 20-pin box header

Pin	*Signal*
1	GND
2	Analog GND
3	Clock 0
4	Analog in 0
5	-12 V
6	Analog in 1
7	+12 V
8	Analog in 2
9	+5 V

10	Step Pulse +	1st	10	Analog in 3
11	Step Pulse -	1st	11	Gate 0
12	Step Direction +	1st	12	Analog in 4
13	Step Direction -	1st	13	Out 0
14	GND	2nd	14	Analog in 5
15	5V	2nd	15	Out 1
16	Encoder A +	2nd	16	Analog in 6
17	Encoder A -	2nd	17	Out 2
18	Encoder B +	2nd	18	Analog in 7
19	Encoder B -	2nd	19	GND
20	Encoder Index +	2nd	20	Analog GND
21	Encoder Index -	2nd		
22	+/- 10V Analog Out	2nd		
23	Step Pulse +	2nd		
24	Step Pulse -	2nd		
25	Step Direction +	2nd		
26	Step Direction -	2nd		

Note: Two motors of same type can be connected to each motor header

DSPpro-VME
Dedicated I/O Connections
50-pin box headers

Pin	*Signal*	*P2 Axis*	*P3 Axis*
1	In-Position Out	3	7
3	Amp Enable Out	3	7
5	In-Position Out	2	6
7	Amp Enable Out	2	6
9	In-Position Out	1	5
11	Amp Enable Out	1	5
13	In-Position Out	0	4
15	Amp Enable Out	0	4
17	Amp Fault Input	3	7
19	Home Input	3	7
21	NEG Limit Input	3	7
23	POS Limit Input	3	7
25	Amp Fault Input	2	6
27	Home Input	2	6
29	NEG Limit Input	2	6
31	POS Limit Input	2	6
33	Amp Fault Input	1	5
35	Home Input	1	5
37	NEG Limit Input	1	5
39	POS Limit Input	1	5
41	Amp Fault Input	0	4
43	Home Input	0	4
45	NEG Limit Input	0	4
47	POS Limit Input	0	4
49	+5V		

Note: Even numbered pins are grounds and pin 49 is +5V.

DSPpro-VME P1 User I/O connections
50-pin Opto-22 compatible

Pin	*Signal*
1	PC Interrupt Input
3	DSP Interrupt Input
5	I/O Line C-5
7	I/O Line C-4
9	I/O Line C-3
11	I/O Line C-2
13	I/O Line C-1
15	I/O Line C-0
17	I/O Line B-7
19	I/O Line B-6
21	I/O Line B-5
23	I/O Line B-4
25	I/O Line B-3
27	I/O Line B-2
29	I/O Line B-1
31	I/O Line B-0
33	I/O Line A-7
35	I/O Line A-6
37	I/O Line A-5
39	I/O Line A-4
41	I/O Line A-3
43	I/O Line A-2
45	I/O Line A-1
47	I/O Line A-0
49	+5V

Note: Each I/O port (A,B,C) can be defined as inputs or output in groups of eight.

A.3 DSPpro-Serial Connector Layout

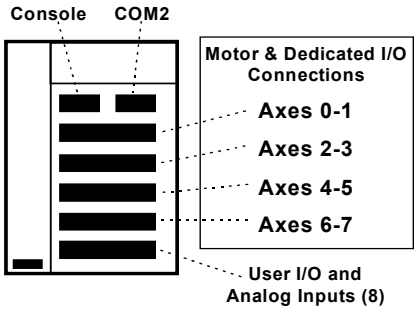

Figure A-3. Connector Locations -DSPpro-Serial

A.4 DSPpro-Serial Connector Pinouts

DSPpro-Serial P1
(Axes 0 and 1)

Pin	*Signal*	*Pin*	*Signal*
1	+5V	26	+5V
2	GND	27	GND
3	Encoder A(0) +	28	Encoder A(1) +
4	Encoder A(0) -	29	Encoder A(1) -
5	Encoder B(0) +	30	Encoder B(1) +
6	Encoder B(0) -	31	Encoder B(1) -
7	Encoder Index(0) +	32	Encoder Index(1) +
8	Encoder Index(0) -	33	Encoder Index(1) -
9	+/- 10V Analog Out(0)	34	+/- 10V Analog Out(1)
10	Step Pulse(0) +	35	Step Pulse(1) +
11	Step Pulse(0) -	36	Step Pulse(1) -
12	Direction(0) +	37	Direction(1) +
13	Direction(0) -	38	Direction(1) -
14	None	39	None
15	Amp Enable(0)	40	Amp Enable(1)
16	In Position(0)	41	In Position(1)
17	Amp Fault(0)	42	Amp Fault(1)
18	Home Input(0)	43	Home Input(1)
19	Positive Limit(0)	44	Positive Limit(1)
20	Negative Limit(0)	45	Negative Limit(1)
21	Opto +Volt	46	Opto +Volt
22	Opto Ground	47	Opto Ground
23	None	48	None
24	None	49	None
25	None	50	None

DSPpro-Serial P2
(Axes 2 and 3)

Pin	*Signal*	*Pin*	*Signal*
1	+5V	26	+5V
2	GND	27	GND
3	Encoder A(2) +	28	Encoder A(3) +
4	Encoder A(2) -	29	Encoder A(3) -
5	Encoder B(2) +	30	Encoder B(3) +
6	Encoder B(2) -	31	Encoder B(3) -
7	Encoder Index(2) +	32	Encoder Index(3) +
8	Encoder Index(2) -	33	Encoder Index(3) -
9	+/- 10V Analog Out(2)	34	+/- 10V Analog Out(3)
10	Step Pulse(2) +	35	Step Pulse(3) +
11	Step Pulse(2) -	36	Step Pulse(3) -
12	Direction(2) +	37	Direction(3) +
13	Direction(2) -	38	Direction(3) -
14	None	39	None
15	Amp Enable(2)	40	Amp Enable(3)
16	In Position(2)	41	In Position(3)
17	Amp Fault(2)	42	Amp Fault(3)
18	Home Input(2)	43	Home Input(3)
19	Positive Limit(2)	44	Positive Limit(3)
20	Negative Limit(2)	45	Negative Limit(3)
21	Opto +Volt	46	Opto +Volt
22	Opto Ground	47	Opto Ground
23	None	48	None
24	None	49	None
25	None	50	None

DSPpro-Serial P3
(Axes 4 and 5)

Pin	*Signal*	*Pin*	*Signal*
1	+5V	26	+5V
2	GND	27	GND
3	Encoder A(4) +	28	Encoder A(5) +
4	Encoder A(4) -	29	Encoder A(5) -
5	Encoder B(4) +	30	Encoder B(5) +
6	Encoder B(4) -	31	Encoder B(5) -
7	Encoder Index(4) +	32	Encoder Index(5) +
8	Encoder Index(4) -	33	Encoder Index(5) -
9	+/- 10V Analog Out(4)	34	+/- 10V Analog Out(5)
10	Step Pulse(4) +	35	Step Pulse(5) +
11	Step Pulse(4) -	36	Step Pulse(5) -
12	Direction(4) +	37	Direction(5) +
13	Direction(4) -	38	Direction(5) -
14	None	39	None
15	Amp Enable(4)	40	Amp Enable(5)
16	In Position(4)	41	In Position(5)
17	Amp Fault(4)	42	Amp Fault(5)
18	Home Input(4)	43	Home Input(5)
19	Positive Limit(4)	44	Positive Limit(5)
20	Negative Limit(4)	45	Negative Limit(5)
21	Opto +Volt	46	Opto +Volt
22	Opto Ground	47	Opto Ground
23	None	48	None
24	None	49	None
25	None	50	None

DSPpro-Serial P4
(Axes 6 and 7)

Pin	*Signal*	*Pin*	*Signal*
1	+5V	26	+5V
2	GND	27	GND
3	Encoder A(6) +	28	Encoder A(7) +
4	Encoder A(6) -	29	Encoder A(7) -
5	Encoder B(6) +	30	Encoder B(7) +
6	Encoder B(6) -	31	Encoder B(7) -
7	Encoder Index(6) +	32	Encoder Index(7) +
8	Encoder Index(6) -	33	Encoder Index(7) -
9	+/- 10V Analog Out(6)	34	+/- 10V Analog Out(7)
10	Step Pulse(6) +	35	Step Pulse(7) +
11	Step Pulse(6) -	36	Step Pulse(7) -
12	Direction(6) +	37	Direction(7) +
13	Direction(6) -	38	Direction(7) -
14	None	39	None
15	Amp Enable(6)	40	Amp Enable(7)
16	In Position(6)	41	In Position(7)
17	Amp Fault(6)	42	Amp Fault(7)
18	Home Input(6)	43	Home Input(7)
19	Positive Limit(6)	44	Positive Limit(7)
20	Negative Limit(6)	45	Negative Limit(7)
21	Opto +Volt	46	Opto +Volt
22	Opto Ground	47	Opto Ground
23	None	48	None
24	None	49	None
25	None	50	None

DSPpro-Serial P5
(User I/O, Analog Inputs, Counter/Timer)

Pin	*Signal*	*Pin*	*Signal*
1	+5V	26	GND
2	User I/O PA0	27	Clock 0
3	User I/O PA1	28	-12 Volt
4	User I/O PA2	29	+12 Volt
5	User I/O PA3	30	Opto +Volt
6	User I/O PA4	31	Gate 0
7	User I/O PA5	32	Out 0
8	User I/O PA6	33	Out 1
9	User I/O PA7	34	Out 2
10	User I/O PB0	35	Opto Ground
11	User I/O PB1	36	Analog In 0
12	User I/O PB2	37	Analog Ground
13	User I/O PB3	38	Analog In 1
14	User I/O PB4	39	Analog Ground
15	User I/O PB5	40	Analog In 2
16	User I/O PB6	41	Analog Ground
17	User I/O PB7	42	Analog In 3
18	User I/O PC0	43	Analog Ground
19	User I/O PC1	44	Analog In 4
20	User I/O PC2	45	Analog Ground
21	User I/O PC3	46	Analog In 5
22	User I/O PC4	47	Analog Ground
23	User I/O PC5	48	Analog In 6
24	User I/O PC6	49	Analog Ground
25	User I/O PC7	50	Analog In 7

DSPpro-Serial P8 (CONSOLE)

Pin	*Signal*	*Pin*	*Signal*
1	CD	6	DSR
2	RXD	7	RTS
3	TXD	8	CTS
4	DTR	9	RI
5	Ground		

DSPpro-Serial P9 (COM)

Pin	*Signal*	*Pin*	*Signal*
1	None	6	None
2	TxD B	7	None
3	RxD B	8	None
4	None	9	Ground
5	None		

DSPpro-Serial P100 (Power Input)

Pin	*Signal*	*Pin*	*Signal*
1	Ground In	2	+24 Volts In

Appendix B: SPECIFICATIONS

B.1 DSPpro-VME specifications

Specifications marked with a (*) are optional at no cost in volume.

System Processors

- Intel 386EX, 25 MHz microprocessor, MS-DOS 6.22
- 387 math co-processor
- Analog Devices 40 MHz DSP

Computer Interface

- VME compatible: A24, D16/D08 (EO) DTB slave interrupter I(1) - I(7) RORA, vector D08 (0)
- 8 or 16 bit data transfers
- Switch-selectable address
- Dual-port memory interface

Communications Interface

- RS-232 serial ports (2)
- Speed: 115 kbps maximum

Memory

- 1 Mbyte flash memory
- 1 Mbyte RAM

Motion Control Features

- Point-to-point motion
- Coordinated motion
- Electronic gearing and camming
- Sinusoidal commutation *
- Sinusoidal encoder interpolation *
- Feed speed override
- Dual-loop control
- Tangential following *
- High-speed registration
- Hundreds of other functions

Software Development Tools

- MEI standard C-function libraries (over 250 C-functions available)
- Compilers: Microsoft, Borland
- Host support software: dual-port memory toolkit or serial toolkit

Motion Profiles

- Trapezoidal profile
- S-curve profile
- Parabolic profile
- Custom (user-defined)

Kinematic Ranges

- Position: 32-bit (±2.15 billion counts)
- Velocity: 32-bit (±6.7 million
- Acceleration: 32-bit (±13.4 billion counts/sec2 at 2 kHz sampling)

User I/O

- 24-44 user I/O TTL
- Configurable as inputs or outputs

Dedicated I/O (per axis)

- 6 TTL per axis
- Inputs: positive and negative limits, home, amp-fault
- Outputs: in-position, amp-enable

Servo Loop Update Rate

- User-programmable rate
- Maximum: 10 kHz (1 axis), 3 kHz (4 axes), 1.6 kHz (8 axes)
- Default: 1.25 kHz

Servo Output

- ±10V DC at 16-bit resolution
- ±18 mA current
- Wide range dynamic PID control with VFF and AFF
- Friction compensation
- High inertia compensation

Step Output

- Maximum step frequency (open or closed loop): 375kHz
- Single-ended or differential outputs
- ±20 mA current
- Step/direction
- Clock up/clock down *
- Pulse width: 50% duty cycle

Analog Inputs

- 12-bit resolution, ± ½ LSB linearity
- 8 channel multiplexed inputs with track-and-hold
- Software configurable for 4-channel differential mode
- 0-5 V unipolar, ± 2.5 V bipolar

Position Feedback

- Incremental encoder: 5.0 MHz, single-ended or differential
- Encoder illegal state checking and broken wire detection
- RS-422 line receivers/digital filtering
- Analog position
- Temposonics direct connection *

Power Requirements

- +5 V Icc = 1A max
- +12V Icc = 10mA max
- -12V Icc = 20mA max

Environmental Conditions

- Operating temperature: 0-50 degrees C
- Humidity: 20-95% RH, non-condensing

B.2 DSPpro-Serial specifications

Specifications marked with a (*) are optional at no cost in volume.

System Processors

- Intel 386EX, 25 MHz microprocessor, MS-DOS 6.22
- 387 math co-processor
- Analog Devices 40 MHz DSP

Communications Interface

- RS-232 serial ports (2)
- Speed: 115 kbps maximum

Memory

- 1 Mbyte flash memory
- 1 Mbytes RAM

Motion Control Features

- Point-to-point motion
- Coordinated motion
- Electronic gearing
- Electronic camming
- Sinusoidal commutation *
- Sinusoidal encoder interpolation *
- Feed speed override
- Dual-loop control
- Tangential following *
- High-speed registration
- Hundreds of other features

Software Development Tools

- MEI standard C-function libraries (over 250 C-functions available)
- Compilers: Microsoft, Borland
- Host support software: serial toolkit

Motion Profiles

- Trapezoidal profile
- S-curve profile
- Parabolic profile
- Custom (user-defined)

Kinematic Ranges

- Position: 32-bit (±2.15 billion counts)
- Velocity: 32-bit (±6.7 million counts/sec at 2 kHz sampling)
- Acceleration: 32-bit (±13.4 billion counts/sec2 at 2 kHz sampling)

Servo Output

- ±10V DC at 16-bit resolution
- ±18 mA current
- Wide range dynamic PID control with VFF and AFF
- Friction compensation
- High inertia compensation

Servo Loop Update Rate

- User-programmable rate
- Maximum: 10 kHz (1 axis), 3 kHz
- (4 axes), 1.6 kHz (8 axes)
- Default: 1.25 kHz

Step Output

- Maximum step frequency (open or closed loop): 375kHz
- Single-ended or differential outputs
- ±20 mA current
- Step/direction
- Clock up/clock down *
- Pulse width: 50% duty cycle

Analog Inputs

- 12-bit resolution, ± 1/2 LSB linearity
- 8 channel multiplexed inputs with track-and-hold
- Software configurable for 4-channel differential mode
- 0-5 V unipolar, ± 2.5 V bipolar

Position Feedback

- Incremental encoder: 5.0 MHz, single-ended or differential
- Encoder illegal state checking and broken wire detection
- RS-422 line receivers/digital filtering
- Analog position
- Temposonics: direct connection *

Dedicated I/O (per axis)

- 6 opto-isolated per axis
- Inputs: positive and negative limits, home, amp-fault
- Outputs: in-position, amp-enable
- 5-24 V logic

User I/O

- 8 opto-isolated inputs
- 16 opto-isolated outputs
- 5-24 V logic

Mechanical Specifications

- Dimensions: 2" H x 5" W x 9" L
- Weight: 3.0 lbs (1.36 kg)

Power Requirements

- +24 V DC, 0.7 A (typical)

Environmental Conditions

- Operating temperature: 0-50 degrees C
- Humidity: 20-95% RH, non-condensing

Appendix C: Tuning Your System

C.1 General Description

The DSPpro controls servo motors by comparing the desired (command) and actual positions. The difference between the command and actual positions is defined as the position error. As the position error increases the motor control signals (analog output or step pulse rate) are increased to counteract the error. The computation of the value of the control output for a given position error is determined by the digital filter coefficients.

The process of adjusting these coefficients to provide the best control for a particular system of motors and loads is called "tuning." This section describes each digital filter coefficient and gives guidelines for system tuning.

There are generally two methods used for tuning closed loop digital control systems: calculation and trial and error. Calculation involves rather complex mathematics and precise knowledge of all of the system parameters such as motor and amplifier response, load inertia and friction. Texts on control systems provide methods for calculation of the tuning parameters for a large variety of applications (See references).

The trial and error method has the advantage that no knowledge of the control system parameters is necessary and no calculations are needed. A large number of trial parameters may be needed however to tune a system and some combinations of parameters may produce an unstable or runaway system.

An organized approach to searching for the best combination of tuning parameters helps shorten the tuning time while avoiding an unstable combination which may damage the system.

The methods of tuning described in this section rely upon the SETUP program, described in detail in chapter 5, "MEI Software." Please be sure to review this section before tuning your system.

C.1.1 The Digital Filter

The DSP calculates an axis' output (analog voltage or pulse rate) based on a PID servo control algorithm. The input to the PID algorithm is the current position error. The current position error equals the difference between the command position and the actual position. The actual position is controlled by the feedback device, and command position is controlled by the trajectory calculator. The PID algorithm is based on the following formula:

$$O_n = K^R (K_p*E_n + K_d*(E_n - E_{n-1}) + K_i*S_n + K_v*V_n + 64*K_a*A_n + K_f * M_n) + K_o$$

The subscripted $_n$ represent the sample period. The terms are defined:

$S_n = S_{n-1} + E_n$ if $-S_{max} < S_n < S_{max}$

S_{max} if $S_n > S_{max}$

$-S_{max}$ if $Sn < -S_{max}$

O_n = DAC output

K_p = proportional gain

K_i = integral gain

K_a = acceleration feed-forward

K_f = friction feed-forward

M_n = 0 or 1 based on the command velocity

A_n = command acceleration * 2^{-6}

S_{max} = maximum integrated error

K_R = overall scale factor

K_d = derivative gain

K_v = velocity feed-forward

K_o = static DAC offset

E_n = position error

V_n = command velocity

S_n = integrated error

Each coefficient can be adjusted using the SETUP program and is explained in the following sections.

C.2 Tuning Parameters

C.2.1 Proportional Gain (K_p)

This term determines the overall response of a system to position errors. A low Proportional Gain provides a system which is very stable (doesn't oscillate), has low stiffness and large position errors under load. A high Proportional Gain provides high stiffness and small position errors under load but may oscillate.

Typical gain values are 100-500 for velocity (voltage) controlled servos and 500-2000 for torque (current) controlled servos. Closed loop step systems are similar to velocity controlled servos.

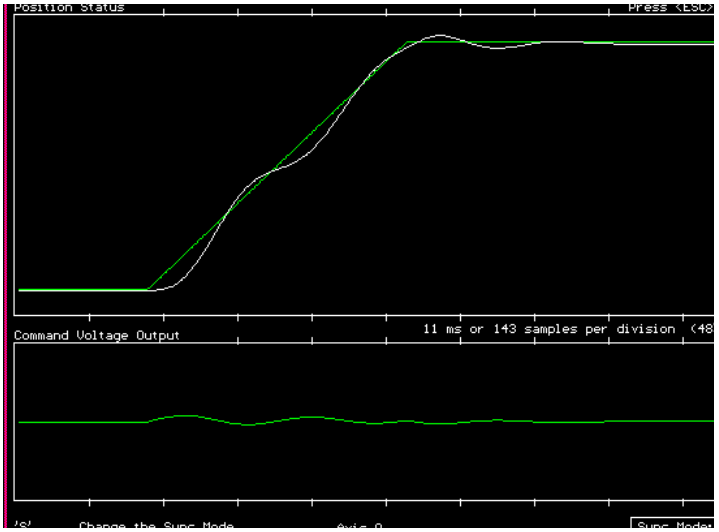

Figure C-1. Insufficient Proportional Gain

In the above plot, the motor (actual position) is not keeping up with the command position, yet the output voltage is not saturated. Clearly, more gain is required.

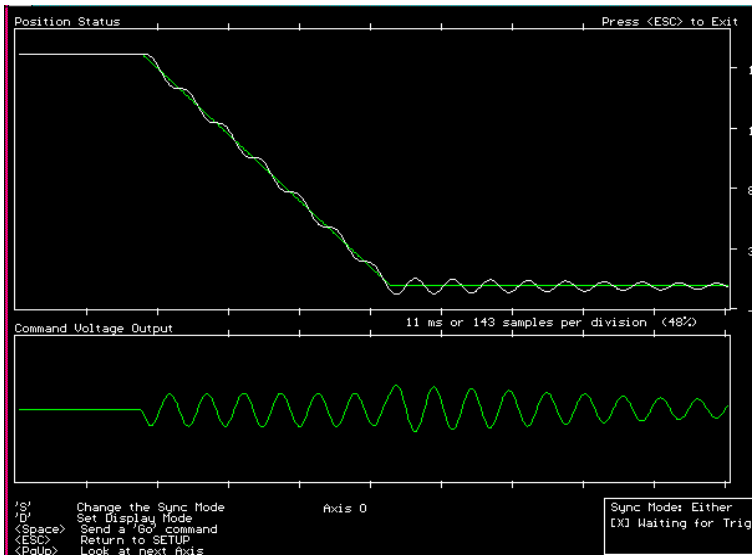

Figure C-2. Excessive Proportional Gain

Excessive proportional gain is characterized by oscillation. In some situations, damping (derivative gain) can be increased to help compensate.

C.2.2 Derivative Gain (K_d)

This term provides damping and stability to the system by preventing overshoot as the error changes. A low value for the Derivative Gain causes the system to have very fast response to changes in position error but may have overshoot or "ringing" after a step change in position. Large values of Derivative gain have slower step response but may allow higher Proportional Gain to be used without oscillation.

Typical values for Derivative Gain are roughly 2 times the proportional gain for velocity (voltage) controlled servos , i.e. 200-1000 and roughly 4 times the proportional gain for torque (current) controlled servos, i.e. 1000-8000.

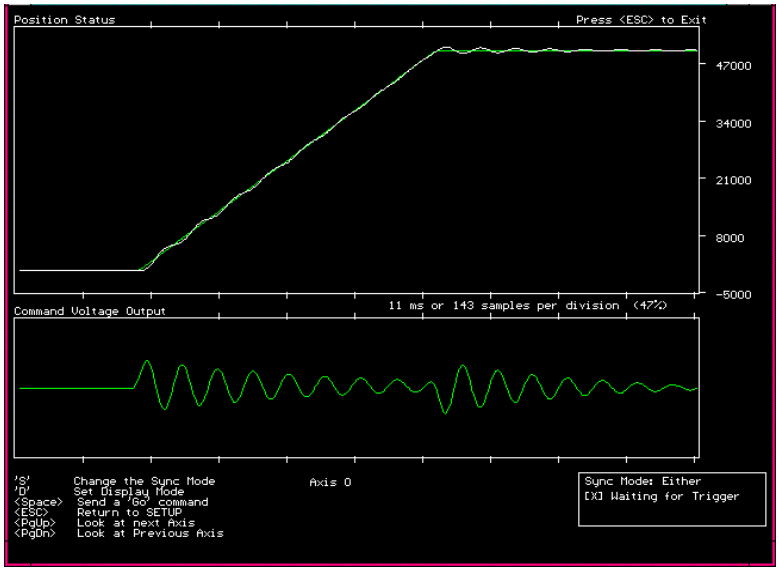

Figure C-3. Insufficient Derivative Gain

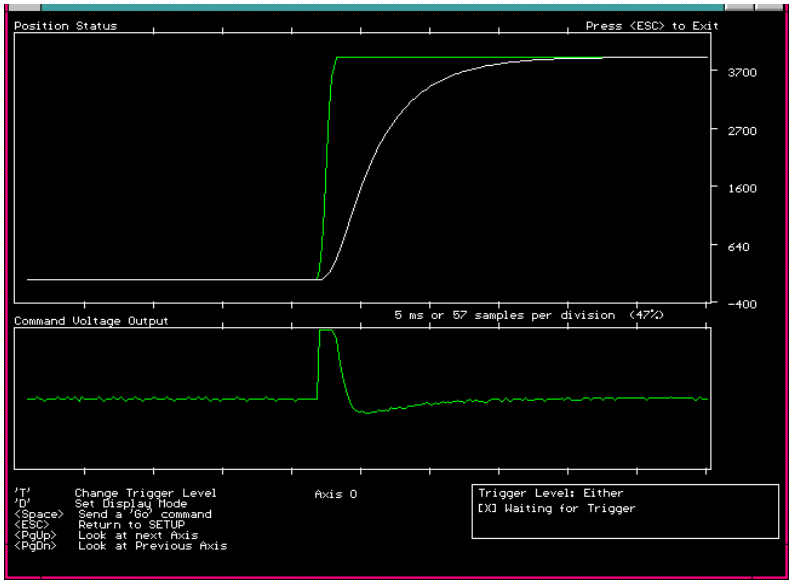

Figure C-4. Excessive Derivative Gain

C.2.3 Integral Gain (K_i)

Integral Gain helps the control system overcome static position errors caused by friction or loading by increasing the output value until the error goes to zero. A low or zero value for Integral Gain may have position errors at rest which depend on the static or frictional loads and the Proportional Gain. Increasing the Integral Gain can reduce these errors. If the Integral Gain is too large the system may "hunt" (oscillate at low frequency) about the desired position.

Typical integral gain values are between 16 and 64 and depend on the integration limit.

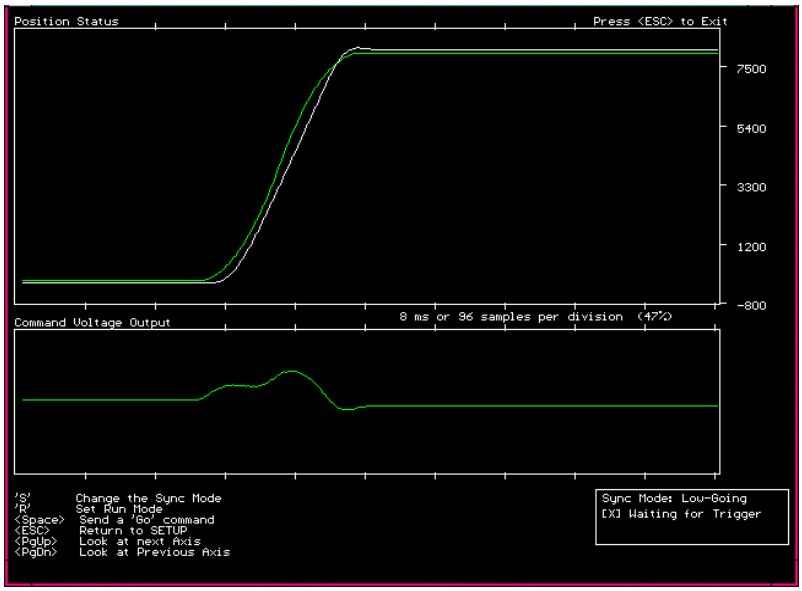

Figure C-5. Insufficient Integral Gain

Note that the integral gain may be inactive during motion is the axis is configured for an Integral mode of "Only Standing". To close the following error during motion, use the feed-forward filter terms. Also note that the integration limit affects the performance of the integral gain.

C.2.4 Velocity Feed Forward (K_v)

The velocity feed-forward term is very important when used with velocity controlled servos or closed-loop step motors. As the speed of a system increases the position error generally increases and therefore a higher output voltage or pulse rate is needed. The Velocity Feed-Forward term reduces the following error by increasing the controller output voltage at high speed. If the Velocity Feed-Forward term is too large the motor will try to travel ahead of the command position.

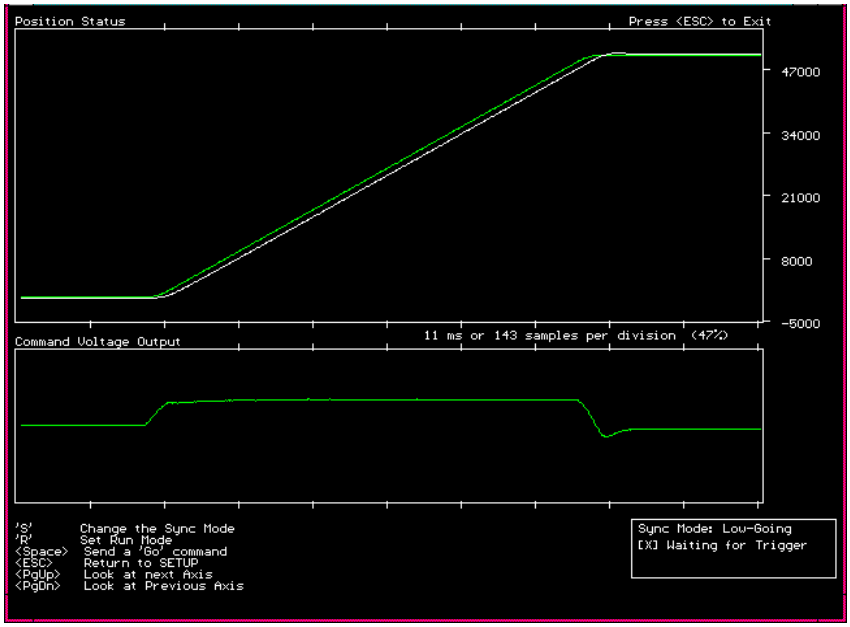

Figure C-6. Insufficient Velocity Feed-Forward

C.2.5 Acceleration Feed-Forward (K_a)

Acceleration feed-forward is used with torque (current) controlled servos. Systems with high inertial loads need more motor current to accelerate or decelerate than systems with light loads. The Acceleration Feed-Forward term causes the controller to increase the motor control signals during periods of acceleration and deceleration.

C.2.6 Integration Limit

The integration of position errors is limited to a fixed DAC output. This prevents the integrator from "windup" in the case of high static friction.

Note that the integration term can be set to active always or active when at rest (command velocity = 0) only.

C.2.7 Offset (K_o)

The offset term compensates for small variations in controller DAC outputs and amplifier offsets. An internal calibration (transparent to the user) is performed at the factory so that when the offset is zero the analog or pulse output is also zero.

If necessary, the CONFIG program can be used to re-calibrate the analog and step pulse output.

C.2.8 Shift

The shift parameter is used to calculate the overall scale factor K_R ($K_R = 2^{shift}$). The overall scale factor increases the accuracy of the other tuning parameters. Decreasing the shift by 1 will divide the equation for O_n by 2. In order to get the same voltage output from the PID, the gains

and feed forward terms must be doubled. This means that a gain of 10 must be changed to a gain of 20. This also means that a gain of 9.5 which before could not have been entered can now be entered as 19. In effect, the shift parameter allows the gains and feed forwards to be finely tuned.

C.2.9 Friction Feed Forward

The friction feed forward parameter simply adds a constant value to the DAC output when the command velocity is non-zero. The sign of the value applied to the DAC is equal to the sign of the command velocity multiplied by the friction feed forward term. The friction feed forward term is 16 bits and has a range from -32,768 to 32,767. Torque controlled motion systems with constant friction can benefit most from friction feed forward.

C.3 Tuning Closed-Loop Servos

The following steps provide a quick method for tuning for a stable system with minimal position errors:

Step 1: Set Proportional Gain

Step 2: Set Derivative Gain

Step 3: Iterate steps 1 and 2

Step 4: Set Integral Gain

Step 5: Set Velocity and Acceleration Feed-Forward

For new systems this sequence of steps should be performed twice; once with no motor load to provide a stable set of starting terms, and once with the motor loaded to fine-tune the initial parameters.

C.3.1 Step 1: Set Proportional Gain (K_p)

Start with all of the gains except the offset (K_p, K_d, K_i, K_v, and K_a) at 0. The motor should not turn and the shaft should be free. If the shaft turns the amplifier offset should be adjusted to reduce the motor torque to zero. Set the Proportional Gain (K_p) to 1. Watch the position error on the F2 window as you change the gain. The position error should reduce to a few hundred counts.

If the motor runs away or the shaft still turns freely, verify the wiring using the procedure described in Installation and Setup Manual. Increase the gain by factors of 2 until the system begins to hum or oscillate. Reduce the Proportional Gain to 1/2 the value that first produces oscillation.

C.3.2 Step 2: Set the Derivative Gain

Start with a K_d of 100. Increase the value of K_d by factors of 2. Set the K_d value to the smallest value which produces no overshoot in the step response.

C.3.3 Step 3: Iterate Steps 1 and 2

With a derivative Gain high enough to eliminate overshoot, increased the Proportional Gain until the system becomes unstable. Then increase the Derivative Gain again and try to reduce overshoot and ringing. Eventually it will be impossible to eliminate the overshoot by raising derivative gain. At this point the Proportional Gain should be reduced to provide the desired motion response. Remember that some overshoot is acceptable in systems which are being tuned for maximum speed.

C.3.4 Step 4: Set Integral Gain (K_i)

The best way to tune the Integral Gain is to observe the static error at the end of a move as the K_i term is increased. Using the two-point motion window, set the following motion parameters:

Parameter	*Setting*
Delay	2
Position 1	0
Position 2	20000
Velocity	10000
Acceleration	10000

Start the motion and observe the position error between moves. Gradually increase K_i until the final position error is 1 or 2 encoder counts. As you increase K_i above this level, watch for oscillation at the beginning or end of the motion. If oscillation occurs reduce the value of K_i.

C.3.5 Step 5: Set Velocity and Acceleration Feed Forward

Set the motion parameters in the **Motion/Two-Point Motion** window for a move which takes 5 to 10 seconds using the highest desired speed and acceleration. Notice the position error during the constant speed portion of the motion. Increase the K_v term until the constant velocity error is reduced to the desired level. Use the same method to adjust the K_a term watching the error during the acceleration and deceleration portions of the motion (look quickly if the acceleration time is short).

C.4 Tuning Closed-Loop Steps

The following steps provide a quick method for tuning for a stable system with minimal position errors:

Step 1: Set Proportional Gain

Step 2: Set Velocity Feed-Forward and Acceleration Feed-Forward

Step 3: Set Integral Gain

For new systems this sequence of steps should be performed twice; once with no motor load to provide a stable set of starting terms, and once with the motor loaded to fine-tune the initial parameters.

The Derivative term should be set to zero.

C.4.1 Step 1: Set Proportional Gain (K_p)

The Proportional Gain is dependent upon the ratio between the number of encoder counts and the number of steps (or microsteps) per revolution of the motor. The greater the number of steps (or microsteps) per encoder count, the larger the Proportional Gain. Most often the Proportional Gain will be between 20 and 400. Start with the Proportional Gain at 20 and all other gains (K_p, K_d, K_i, K_v, and K_a) at 0. Try some two-point motions and increase the Proportional Gain until the motor stalls. Then reduce the Proportional Gain to 1/2 the value that caused the motor to stall.

C.4.2 Step 2: Set Velocity and Acceleration Feed Forward

Set the motion parameters in the **Motion/Two-Point Motion** window for a move which takes 5 to 10 seconds using a typical desired speed and acceleration. Notice the position error during the constant speed portion of the motion. Increase the K_v term until the constant velocity error is reduced to the desired level. Use the same method to adjust the K_a term watching the error during the acceleration and deceleration portions of the motion. The Acceleration Feed-Forward won't be needed for most systems.

C.4.3 Step 3: Setting the Integral Gain (K_i)

The best way to tune the Integral Gain is to observe the static error at the end of a move as the K_i term is increased. Using the two-point motion window, set the following motion parameters:

Parameter	*Setting*
Delay	2
Position 1	0
Position 2	20000
Velocity	10000
Acceleration	10000

Start the motion and observe the position error between moves. Gradually increase K_i until the final position error is 1 or 2 encoder counts. As you increase K_i above this level, watch for oscillation at the beginning or end of the motion. If oscillation occurs reduce the value of K_i.

INDEX

A

Acceleration feed forward, 29, 100, 102, 103
Actual position, 11, 93
Analog inputs, 63
Application development, 65
  Testing your application program, 66
  Writing your application, 66
Applications disk, 1
Axis configuration, 30
Axis status, 35

B

Brush servos
  wiring, 47
Brushless servo wiring, 48

C

C function library, 65
Closed-loop servos
  tuning, 93, 101
Closed-loop steps, 51
  tuning, 93, 102
  wiring, 52
Command position, 11, 93
Communication
  host/controller, 9
Configure menu, 27
Current control, 46

D

Dedicated I/O, 57, 62
Dedicated I/O window, 36
Derivative gain, 29, 97, 102
Digital filter, 93
DSPpro-Serial
  cables, 41
  connection configuration, 43
  connector pinouts, 76
  direction pulse synchronization, 51
  motor signal headers, 45
  servo motor wiring, 46
DSPpro-VME
  cables, 41
  connection configuration, 42
  connector pinouts, 74
  direction pulse synchronization, 51
  header locations, 44
  motor signal headers, 73
  motor signal pinouts, 44
  servo motor wiring, 46
  switch locations, 17
  switch settings, 17, 18
Dual-loop wiring, 53

E

E-mail address, 1
Encoder input, 46
Errors
  motors turn only one direction, 32

F

Frame buffer, 13
Friction feed forward, 30, 101

H

Home switch wiring, 61

I

I/O base address window, 27
I/O Wiring
  Dedicated I/O wiring, 60
  DSPpro-PC and DSPpro-VME, 57
  DSPpro-Serial, 60
  User I/O Wiring, 60
Installing hardware, 3
  DSPpro-PC, 3
  DSPpro-Serial, 4
  DSPpro-VME, 3
Installing software, 4
Installing the DSPpro-PC, 16
Installing the DSPpro-Serial, 18
Installing the DSPpro-VME, 17, 18
Integral gain, 28, 98, 102, 103
Integration limit, 100
Integration limit, 29

L

Limit switch configuration window, 33
Limit switch wiring, 61

M

Memory mapping, 15
Motion menu, 37
Mouse support, 22

O

Offset, 29, 100

Open-loop step motors, 49
optical isolation, 61, 63
Opto-isolation, 63
Output limit, 30

P

PID parameter ranges, 30
Position error, 11, 93
Position status window, 35
Proportional gain, 28, 94, 101, 103
Pulse rate, 31

Q

Quadrature, 46
Quick Start, 2, 3, 68

R

RCONSOLE, 5–7, 22, 37–40
REMSVR, 5–7, 21, 22, 38–40, 66–70, 72
RS-232 serial ports, 4, 18, 67, 83, 86, 89

S

Sample programs, 1
Saving default parameters, 23
Servo motors
  brush, 46
  brushless, 48
  configuring axes, 6, 25, 27
  control algorithm, 11
  DAC output, 12
  functions, 25
  step-and-direction controlled, 49
  wiring, 46, 47
SETUP program, 2, 5–7, 21, 23, 26
  axis configuration window, 30
  axis status window, 35
  buttons, 23
  configure menu, 27
  dedicated I/O window, 36
  defaults, 23
  function keys, 23
  I/O base address window, 27
  limit switch configuration window, 33
  motion menu, 37
  mouse and trackball support, 22
  position status window, 35
  screens, 25
  software limits window, 34
  status menu, 34
  tuning parameters window, 27
Shift, 101
Software & firmware updates, 2
Software limits window, 34
Speed, 31
Status menu, 34
Step motors
  closed-loop, 51
  control, 12
  open-loop, 49

T

Technical Support, 1, 49
Torque mode, 46
Trackball support, 22
Trajectory, 11
Tuning, 93
  closed-loop servos, 101
  closed-loop steps, 102
  parameters, 94
Tuning parameters window, 27
Tuning your system, 5, 29
  SETUP program, 1, 21
  tuning parameters, 5–7, 23, 25, 27, 30

U

User I/O, 57, 62

V

Velocity feed forward, 29, 99, 102, 103
Velocity mode, 46
VFC, 12
Voltage control, 46
Voltage to frequency converter, 12

W

Wiring your system, 4
  Amplifier enable wiring, 63
  Analog input wiring, 63
  dual-loop control, 53
  Home and limit switch wiring, 61
  I/O Wiring, 57
  Interferometers, 53
  motor wiring, 41
  servo motors, 46
  step motors, 49