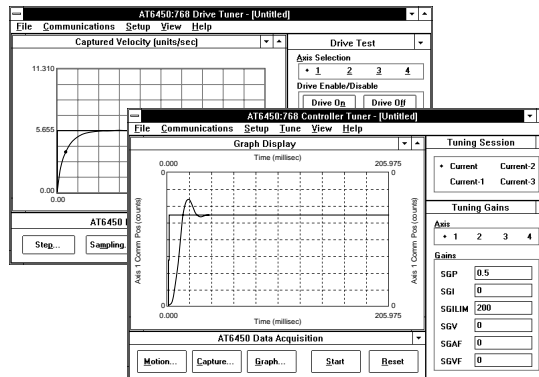


# Compumotor

## Servo Tuner™ User Guide

*Servo Tuning Software  
for 6000 Series Controllers*



Compumotor Division  
Parker Hannifin Corporation  
p/n 88-014249-01B November, 1994



# Servo Tuner™

The information in this document is subject to change without notice and does not represent a commitment on the part of Parker Hannifin Corporation. Servo Tuner is furnished under a license agreement or nondisclosure agreement, and may be installed and used only in accordance with the terms of the agreement.

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without permission of Parker Hannifin Corporation.

Motion Architect is a registered trademark and Servo Tuner is a trademark of Parker Hannifin Corporation.

Microsoft and MS-DOS are registered trademarks and Windows is a trademark of Microsoft Corporation.

© Copyright 1993-4 by Parker Hannifin Corporation.

All rights reserved.

Printed in the United States of America.



## WARNING



The tuning process requires operation of your system's electrical and mechanical components. Therefore, you should test your system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

## Technical Assistance

*Contact your local automation technology center, or ...*

North America: Compumotor Division of Parker Hannifin  
5500 Business Park Drive  
Rohnert Park, CA 94928  
Telephone: (800) 358-9070  
Fax: (707) 584-3793  
BBS: (707) 584-4059

Europe: Parker Digiplan  
21 Balena Close  
Poole, Dorset  
England BH17 7DX  
Telephone: 0202-690911  
Fax: 0202-600820

# Change Summary

## *Servo Tuner User Guide*

Revision B

This document, p/n 88-014249-01 **B** (released in November 1994), supersedes 88-014249-01 **A**. *The primary changes are in support of Servo Tuner version 3.1.*

### Summary of Technical Changes

Topic	Description of Change	See Page
Feedback Source for Drive Tuner Module	Under the Setup pull-down menu, you may type in the encoder or ANI resolution in the Feedback dialog box. The typical encoder resolution is 4000 counts/rev (check your encoder specifications). The resolution for ANI feedback is 819 counts/volt.	8 & 9
Product Revision for Controller Tuner Module	You no longer need to supply the revision number of your 6000 Series product in the System dialog box under the Setup menu. Instead, when you launch the Controller Tuner module, the product revision is automatically read from the product.	14 & 18
Resolver Feedback	The Controller Tuner module now supports the resolver feedback from the recently-released APEX6154 product. The resolver's resolution is 4096 counts/rev.	19, 25, 32
Torque Drive Users	<b>TIP:</b> Although designed for tuning velocity drives only, you can use Drive Tuner to verify drive connections and feedback polarity if you are using a torque drive (see steps 4 & 5 in the Drive Tuning Procedure).	9



# C O N T E N T S

---

---

## Welcome

Introduction.....	2
Before You Begin.....	2
User Guide Organization .....	2
Reference Documentation .....	3
Hardware and Software Requirements.....	3
Installation Procedure.....	4
Launching Servo Tuner Modules .....	5
Working with Servo Tuner Files .....	5

## Chapter 1—Drive Tuner

Drive Tuner Basics.....	8
Drive Tuning Procedure (velocity drives).....	9

## Chapter 2—Controller Tuner

Controller Tuner Basics.....	14
Motion Profile Setup .....	15
Data Capture Setup.....	15
Graph Setup .....	16
Controller Tuning Procedure.....	17
Tuning Scenario .....	24

## Appendix A—Servo Tuning Principles

Servo System Terminology.....	27
Servo Tuning Terminology.....	27
Position Variable Terminology.....	28
Servo Response Terminology.....	30
Tuning-Related Software Commands .....	32
Servo Control Techniques .....	34
Proportional Feedback Control (SGP).....	35
Integral Feedback Control (SGI).....	35
Velocity Feedback Control (SGV).....	37
Velocity Feedforward Control (SGVF).....	38
Acceleration Feedforward Control (SGAF).....	38

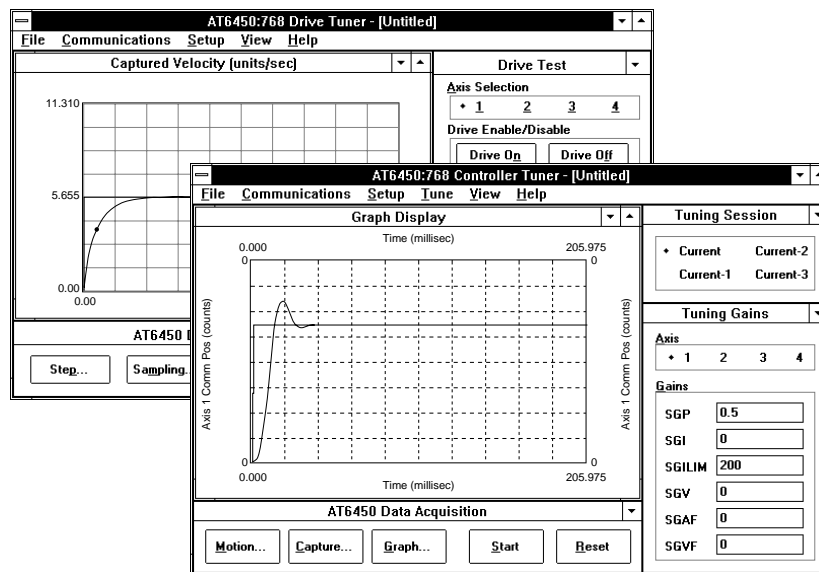
## Appendix B—Target Zone Mode.....

Index.....	43
------------	----



# W E L C O M E

*to Servo Tuner*



## Introduction

---

The Servo Tuner™ option for Motion Architect® is a Microsoft® Windows™ based program comprising two utilities designed to help you tune your motion control servo system:

- Drive Tuner—Graphically tune and set up your velocity drive system without the position loop enabled. *This module is not designed for use with torque drives, packaged controller/drive products, or servo valves.*
- Controller Tuner—Tune the servo controller's position control loop. The graphical feedback assists you in analyzing exactly what is happening with the motion of the system. *When using a velocity drive, you should finish the drive tuning procedures with the Drive Tuner module before proceeding to the Controller Tuner module.*

## Before You Begin

---



### WARNING



The tuning process requires operation of your system's electrical and mechanical components. Therefore, you should test your system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

**EMERGENCY SHUTDOWN:** You should be prepared to shut down the drive(s) during the tuning process (for instance, if the system becomes unstable or experiences a runaway). You can use the **ENBL** input (disconnect it from ground) to disable the controller's analog output signal to the drive. An alternative is to issue the @DRIVEØ command to the controller over the communication interface, but this requires connecting a shutdown output to the drive. If the drive does not have a shutdown input, use a manual emergency stop switch to disable the drive's power supply.

## User Guide Organization

- Chapter 1 (*Drive Tuner*) is an overview of the features in the Drive Tuner module. A recommended velocity drive tuning procedure is provided.
- Chapter 2 (*Controller Tuner*) is an overview of the features in the Controller Tuner module and provides a recommended tuning procedure. A tuning scenario is also provided to demonstrate the basic tuning process.
- Appendix A is an overview of servo tuning principles and terminology.
- Appendix B describes the Target Zone Mode, a feature that lets you precisely define the move completion criteria relative to distance and velocity.

The tuning processes described in this document are provided as guidelines; you may need to alter these processes to fit your particular tuning requirements.

## Reference Documentation

- *Motion Architect User Guide*
- 6000 controller's user guide
- On-line documentation (accessed from the **Help** pull-down menu):
  - Servo Tuner Help. Select **H**elp for help, **K**ey help, or **H**elp index, or press the F1 key.
  - **6000 C**ommands (Controller Tuner only). Displays the Command Dialog Box, from which you can look up commands, edit them, and insert them into the active **C**omm terminal emulator window.
  - **6000 S**oftware Reference (Controller Tuner only). Brings you directly to the Contents menu of the on-line *6000 Series Software Reference Guide*. This is a valuable resource for detailed command descriptions and programming guidelines.
  - **6000 F**ollowing Reference (Controller Tuner only). Brings you directly to the Contents menu of the on-line *6000 Series Following User Guide*.

## Hardware and Software Requirements

Servo Tuner requires the resources listed below (this does not include memory requirements for optional add-on modules).

- **Motion Architect (rev 2.2 or later) must first be installed.**
- Microsoft Windows release 3.1 or later
- IBM/compatible with at least 2 MB of RAM
- At least 1 MB of hard disk space

## Installation Procedure

### Copy Protection

The Servo Tuner diskette is copy protected to authorize a total of two installations. The diskette cannot be copied (no backups possible). After installation, Servo Tuner cannot be copied from your hard drive to another hard drive. If you install Servo Tuner on your hard drive and then find that you need to use Servo Tuner on another computer, you must first "uninstall" it from your present hard drive.

To uninstall Servo Tuner, repeat installation steps 2-5 below, and select **Uninstall** in the **Custom Installation** dialog box. This removes the authorization from your hard drive and returns it to the Servo Tuner diskette. Then you can use the Servo Tuner diskette to install Servo Tuner in another computer. To order additional copies, contact your local Automation Technology Center or distributor.

#### **Step 1**

If you have not already done so, install Motion Architect now. *Servo Tuner will not install unless Motion Architect is already installed.* Refer to the *Motion Architect User Guide* for installation instructions.

#### **Step 2**

Insert the Servo Tuner diskette into floppy disk drive **A**.

#### **Step 3**

From the Program Manager, click **File** and choose **Run**. You can also do this from the File Manager. When the dialog box appears, type **a:setup**. When the Welcome screen appears, click **Continue**. After a short period, the registration screen will appear.

#### **Step 4**

In the registration box, enter your name, company name, and the serial number for your copy of Servo Tuner. (The serial number is located on the back of the Servo Tuner diskette.) After you enter the information, click **OK**.

Verify the information in the Registration Confirmation box. If everything is correct, click **Yes**. If there is an error, click **No** and correct the information.

#### **Step 5**

In the Custom Installation dialog box, you can specify the location where you want to install Servo Tuner, and you can select only those parts of Servo Tuner you wish to install.

Unless otherwise specified with the **Set Location** option, Servo Tuner will be installed in the Motion Architect sub-directory. This is recommended so that Servo Tuner can access the Motion Architect help system.

In the Installation Options area, select the parts of Servo Tuner you wish to install:

- **Drive Tuner Program** installs the files necessary to run Drive Tuner.
- **Controller Tuner Program** installs the files necessary to run Controller Tuner.
- **Controller Tuner Examples** installs sample tuner files that illustrate controller tuning.

After you have made the appropriate selections, click **Install** and follow the instructions in the Installing Authorization window.

If you need to **Uninstall** Servo Tuner, refer to the instructions in the Copy Protection notice above.

### **Step 6**

After the installation is complete, the Installation Complete dialog box appears. At this point, you have the options of reading the README file, running Motion Architect (you can then launch Drive Tuner or Controller Tuner from the **Utilities** menu), or returning to Windows.

## Launching Servo Tuner Modules

After Servo Tuner is installed on your hard drive, you can launch the Drive Tuner module or the Controller Tuner module from the **Utilities** pull-down menu in Motion Architect.

<b>P</b> roduct	<b>S</b> etup	<b>E</b> ditor	<b>T</b> erminal	<b>P</b> anel	<b>U</b> tilities	<b>H</b> elp
					<b>D</b> rive Tuner	
					<b>C</b> ontroller Tuner	

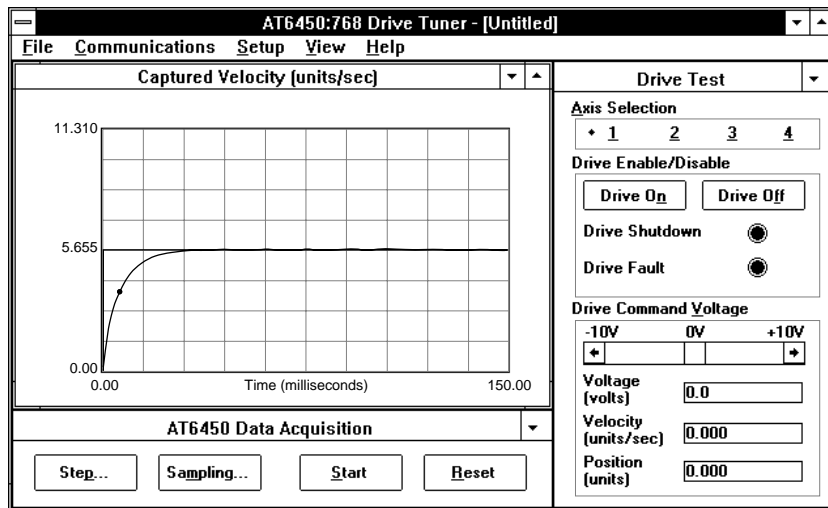
## Working with Servo Tuner Files

Each session of the Drive Tuner or Controller Tuner can be saved (.srv file) and then recalled for further analysis or modifications. When you save the session, the resulting 6000 code is saved to a program (.prg) file that you can edit in the Editor module or download in the Terminal or Panel modules. Instructions on using Motion Architect's main modules are provided in the *Motion Architect User Guide*.



# CHAPTER ①

## *Drive Tuner*



# Drive Tuner Basics

## Before You Begin

Before you begin the tuning process, make sure you have completed the system connection and test procedures provided in the *Installation* chapter of your 6000 Series controller's user guide.

The following is an overview of the Drive Tuner module's features. The illustration below shows the Drive Tuner that is accessed from the **Utilities** pull-down menu.

**Connect**—engages the communication link with the controller. Bus-based controllers must verify the **Board Address** and **Send Operating System**. Serial controllers must select a **Serial Port**.

**Drive**—Set the drive fault active level, enable/disable the drive fault input.

**Feedback**—Select the feedback device per axis and set the feedback device resolution (and scaling factor if required).

Under the **View** menu, you can open the Graph, Data Acquisition, and Drive Test displays (as shown in this illustration). Also available is the Cursor display. After the captured data is graphed, open the Cursor display and click on the graph. As you move the cross-hairs with the mouse or with the arrow keys, the Cursor display shows the fine velocity and time increments.

Select the axis for the current tuning session.

Press these buttons to enable or disable the drive with the shutdown output (SHTNO or SHTNC).

This LED turns red if the shutdown output is active.

This LED turns red if the drive fault input is activated (be sure to enable the drive fault input and set the correct fault level for your drive).

Move this control button (with the mouse or with the arrow keys) to adjust the analog output voltage. Notice the voltage, velocity and position values change as you move the button.

The **Step** button brings up a dialog box in which you can select the voltage and duration for the step output (commanded velocity) used to tune the dynamics of the system. In the graph, this is the red plot, against which the actual velocity (the blue plot) is compared. Also set the **Drive Gain** (this is the **Velocity** reading when the **Drive Command Voltage** is set to 1V).

The **Sampling** button brings up a dialog box in which you can select the data sampling period and the frequency of samples for the data gathering function.

The **Start** button initiates the step output and the data gathering function. After a short period, the data is displayed in the graph above.

The **Reset** button resets the controller to its default settings (shutdown outputs are activated, the drive fault input is disabled, and the command output is held to zero volts).

With the voltage set to 1V, record the **Velocity** reading and enter this value in the 1V = \_\_\_\_\_ units/sec data field when you set up the step output (in the **Step** dialog box).

**AT6450:768 Drive Tuner - [Untitled]**

File Communications Setup View Help

**Captured Velocity (units/sec)**

11.310  
5.655  
0.00

The data captured after pressing the **Start** button is displayed in this graph area.

Commanded Velocity (step output from the servo controller)

Time Constant (time taken to reach 63.2% of final velocity)

Actual Velocity (measured by the feedback device)

0.00 150.00

Time (milliseconds)

**AT6450 Data Acquisition**

Step... Sampling... Start Reset

**Drive Test**

Axis Selection  
1 2 3 4

Drive Enable/Disable  
Drive On Drive Off

Drive Shutdown  
Drive Fault

Drive Command Voltage  
-10V 0V +10V

Voltage (volts) 0.0

Velocity (units/sec) 0.000

Position (units) 0.000

## Drive Tuning Procedure *(velocity drives)*

---

The following is a recommended procedure for tuning a velocity drive. This procedure must be performed one axis at a time (**repeat steps 3-11 for each axis**).

*Drive Tuner is not designed for tuning torque drives, servo valves, or packaged 6000 Series drive/controller products.*

### CAUTION

To ensure safe operation of your system's mechanical components, you should perform this procedure while the load is **not** attached to the motor.

#### **Step 1**

Launch the Drive Tuner from the **Utilities** menu in Motion Architect.

#### **Step 2**

*This step assumes you have already selected your controller (see **Product menu** in Motion Architect).* Verify that there is a check mark next to **Connect** under the **Communications** menu; this indicates that the communication link to the controller is established.

#### **Step 3**

Under the **Setup** menu, select **Drive** and select the axis you wish to tune, select the drive fault level, and enable fault detection. For information on drive fault operation, refer to the DRFLVL command description in the on-line **6000 Software Reference**.

Also under the **Setup** menu, select **Feedback** and select the axis you wish to tune, the feedback device for that axis, and enter the feedback device's resolution (and scaling factor if required). Encoder resolution is typically 4000 counts/rev (check your encoder specifications); ANI resolution is 819 counts/volt.

#### **Step 4**

*(This step assumes you have connected the drive shutdown and drive fault signals between the controller and the drive.)* Check the Drive Shutdown and Drive Fault status in the **Drive Test** window. If the **Drive Shutdown** LED is red, press the **Drive On** button to enable the drive (LED turns green). If the **Drive Fault** LED is red but the drive checks out O.K., the problem could be an incorrect fault level (see Step 3 above), or a bad drive fault signal connection.

### Step 5

This step tests the polarity of the connections to the feedback device (encoder or ANI input) and the drive. Servo stability requires a direct correlation between the sign of the output voltage from the controller and the sign of the feedback device counts (i.e., a positive voltage to the drive must result in positive feedback device counts).

- a. Note the current **Voltage** and **Position** readings in the **Drive Test** window.
- b. Make the controller command a short-duration positive voltage to the drive. To do this, click once on the right arrow of the **Drive Command Voltage** slide bar to command +0.1V to the drive (note the sign of the voltage reading) and then click the left arrow once to stop the motor.

If the sign of the voltage reading is inverted, swap the CMD+ and CMD- connections to the drive (the CMD swap will work only with drives that accept differential output).

The position should increase due to the positive voltage output. If this is not true (position counts in the negative direction), use the appropriate corrective action described below:

Encoder... Swap the encoder A+ and A- connections at one end only.

ANI ..... Swap the ANI input positive and ground connections at one end only.

### Step 6

Using the **Drive Command Voltage** slide bar, set the analog output to the maximum voltage that your drive can accept, and then adjust the drive gain factor (sometimes called the *tach gain*) so that the drive's velocity just reaches its maximum value at the maximum input voltage.

#### EXAMPLE

Suppose your drive can run at a maximum velocity of 7000 rpm (116.67 rps). If the drive gain factor is 20 rps/V, then the drive will reach the maximum velocity (116.67 rps) when the controller's command output is only 5.833V. This means the full range of  $\pm 10V$  is not fully usable. To use the full range of  $\pm 10V$ , the gain factor has to be adjusted to 11.667 rps/V.

Drive manufacturers usually provide a potentiometer for adjusting this gain factor. Some manufacturers provide a few preset values selectable with jumpers or DIP switches.

**Step 7**

Set the **Drive Command Voltage** to 1V and record the velocity.

**Step 8**

Select the **Step** button and set the step voltage to the desired level, select the desired step duration, and enter the velocity that you recorded during step 7 in the **1V = \_\_\_ units/sec** data field.

**Step 9**

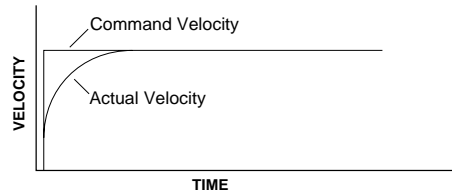
Select the **Sampling** button and set the sample period and the number of samples as desired.

**Step 10**

Select the **Start** button to initiate the step voltage and collect the data. The response will be displayed on the graph.

**Step 11**

Using the tuning parameters provided to you by your drive system, tune the drive until you achieve a *fast first-order response* to the step output (see drawing below). This is accomplished by iteratively issuing step voltage outputs to the drive, viewing the response, and modifying the available tuning parameters on the drive.

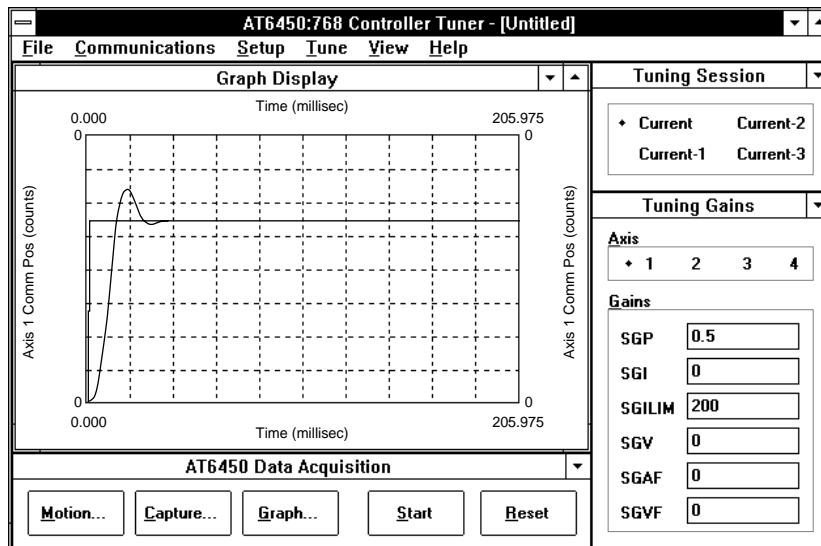


The  $K_a$  gain and the time constant can both be used in the Controller Tuner module for “automatic gain selection”.



# CHAPTER ②

## *Controller Tuner*



# Controller Tuner Basics

## Before You Begin

Before you begin the tuning process, make sure you have completed the system connection and test procedures provided in the *Installation* chapter of your 6000 Series controller's user guide.

The Controller Tuner is a graphical tuning aide that helps you tune the servo controller's position loop system. **If you have velocity drives, you should first tune the drives using the Drive Tuner module.**

The illustration below shows the main Controller Tuner window that is accessed from the **Utilities** pull-down menu.

Recommended tuning procedures and a sample tuning scenario are provided below.

**Connect**—engages the communication link with the controller.  
Bus-based controllers: verify the **Board Address** and **Send Operating System**.  
Serial controllers: select a **Serial Port**.

**System**—Select the # of axes you will use, and the servo sampling ratio.  
**Feedback**—Select the feedback device per axis and set up the operating conditions. Set the maximum allowable position error. Enable or disable the hardware end-of-travel limits.

**Automatic Gain Selection:** For each axis, select the PV or PIV tuning law, select the drive type, and enter the appropriate drive data. If you have a velocity drive, you can select the **Get Data** button to automatically fill in the drive scale factor and step response data from the last step output used in the last Drive Tuner session. The resulting gains, although possibly not the exact gains you need, will get you close to the final few tuning iterations.

From the **View** menu, you can open these displays:  
**Motion**—displays the commanded position and the actual position.  
**I/O**—displays the state of the inputs and outputs.  
**Status**—displays the axis status (TAS) or the system status (TSS).  
**Comm**—opens a terminal emulator window to communicate directly with the controller (use the **6000 Commands** reference under the **Help** menu if you need help with 6000 Series commands).  
**Cursor**—displays the values of the vertical and horizontal axes of the graph, from the point of the cursor.

Recall any of the last 4 tuning sessions. All the data used in the 4 respective sessions is retrieved and displayed.

Select the axis for the current tuning session.

**Iterative Tuning Process:**  
1. Enter the initial gain values (to save time, use the Automatic Gain Selection feature under the **Tune** menu).  
2. Select the **Start** button to test the response (check the graph).  
3. Adjust the gain values to achieve the desired response.  
4. Repeat steps 2 and 3 as necessary.

The **Motion** button displays a dialog box in which you can set up the motion to be executed while tuning. Options are: step input, trap, or S-curve profile, user program, or manual.  
*(See description below.)*

The **Capture** button displays a dialog box in which you can set up which data to capture, when to capture it, and the data sampling parameters and interval.  
*(See description below.)*

The **Graph** button displays a dialog box in which you can select which captured data to display in the graph. At any one time, 2 graphs can be displayed simultaneously.  
*(See description below.)*

The **Start** button initiates the step output and the data gathering function. After a short period, the data is displayed in the graph above.

The **Reset** button resets the controller to its default settings (e.g., the shutdown outputs are activated, the command output is held to zero volts, SGP is set to 0.5, SGLIM is set to 200, all other gain values are set to zero, etc.).

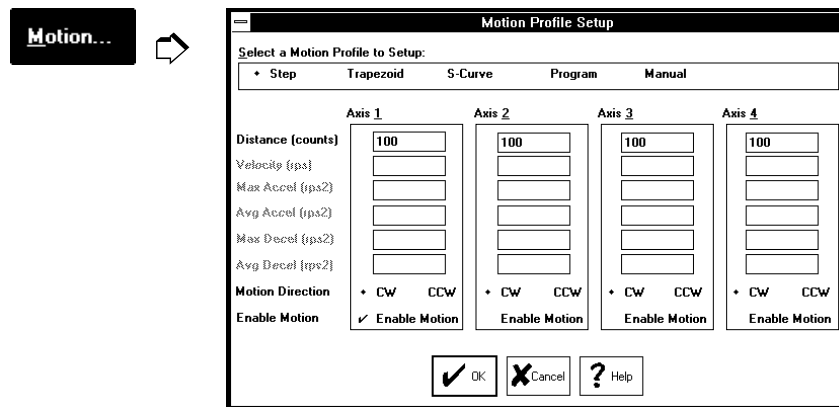
## Motion Profile Setup

Select the activity (motion profile, user program, or manual) you would like the controller to execute in the tuning process.

You could use a **Step** input profile for performance testing, or you can use a **Trapezoidal**, or an **S-Curve** profile. All the appropriate motion parameters are available for each profile type.

The **Program** option allows you to capture data relative to the execution of a predefined user program. Pressing the **Start** button begins execution of the program. The tuning data is captured when the trigger condition you specify in the Data Capture Setup dialog box is met.

The **Manual** option is provided so that you can trigger the data capture based on the trigger condition you specify in the Data Capture Setup dialog box. Pressing the **Start** button “arms” the wait-on-trigger condition and when the trigger condition is met, the tuning data is captured.



## Data Capture Setup

Set up these data capture parameters:

- The data to be captured.
- The trigger point used to initiate the data capture function.
- The data sampling parameters. You can manually enter these parameters (sample period and number of samples), or you can move the slide bar to select the portion of the move profile from which you want data to be selected. The total number of samples is divided by the number of items you are capturing (e.g., in the illustration below, there will be 750 commanded position samples and 750 actual position samples); therefore, selecting more items to be captured means there will be fewer samples per item.

**Capture...**



**Data Capture Setup**

<b>Data to be Captured</b>		<b>Data Sampling Parameters</b>		<input checked="" type="checkbox"/> OK <input type="checkbox"/> Cancel <input type="checkbox"/> Help
Capture from Axis: + 1 2 3 4 Commanded Acceleration Commanded Velocity <input checked="" type="checkbox"/> Commanded Position <input checked="" type="checkbox"/> Actual Position	Analog Servo Output Inputs Outputs Reserved	Sample Period (millisec) 0.275	Total Number of Samples (maximum 2500) 1500	
<b>Trigger Point that Initiates Data Capture</b>		<b>Data Sampling Interval (Inactive)</b>		
Trigger on Axis: + 1 2 3 4 Commanded Accel (rps2) 0 Commanded Vel (rps) 0 Comm Abs Pos (counts) 0 Actual Abs Pos (counts) 0 Analog Servo Out (volts) 0 Input Bit # 1 Level 1 Output Bit # 1 Level 1 + Go Command		Slide to Start-of-Sample Interval: 0.000 millisec <div style="border: 1px solid black; padding: 5px; text-align: center;">             A Profile appears here only if you select Acceleration, Velocity, or Position as the data capture trigger.           </div> Slide to End-of-Sample Interval: 0.000 millisec		

## Graph Setup

Use the Graph Setup dialog box to select what will be displayed in the graph. Any information selected in a session can be displayed on the graph (see *Data Capture Setup* above).

The vertical and horizontal axes are selected for each plot on the graph. At any one time, up to two plots can be displayed simultaneously, as in the example on page 10.

You can compare the data from one session to the data from a previous session. To access the previous sessions, use the **Display Session** portion of the dialog box.

**Graph...**



**Graph Setup**

Select a Graph to Setup: + Graph 1 Graph 2

<b>Vertical Axis</b> Actual Acceleration Command Acceleration Actual Velocity Commanded Velocity Velocity Error Actual Position <input checked="" type="checkbox"/> Commanded Position Position Error Analog Servo Output + 1 2 3 4	<b>Horizontal Axis</b> <input checked="" type="checkbox"/> Time Actual Position Commanded Position Position Error Axis: + 1 2 3 4	<b>Options</b> <input checked="" type="checkbox"/> Display Graph <b>Display Session</b> <input checked="" type="checkbox"/> Current Previous <b>Scale Graph Axes</b> Vertical(%) 80 Horizontal(%) 100
---	---	--

OK   
  Cancel   
  Help

# Controller Tuning Procedure

---

## Before you tune the Controller

If you are using a velocity drive, be sure to complete the *Drive Tuning Procedure* before proceeding with the tuning procedure below.

You must tune one axis at a time; therefore, you will have to repeat Steps 2 through 6 below for each additional axis.

**Switching Feedback Sources:** If your application requires switching between feedback sources on the same axis, then for each feedback source on each axis, you must select the feedback source (see Step 1.d.) and repeat Steps 2 through 6.

The *Controller Tuning Procedure* below leads you through these main steps:

1. Set up the Controller Tuner module (sampling ratio, feedback device, maximum position error, etc.).
2. Set up the Controller Tuner module's data gathering functions.
3. Optimize the Proportional (**SGP**) and Velocity (**SGV**) gains.  
An alternative would be to use the Automatic Gain Selection feature, which automatically calculates **SGP** and **SGV** gains (and **SGI**, if so desired) based on the last drive tuning session or motor/drive and load parameters.
4. Use the Integral Feedback Gain (**SGI**) to reduce steady state error.
5. Use the Velocity Feedforward Gain (**SGVF**) to reduce position error at constant velocity.
6. Use the Acceleration Feedforward Gain (**SGAF**) to reduce position error during acceleration and deceleration.

During the tuning process, you may want to take advantage of the display options available from the **View** pull-down menu:

- **Motion**.....Commanded position and actual position of all axes.
- **I/O**.....State of inputs and outputs.
- **Status**.....Axis Status and System Status information for all axes.
- **Comm**.....Opens a terminal emulator window to communicate directly with the controller.
- **Cursor**.....Values of the vertical and horizontal axes of the graph, from the position of the cursor.

**Step 1 – Set up Controller Tuner**

- a. Launch the Controller Tuner from the **Utilities** menu in Motion Architect.
- b. *This step assumes you have already selected your controller (see **Product** menu in Motion Architect).* Verify that there is a check mark next to **Connect** under the **Communications** menu; this indicates that the communication link to the controller is established.
- c. Select **System** from the **Setup** pull-down menu. In the dialog box, select the number of axes you will be using, and select the sampling frequency ratio (SSFR) value appropriate for your application. (The default setting, SSFR4, is sufficient for most applications.) Use the table below as a guide for selecting the appropriate SSFR setting.

# of Axes (INDAX)	SSFR Setting	Servo Sampling Update		Motion Trajectory Update		System Update	
		Frequency (samples/sec.)	Period (µsec)	Frequency (samples/sec.)	Period (µsec)	Frequency (samples/sec.)	Period (µsec)
I	1 1	3030	330	3030	330	757	1320
	1 2	4000	250	2000	500	500	2000
	1 4	4651	215	1163	860	581	1720
	1 8	4878	205	610	1640	610	1640
II	2 1	1667	600	1667	600	417	2400
	2 2	2272	440	1136	880	568	1760
	2 4	2500	400	625	1600	625	1600
	2 8	2667	375	333	3000	333	3000
	3 1	1282	780	1282	780	427	2340
	3 2	1515	660	758	1320	379	2640
	3 4	1667	600	417	2400	417	2400
	4 1	976	1025	976	1025	325	3075
	4 2	1143	875	571	1750	286	3500
IV	4 4	1274	785	318	3140	318	3140

- I Default setting for single axis controllers and packaged drive/controller systems.
- II Default setting for two-axis controllers and packaged drive/controller systems.
- IV Default setting for four-axis controllers and packaged drive/controller systems.

The general rule to determining the proper SSFR value is to first select the slowest servo sampling frequency that is able to give a satisfactory response. This can be done by experiment or based on the closed-loop bandwidth requirement for your application. (Keep in mind that increasing the SSFR value allows for higher bandwidths, but produces a rougher motion profile; conversely, decreasing the SSFR value provides a smoother profile, but makes the servo system less stable and slower to respond.)

As an example, if your application requires a closed-loop bandwidth of 300 Hz, and you determine the minimum servo sampling frequency by using the rule of thumb (setting the servo sampling frequency at least 8 times higher than the bandwidth frequency), the required minimum servo sampling frequency would be 2400 Hz. If two axes are running (INDAX2), then you should try using the SSFR4 setting.

For more in-depth discussion on the different update parameters (servo, motion and system), refer to the SSFR command description in the on-line **6000 Software Reference**.

- d. Select **Feedback** from the **Setup** pull-down menu. In the dialog box, select the type of feedback device for each axis and select the operating conditions appropriate to your application.

Select the appropriate resolution value (specific to the feedback source) from the table below.

	<b>Encoder</b>	<b>Resolver</b>	<b>ANI</b>	<b>LDT</b>
<b>Resolution Value</b>	4000 (counts/rev)	4096 (counts/rev)	819 (counts/volt)	432 x recirculations (counts/inch)

Set the maximum allowable position error. Larger values allow greater oscillations/motion when unstable; therefore, smaller values are safer. Entering a value of zero disables monitoring for position error. Axis status bit #23 will indicate if the maximum position error has been exceeded (see **Status** display under the **View** menu).

If you are using hardware end-of-travel limits, enable them here.

### **Step 2 – Set up data gathering functions**

- a. Click on the **Motion** button in the **Data Acquisition** display to show the **Motion Profile Setup** dialog box. Select the **Step** profile, set the **Distance** to 100 steps, select the direction of motion (CW = positive counting; CCW = negative counting), select **Enable Motion** for the axis you are tuning (make sure to deselect **Enable Motion** for the other axes), and click **✓OK**.
- b. Click on the **Capture** button in the **Data Acquisition** display to show the **Data Capture Setup** dialog box. Select the axis you are currently tuning, select **Commanded Position**, **Actual Position**, and **Analog Servo Output** (deselect all other options), select **Go Command** as the trigger point that initiates data capture, and click **✓OK**.
- c. Click on the **Graph** button in the **Data Acquisition** display to show the **Graph Setup** dialog box.

Select **Graph 1**. Under **Vertical Axis**, select **Commanded Position** and select the axis number you are currently tuning. Under **Horizontal Axis**, select **Time** and select the axis number you are currently tuning. Under **Options**, select **Display Graph**.

Select **Graph 2**. Under **Vertical Axis**, select **Actual Position** and select the axis number you are currently tuning. Under **Horizontal Axis**, select **Time** and select the axis number you are currently tuning. Under **Options**, select **Display Graph**.

Click **✓OK**.

**Step 3 – Optimize proportional (SGP) and velocity (SGV) gains**  
(refer to illustration below for suggested tuning process)

**Automatic Gain Selection Option (rotary drives only)**

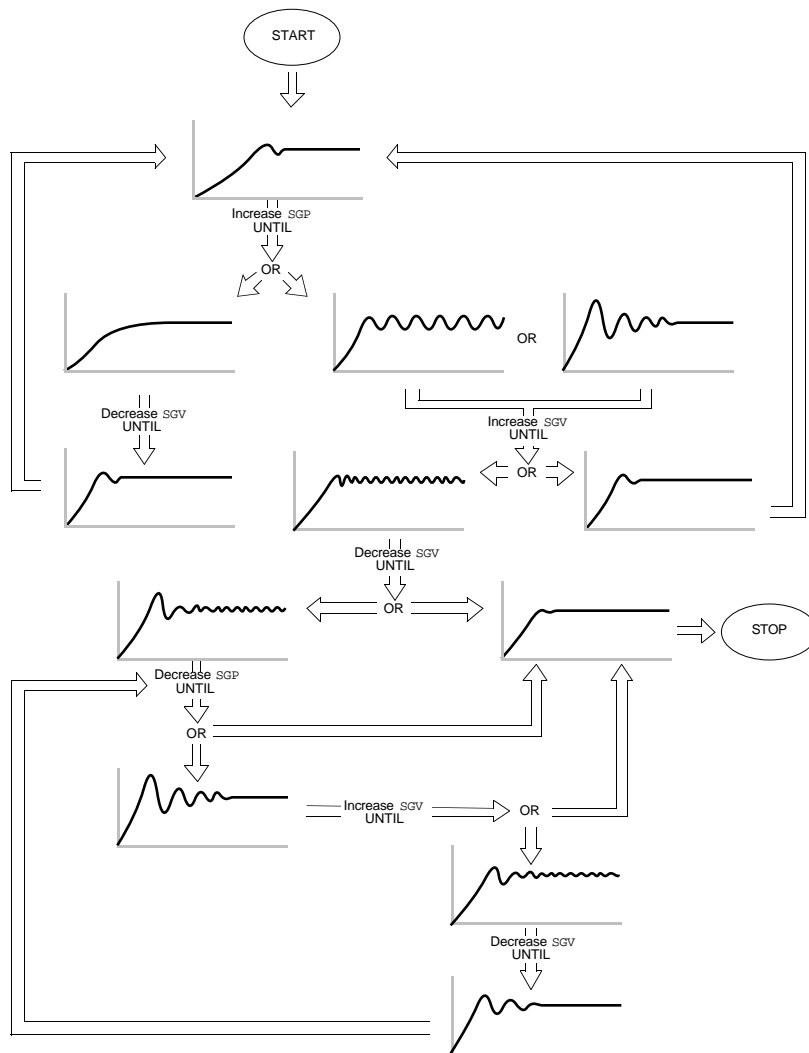
As an alternative to Steps 3.a. through 3.e. below, you may use the Automatic Gain Selection feature to automatically select **SGP** and **SGV** gains that should greatly shorten the iterative nature of the controller tuning process.

To use this feature, select **Automatic Gain Selection** from the **Tune** menu. For each axis, provide the following information:

1. Under **Control Law**, select **PV**.
2. Under **Drive Type**, select either **Velocity** or **Torque**.
3. Enter the drive data:
  - Velocity drive: fill in the **Drive Scale Factor** and **Step Response** data. If you select the **Get Data** button, these data fields will be filled in with the data captured from the last Drive Tuner session.
  - Torque drive: fill in the **Total Inertia** and **Motor/Drive Constants** data.
4. Click the **✓OK** button. Notice that the data fields in the **Tuning Gains** display show the new gain settings.

- a. In the **Data Acquisition** display, select the **Start** button to trigger the step input move and gather data.
- b. Observe the plot of the commanded position versus the actual position in the **Graph Display** area. If the response is already very oscillatory, lower the gain (**SGP**); if it is *sluggish* (overdamped), increase the **SGP** gain. *Repeat Steps 3.a. and 3.b. until the response is slightly under-damped.*
- c. In the **Tuning Gains** panel, set the initial **SGV** value to **0.1**.
- d. As you did in Step 3.a., select the **Start** button.
- e. Observe the plot in the **Graph Display** area. If the response is *sluggish* (overdamped), reduce the **SGV** gain. *Repeat Steps 3.d. and 3.e. until the response is slightly under-damped.*
- f. The flow diagram below shows you how to get the values of the proportional and velocity feedback gains for the fastest, well-damped response in a step-by-step fashion. (Refer to the *Tuning Scenario* section later in this chapter for a case example.) The tuning principle here is based on these four characteristics:
  - Increasing the proportional gain (**SGP**) can speed up the response time and increase the damping.
  - Increasing the velocity feedback gain (**SGV**) can increase the damping more so than the proportional gain can, but also may slow down the response time.

- When the **SGP** gain is too high, it can cause instability.
- When the **SGV** gain is too high, it can cause the motor (or hydraulic cylinder, etc.) to chatter.



**Step 4 – Use the integral feedback gain (SGI) to reduce steady state position error**

- a. Determine the steady state position error (the difference between the commanded position and the actual position). You can ascertain this error value by using the **Graph** feature, or by viewing the **Motion Display** (selected from the **View** pull-down menu).

**NOTE**

If the steady state position error is zero or so small that it is acceptable for your application, **you do not need to use the integral gain**. The use of the Target Zone Settling Mode is recommended, however (details are provided in Appendix B).

- b. If you have to enter the **SGI** gain to reduce the steady error, start out with a small value (e.g., **0.1**). After the gain is entered, observe two things from the response:
  - If the magnitude of steady state error reduces.
  - If the steady state error reduces to zero at a faster rate.
- c. Keep increasing the **SGI** gain to further improve these two measurements until the overshoot starts to increase and the response becomes oscillatory.
- d. There are three things you can do at this point (If these three things do not work, that means the integral gain is too high and you have to lower it.):
  - 1<sup>st</sup> Lower the **SGI** gain value to reduce the overshoot.
  - 2<sup>nd</sup> Check whether the controller's analog output saturates the  $\pm 10V$  limit; you can do this by using the data gathering feature (in the **Graph Setup** dialog box, select **Analog Servo Output** versus **Time** for **Graph 1** and cancel the display for **Graph 2**). If it saturates, then lower the integral output limit by using the **SGILIM** parameter. This should help reduce the overshoot and shorten the settling time. Sometimes, even if the analog output is not saturated, you can still reduce the overshoot by lowering **SGILIM** to a value less than the maximum output value. *However, lowering it too much can impair the effectiveness of the integral feedback.*
  - 3<sup>rd</sup> You can still increase the velocity feedback gain (**SGV** value) further, provided that it is not already at the highest possible setting (causing the motor to chatter).

**Step 5 – Use the velocity feedforward gain (SGVF) to reduce position error at constant speed**

- a. In the **Graph Setup** dialog box (via the **Graph** button), set Graph 1 to display commanded position versus time, and set Graph 2 to display actual position versus time.

**Alternative setup if velocity error is critical:**

Set up the **Graph** feature to compare commanded velocity versus time and actual velocity versus time, and set up the **Capture** feature to include commanded velocity.

- b. In the **Motion Profile Setup** dialog box (via the **Motion** button), select a trapezoidal or s-curve profile and set the acceleration, deceleration and velocity to the values appropriate to your application.
- c. In the **Tuning Gains** panel, set the initial **SGVF** value equal to the product of **SGP \* SGV**. For example, if **SGP = 1.2** and **SGV = 0.5**, then **SGVF = 1.2 \* 0.5 = 0.6**. If **SGV = zero**, then just set **SGVF = SGP**.
- d. In the **Data Acquisition** display, select the **Start** button to trigger the move and gather data.
- e. Note the plot in the **Graph Display**; the actual position (or velocity) probably lags the commanded position (or velocity).

The objective is to increase **SGVF** until the lag is reduced to a level suitable for your application.

**Step 6 – Use the acceleration feedforward gain (SGAF) to reduce position error during acceleration**

- a. In the **Tuning Gains** panel, set the initial **SGAF** value to **0.01**.
- b. In the **Data Acquisition** display, select the **Start** button.
- c. Note the plot in the **Graph Display**; the actual position (or velocity) probably lags the commanded position (or velocity).

The objective is to increase **SGAF** until the lag is reduced to a level suitable for your application.

## Tuning Scenario

This example shows how to obtain the highest possible proportional feedback (**SGP**) and velocity feedback (**SGV**) gains experimentally by using the flow diagram illustrated earlier in Step 3 of the *Controller Tuning Procedure*.

### NOTE

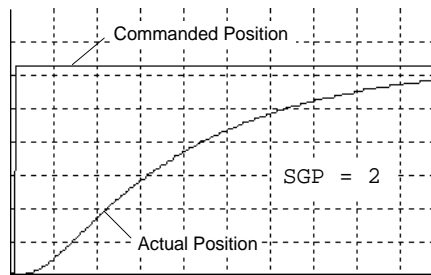
The steps shown below (steps 1 - 11) represent the major steps of the process; the actual progression between these steps may require several iterations.

The motion command used for this example is a step command with a step size of 100. The plots shown are as they appear in the Controller Tuner Module (X axis = time, Y axis = position).

#### Step 1

For a starting trial, we set the proportional feedback gain (**SGP**) to 2. As you can see by the plot, the response is slow.

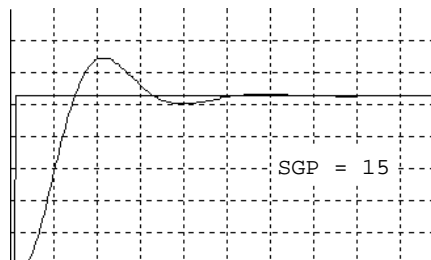
In the next step, we should increase **SGP** until the response is slightly underdamped.



#### Step 2

With **SGP** equal to 15, the response becomes slightly underdamped (see plot).

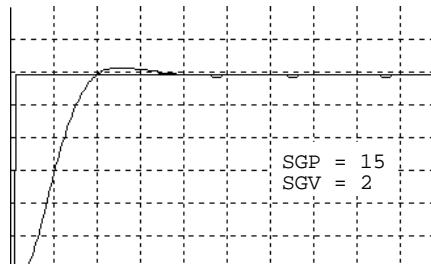
Therefore, we should introduce the velocity feedback gain (**SGV**) to *damp out* the oscillation.



#### Step 3

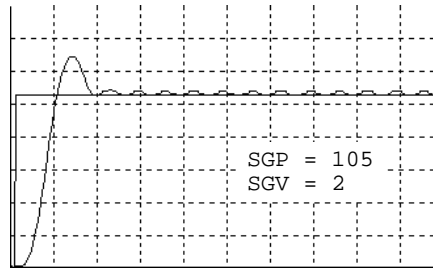
With **SGV** equal to 2, the response turns out fairly well damped (see plot).

At this point, the **SGP** should be raised again until oscillation or excessive overshoot appears.



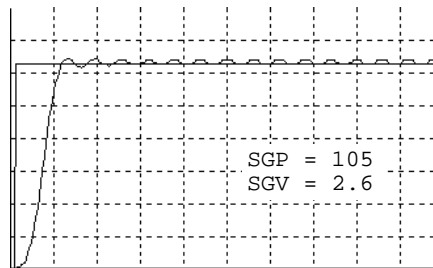
**Step 4**

As we iteratively increase **SGP** to 105, overshoot and chattering becomes significant (see plot). This means the **SGV** gain is too low and/or the **SGP** is too high. Next, we should try raising the **SGV** gain to see if it could damp out the overshoot and chattering.



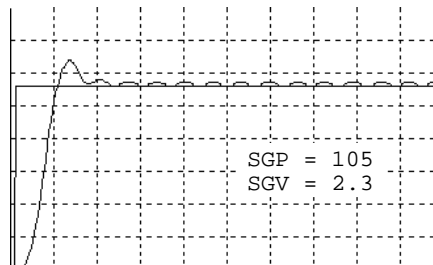
**Step 5**

After the **SGV** gain is raised to 2.6, the overshoot was reduced but chattering is still quite pronounced. This means either one or both of the gains is too high. The next step should be to lower the **SGV** gain first.



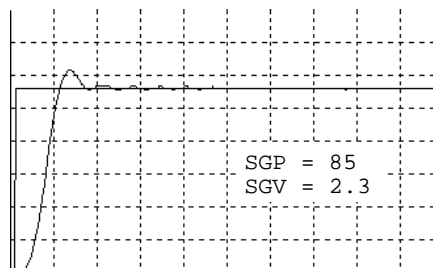
**Step 6**

Lowering the **SGV** gain to 2.3 does not help reduce the chattering by much. Therefore, we should lower the **SGP** gain until chattering stops.



**Step 7**

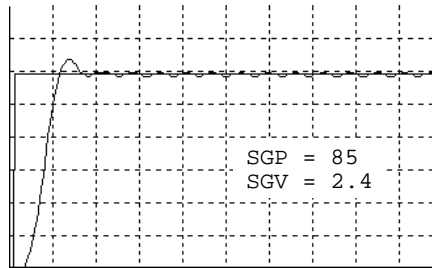
Chattering stops after reducing the **SGP** gain to 85. However, the overshoot is still a little too high. The next step should be to try raising the **SGV** to damp out the overshoot.



**Step 8**

After raising the **SGV** gain to 2.4, overshoot is reduced a little, but chattering reappears. This means the gains are still too high.

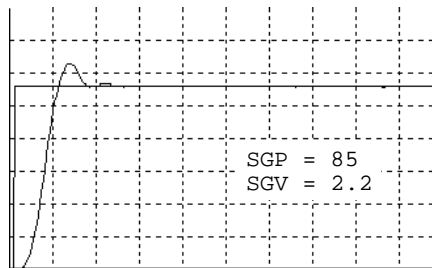
Next, we should lower the **SGV** gain until chattering stops.



**Step 9**

After lowering the **SGV** gain to 2.2 (even less than in Step 7—2.3), chattering stops.

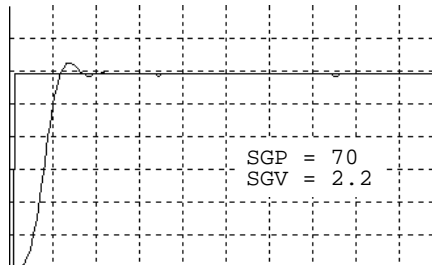
Next we should lower the **SGP** gain.



**Step 10**

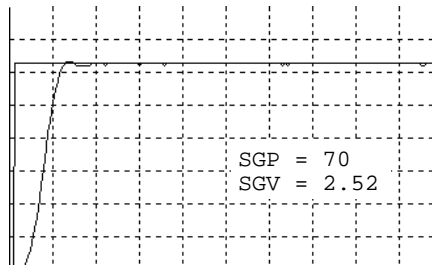
Overshoot is reduced very little after lowering the **SGP** gain to 70. (The **SGV** gain might have been lowered too much in Step 9.)

Next, we should try raising the **SGV** gain again until the overshoot is gone.



**Step 11**

When we raised the **SGV** gain to 2.52, the step response became fast and very stable.



## Appendix A

# Servo Tuning Principles

---

---

## Servo System Terminology

---

This section gives you an overall understanding of the principles and the terminology used in tuning your 6000 Series Servo Controller.

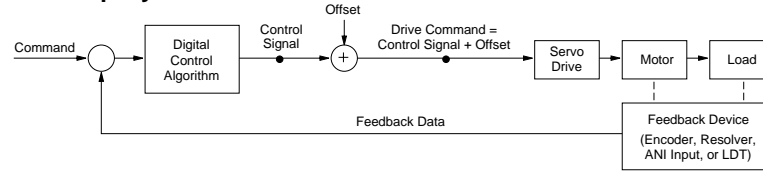
### Servo Tuning Terminology

The controller uses a digital control algorithm to control and maintain the position and velocity. The digital control algorithm consists of a set of numerical equations used to periodically (once every **servo sampling period**) calculate the value of the **control signal** output. The numerical terms of the equations consist of the current commanded and actual position values (plus a few from the past sampling period) and a set of control parameters. Each control parameter, commonly called a **gain**, has a specific function (see *Servo Control Techniques* later in this appendix). **Tuning** is the process of selecting and adjusting these gains to achieve optimal servo performance.

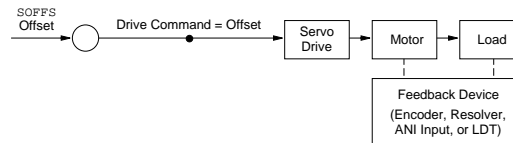
When this control algorithm is used, the whole servo system is a **closed-loop** system (see diagram below). It is called closed loop because the control algorithm accounts for both the **command** (position, velocity, tension, etc.) and the **feedback data** (from the encoder, resolver, ANI input, or LDT); therefore, it forms a *closed loop* of information flow.

When all gains are set to zero, the digital control algorithm is disabled. During system setup or troubleshooting, it is desirable to disable the algorithm (by setting all the gains to zero) and use the `SOFFS` command to directly control the analog output.

## Closed Loop System



## Servo Algorithm Disabled



The controller has the capability of providing an analog voltage output of  $\pm 10V$  for commanding the drive (hydraulic controllers can be configured for current output). After the digital control algorithm has calculated the digital control signal, this digital value is sent out from the DSP (digital signal processor) to the Digital-to-Analog converter (DAC). The DAC has an analog output range of  $-10V$  to  $+10V$ . It is often possible that the digital control signal calculated by the control algorithm can exceed this limit. When this happens, the analog output would just stay, or *saturate*, at the maximum limit until the position error changes such that the control algorithm would calculate a control signal less than the limit. This phenomenon of reaching the output limit is called **controller output saturation**. When saturation occurs, increasing the gains does not help improve performance since the DAC is already operating at its maximum level.

## Position Variable Terminology

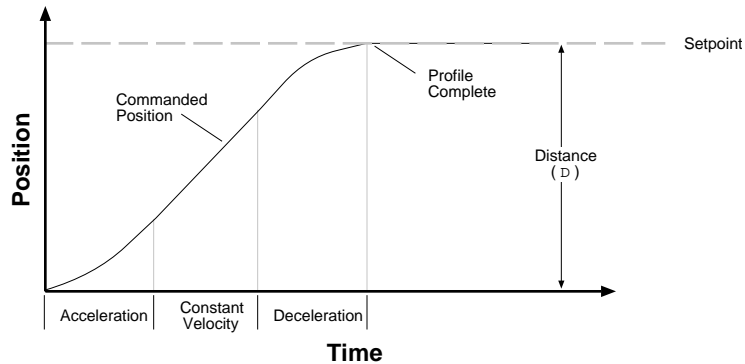
In a servo system, there are two types of **time-varying** (value changes with time) position information used by the controller for control purposes: commanded position and actual position. You can use this information to determine if the system is positioning as you expect.

### Commanded Position

The **commanded position** is calculated by the motion profile routine based on the acceleration (A, AA), deceleration (AD, ADA), velocity (V) and distance (D) command values and it is updated every servo sampling period. Therefore, the commanded position is the intended position at any given point of time. To view the commanded position, use the TPC (Transfer Commanded Position) command; the response represents the commanded position at the instant the command is received.

When this user guide refers to the *commanded position*, it means this calculated time-varying commanded position, not the distance (D) command. Conversely,

when this user guide refers to the **position setpoint**, it means the final intended distance specified with the distance (D) command. The following plot is a typical profile of the commanded position in preset (MCØ) mode.



### Actual Position

The other type of time-varying position information is the **actual position**; that is, the actual position of the motor (or load) measured with the feedback device (encoder, resolver, ANI input, or LDT). Since this is the position achieved when the drive responds to the commanded position, we call the overall picture of the actual position over time the **position response** (see further discussion under *Servo Response Terminology*).

To view the actual position, use the TFB (Transfer Position of Feedback Device) command; the response represents the actual position at the instant the command is received. The goal of tuning the servo system is to get the actual position to track the commanded position as closely as possible.

The difference between the commanded position and actual position is the **position error**. To view the position error, use the TPER (Transfer Position Error) command; the response represents the position error at the instant the command is received. When the motor is not moving, the position error at that time is called the **steady-state position error** (see definition of steady-state under *Servo Response Terminology*). If a position error occurs when the motor is moving, it is called the **position tracking error**.

In some cases, even when the system is properly tuned, the position error can still be quite significant due to a combination of factors such as the desired profile, the motor's limitation, the dynamic characteristics of the system, etc. For example, if the value of the velocity (V) command is higher than the maximum velocity the motor can physically achieve, then when it is commanded to travel at this velocity, the actual position will always lag behind the commanded position and a position error will accumulate, no matter how high the gains are.

## Servo Response Terminology

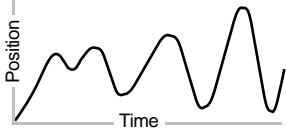
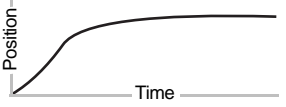
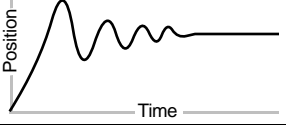
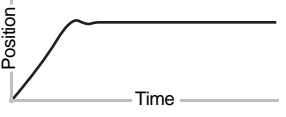
### Stability

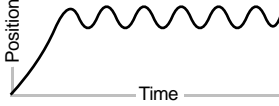
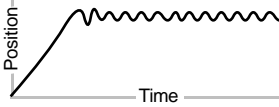
The first objective of tuning is to stabilize the system. The formal definition of **system stability** is that when a bounded input is introduced to the system, the output of the system is also bounded. What this means to a motion control system is that if the system is **stable**, then when the position setpoint is a finite value, the final actual position of the system is also a finite value.

On the other hand, if the system is **unstable**, then no matter how small the position setpoint or how little a disturbance (motor torque variation, load change, noise from the feedback device, etc.) the system receives, the position error will increase continuously, and exponentially in almost all cases. In practice, when the system experiences instability, the actual position will oscillate in an exponentially diverging fashion as shown in the drawing below. The definition here might contradict what some might perceive. One common perception shared by many is that whenever there is oscillation, the system is unstable. However, if the oscillation finally diminishes (damps out), even if it takes a long time, the system is still considered stable. The reason for this clarification is to avoid misinterpretation of what this user guide describes in the following sections.

### Position Response Types

The following table lists, describes, and illustrates the six basic types of position responses. The primary difference among these responses is due to **damping**, which is the suppression (or cancellation) of oscillation.

Response	Description	Profile (position/time)
Unstable	Instability causes the position to oscillate in an exponentially diverging fashion.	
Over-damped	A highly damped, or <i>over-damped</i> , system gives a smooth but slower response.	
Under-damped	A slightly damped, or <i>under-damped</i> , system gives a slightly oscillatory response.	
Critically damped	A critically-damped response is the most desirable because it optimizes the trade-off between damping and speed of response.	

Oscillatory	An oscillatory response is characterized by a sustained position oscillation of equal amplitude.	
Chattering	Chattering is a high-frequency, low-amplitude oscillation which is usually audible.	

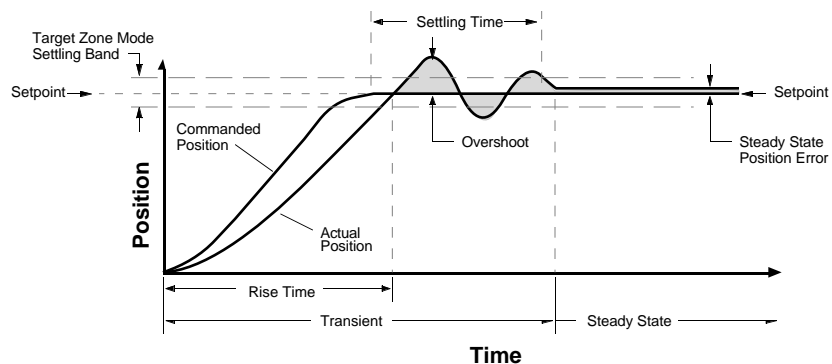
### Performance Measurements

When we investigate the plot of the position response versus time, there are a few measurements that you can make to quantitatively assess the performance of the servo:

- **Overshoot**—the measurement of the maximum magnitude that the actual position exceeds the position setpoint. It is usually measured in terms of the percentage of the setpoint value.
- **Rise Time**—the time it takes the actual position to pass the setpoint.
- **Settling Time**—the time between when the commanded position reaches the setpoint and the actual position settles within a certain percentage of the position setpoint. (Note the settling time definition here is different from that of a control engineering text book, but the goal of the performance measurement is still intact.)

These three measurements are made before or shortly after the motor stops moving. When it is moving to reach and settle to the setpoint, we call such period of time the **transient**. When it is not moving, it is defined as in **steady-state**.

A typical stable position response plot in preset mode (MCØ) is shown below.



## Tuning-Related Software Commands

More detailed information on each 6000 Series command can be found in the on-line **6000 Software Reference** (access from the **Help** pull-down menu).

Command	Description
LDTUPD	<b>Hydraulic controllers only.</b> Use LDTUPD to select the rate at which the LDT position is sampled. Decreasing the LDTUPD command value increases the update rate and improves the quality of the dynamic response. However, if the update rate is too fast, the LDT will not have enough time to read the position, resulting in position read errors. The occurrence of read errors can be monitored with the ER bit #15 if you enable ERROR bit 15, or you can monitor the LDT status with AS bit #27.
SFB	Selects the servo feedback transducer. Options are encoder, resolver, ANI input, or LDT.
SGAF	Sets the acceleration feedforward gain in the PIV&F <sub>a</sub> servo algorithm.
SGENB	Enables a previously-saved set of PIV&F gains. A set of gains (specific to the current feedback source selected with the SFB command) is saved using the SGSET command.
SGI	Sets the integral gain in the PIV&F servo algorithm.
SGILIM	Sets a limit on the correctional control signal that results from the integral gain action trying to compensate for a position error that persists too long.
SGP	Sets the proportional gain in the PIV&F servo algorithm.
SGSET	Saves the presently-defined set of PIV&F gains as a particular <i>gain set</i> (specific to the current feedback source on each axis). Up to 5 gain sets can be saved and enabled at any point in a move profile, allowing different gains at different points in the profile.
SGV	Sets the velocity gain in the PIV&F servo algorithm.
SGVF	Sets the velocity feedforward gain in the PIV&F <sub>v</sub> servo algorithm.
SMPER	Sets the maximum allowable error between the commanded position and the actual position as indicated by the feedback device. If the error exceeds this limit, the controller activates the Shutdown output and sets the DAC output to zero (plus any SOFFS offset). If there is no offset, the motor will freewheel to a stop. You can enable the ERROR command to continually check for this error condition, and when it occurs to branch to a programmed response defined in the ERRORP program.
SOFFS	Sets an offset to the commanded analog output voltage, which is sent to the drive system.
SSFR	Sets the ratio between the update rate of the move trajectory and the update rate of the servo action. The intermediate position setpoints calculated by the trajectory generator is updated at a slower rate than the servo position correction. SSFR allows you to optimize this for your application. The default setting (SSFR4) is sufficient for most applications.

STRGTD STRGTE STRGTT STRGTV	<p>When using the Target Zone Mode, enabled with the <code>STRGTE</code> command, the actual position and actual velocity must be within the <i>target zone</i> (that is, within the distance zone defined by <code>STRGTD</code> and within the velocity zone defined by <code>STRGTV</code>). If the motor/load does not settle into the target zone before the timeout period set by <code>STRGTT</code>, the controller detects an error.</p> <p>To prevent subsequent commands/moves from being executed when this error condition occurs, you must enable the <code>ERROR</code> command to continually check for this error condition, and when it occurs to branch to a programmed response defined in the <code>ERRORP</code> program. Otherwise, subsequent commands/moves can be executed regardless of the actual position and velocity.</p> <p>Target Zone Mode is explained in greater detail in Appendix B.</p>
TFB and [ FB ]	Transfers [or assigns/compares] the actual position of the transducers selected for feedback (see <code>SFB</code> ).
TDAC and [ DAC ]	Transfers [or assigns/compares] the output from the controller's digital-to-analog converter. This is the analog control signal output at the controller's <code>CMD</code> terminal.
TGAIN	Transfers the currently active set of PIV&F gains. The servo gain set reported represents the last gain values specified with the individual servo gain commands ( <code>SGI</code> , <code>SGP</code> , <code>SGV</code> , <code>SGAF</code> , and <code>SGVF</code> ), or the last gain set enabled with the <code>SGSET</code> command.
TPC and [ PC ]	Transfers [or assigns/compares] the commanded position (intermediate position setpoint) to the drive.
TPER and [ PER ]	Transfers [or assigns/compares] the error between the commanded position ( <code>TPC</code> ) and the actual position ( <code>TFB</code> , <code>TPE</code> , or <code>TANI</code> ) as measured by the feedback device.
TSGSET	Transfers a previously-saved set of servo gain parameters. A gain set is saved with the <code>SGSET</code> command.
TSTLT	Transfers the time it took the last move to settle within the <i>target zone</i> (that is, within the distance zone defined by <code>STRGTD</code> and within the velocity zone defined by <code>STRGTV</code> ). The Target Zone Mode does not need to be enabled to use this command.

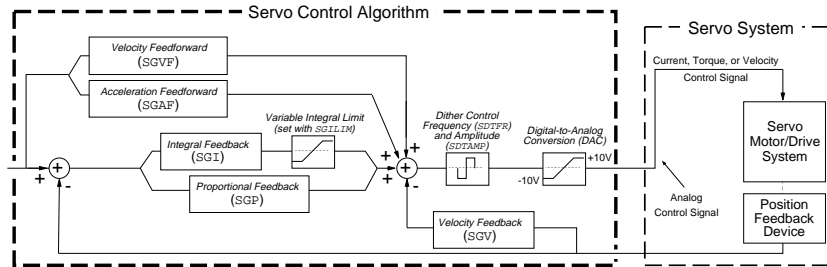
# Servo Control Techniques

To ensure that you are tuning your servo system properly, you should understand the tuning techniques described in this section.

The 6000 Series controller employs a *PIV&F* servo control algorithm. The control techniques available in this system are as follows:

- P* .... Proportional Feedback (controlled with the SGP command)
- I* .... Integral Feedback (controlled with the SGI command)
- V* .... Velocity Feedback (controlled with the SGV command)
- F* .... Velocity and Acceleration Feedforward (controlled by the SGVF and SGAF commands, respectively)

The block diagram below shows these control techniques in relation to the servo control algorithm configuration. The following table presents a condensed summary of each control's effect on the servo system.



Gain	Stability	Damping	Disturbance Rejection	Steady State Error	Tracking Error
Proportional (SGP)	Improve	Improve	Improve	Improve	Improve
Integral (SGI)	Degrade	Degrade	Improve	Improve	Improve
Velocity Feedback (SGV)	Improve	Improve	-----	-----	Degrade
Velocity Feedforward (SGVF)	-----	-----	-----	-----	Improve
Acceleration Feedforward (SGAF)	-----	-----	-----	-----	Improve

## Proportional Feedback Control (SGP)

**Proportional feedback is the most important feedback for stabilizing a servo system.** When the controller uses *proportional feedback*, the control signal is linearly proportional to the position error (the difference between the commanded position and the actual position—see TPER command). The proportional gain is set by the Servo Gain Proportional (SGP) command. Proportional feedback can be used to make the servo system more responsive, as well as reduce the steady state position error.

Since the control is proportional to the position error, whenever there is any disturbance (such as torque ripple or a spring load) forcing the load away from its commanded position, the proportional control can immediately output a signal to move it back toward the commanded position. This function is called *disturbance rejection*.

If you tune your system using only the proportional feedback, increasing the proportional feedback gain (SGP value) too much will cause the system response to be oscillatory, underdamped, or in some cases unstable.

### NOTE

The proportional feedback gain (SGP) should never be set to zero, except when open-loop operation is desired.

## Integral Feedback Control (SGI)

Using *integral feedback control*, the value of the control signal is integrated at a rate proportional to the feedback device position error. The rate of integration is set by the Servo Gain Integral (SGI) command.

The primary function of the integral control is to overcome friction and/or gravity and to reject disturbances so that steady state position error can be minimized or eliminated. This control action is important for achieving high system accuracy. *However, if you can achieve acceptable position accuracy by using only the proportional feedback (SGP), then there is no need to use the integral feedback control.*

**Hydraulic Controllers:** It is usually best to set the SGI gain to zero during the move. Then, after motion has stopped, use the proper SGI gain to hold position.

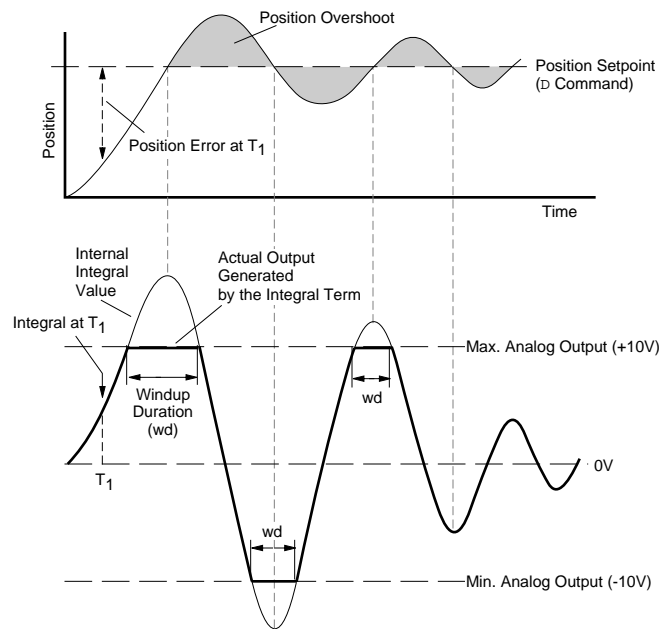
In the task of reducing position error, the integral gain (SGI) works differently than the proportional gain (SGP); this is because the magnitude of its control signal is not dependent on the magnitude of the position error as in the case of proportional feedback. If any position error persists, then the output of the integral term will ramp up over time until it is high enough to drive the error back to zero. Therefore, even a very small position error can be eliminated by the integral feedback control. By the same principle, integral feedback control can also reduce the tracking error when the system is commanded to cruise at constant velocity.

## Controlling Integral Windup

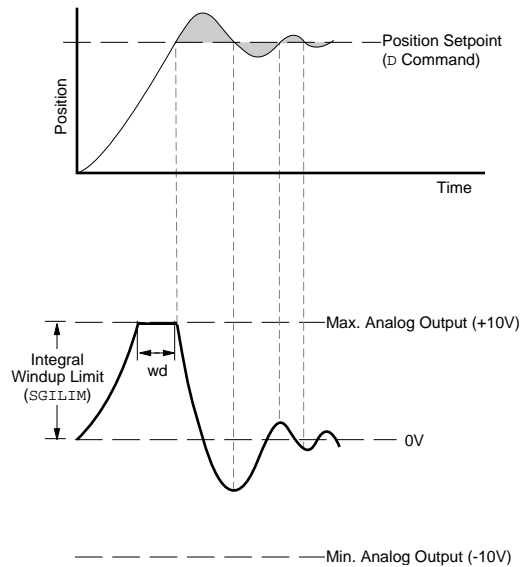
If integral control (SGI) is used and an appreciable position error has persisted long enough during the transient period (time taken to reach the setpoint), the control signal generated by the integral action can end up too high and saturate to the maximum level of the controller's analog control signal output. This phenomenon is called *integrator windup*.

After windup occurs, it will take a while before the integrator output returns to a level within the limit of the controller's output. Such a delay causes excessive position overshoot and oscillation. Therefore, the integral windup limit (SGILIM) command is provided for you to set the absolute limit of the integral and, in essence, turn off the integral action as soon as it reaches the limit; thus, position overshoot and oscillation can be reduced (see illustration below). The application of this feature is demonstrated in Step 5 of the *Controller Tuning Procedure* below.

### Without SGILIM



### With SGILIM



## Velocity Feedback Control (SGV)

When *velocity feedback control* is used, the control signal is proportional to the feedback device's velocity (rate of change of the actual position). The Servo Gain Velocity (SGV) command sets the gain, which is in turn multiplied by the feedback device's velocity to produce the control signal. Since the velocity feedback acts upon the feedback device's velocity, its control action essentially anticipates the position error and corrects it before it becomes too large. Such control tends to increase damping and improve the stability of the system.

A high velocity feedback gain (SGV) can also increase the position tracking error when traveling at constant velocity. In addition, setting the velocity feedback gain too high tends to slow down (*overdamp*) the response to a commanded position change. If a high velocity feedback gain is needed for adequate damping, you can balance the tracking error by applying velocity feedforward control (increasing the SGVF value—discussed below).

Since the feedback device's velocity is derived by differentiating the feedback device's position with a finite resolution, the finite word truncation effect and any fluctuation of the feedback device's position would be highly magnified in the velocity value, and even more so when multiplied by a high velocity feedback gain. When the value of the velocity feedback gain has reached such a limit, the motor (or hydraulic cylinder, etc.) will *chatter* (high-frequency, low-amplitude oscillation) at steady state.

## Velocity Feedforward Control (SGVF)

The purpose of velocity feedforward control is to improve *tracking performance*; that is, reduce the position error when the system is commanded to move at constant velocity. The tracking error is mainly attributed to three sources—friction, torque load, and velocity feedback control (SGV).

*Velocity feedforward control* is directed by the Servo Gain Velocity Feedforward (SGVF) setting, which is in turn multiplied by the rate of change (velocity) of the commanded position to produce the control signal. Consequently, because the control signal is now proportional to the velocity of the commanded position, the controller essentially anticipates the commanded position and initiates a control signal ahead of time to more closely follow (*track*) the commanded position.

Applications requiring linear interpolation can benefit from improved tracking performance; however, *if your application only requires short, point-to-point moves, velocity feedforward control is not necessary.*

Because velocity feedforward control is not in the servo feedback loop (see *Servo Control Algorithm* drawing above), it does not affect the servo system's stability. Therefore, there is no limit on how high the velocity feedforward gain (SGVF) can be set, except when it *saturates the control output* (tries to exceed the controller's analog control signal range of  $\pm 10V$ ).

## Acceleration Feedforward Control (SGAF)

The purpose of acceleration feedforward control is to improve position tracking performance when the system is commanded to accelerate or decelerate.

*Acceleration feedforward control* is directed by the Servo Gain Acceleration Feedforward (SGAF) setting, which is in turn multiplied by the acceleration of the commanded position to produce the control signal. Consequently, because the control signal is now proportional to the acceleration of the commanded position, the controller essentially anticipates the velocity of the commanded position and initiates a control signal ahead of time to more closely follow (*track*) the commanded position.

Same as velocity feedforward control, this control action can improve the performance of linear interpolation applications. In addition, it also reduces the time required to reach the commanded velocity. *However, if your application only requires short, point-to-point moves, acceleration feedforward control is not necessary.*

Acceleration feedforward control does not affect the servo system's stability, nor does it have any effect at constant velocity or at steady state.

## Appendix B

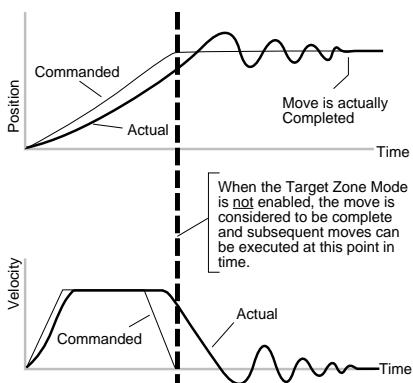
# Target Zone Mode

(Determining Move Completion Criteria)

### NOTE

The Target Zone Mode set up process is not part of the Servo Tuner functions. Setup commands may be issued to the controller via the **Comm** window (see **View** pull-down menu in the Controller Tuner module) or via the Terminal module in Motion Architect.

Under default operation (Target Zone Mode not enabled), the controller's move completion criteria is simply derived from the move trajectory. The controller considers the current preset move to be complete when the commanded trajectory has reached the desired target position; after that, subsequent commands/moves can be executed for that same axis. Consequently, the next move or external operation can begin **before** the actual position has settled to the commanded position (see diagram).



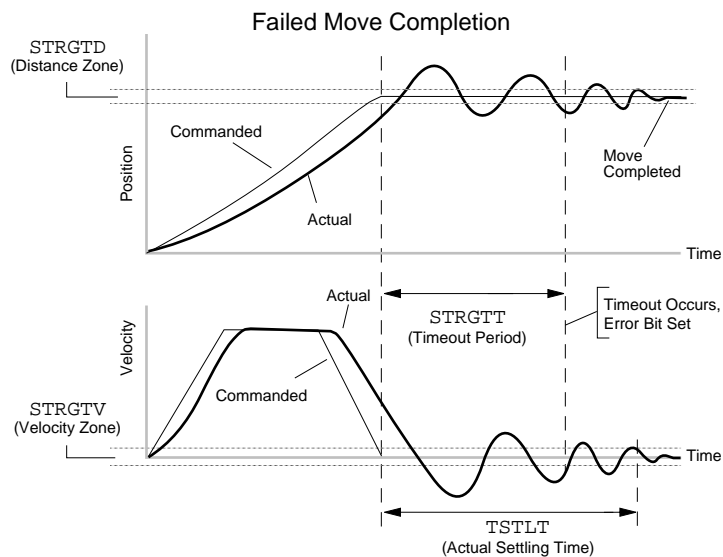
To prevent premature command execution before the actual position settles into the commanded position, use the *Target Zone Mode*. In this mode, enabled with the STRGTE1 command, the move cannot be considered complete until the actual position and actual velocity are within the *target zone* (that is, within the distance zone defined by STRGTD and less than or equal to the velocity defined by STRGTV). If the load does not settle into the

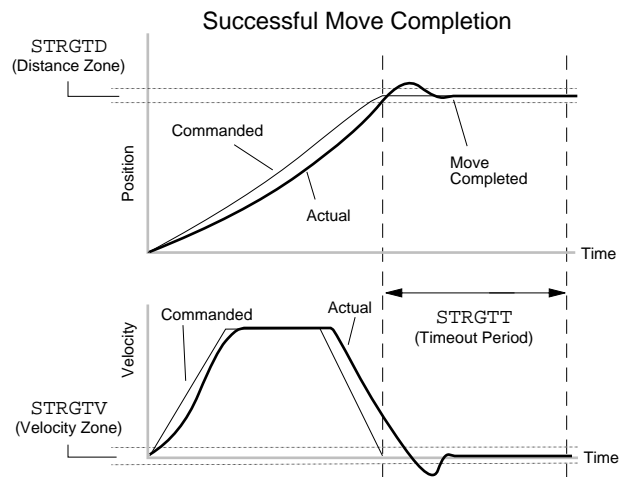
target zone before the timeout period set with the STRGTT command, the controller detects a *timeout error* (see illustration below).

If the timeout error occurs, you can prevent subsequent command/move execution only if you enable the ERROR command to continually check for this error condition (i.e., ERROR . 11-1), and when it occurs to branch to a programmed response you can define in the ERRORP program. Refer to the *Error Handling* portion of the on-line **6000 Software Reference** for error programming help.

As an example, setting the distance zone to  $\pm 5$  counts (STRGTD5), the velocity zone to  $\leq 0.5$  units/sec (STRGTV0.5), and the timeout period to 500 milliseconds (STRGTT500), a move with a distance of 8,000 counts (D8000) must end up between position 7,995 and 8,005 and settle down to  $\leq 0.5$  units/sec within 500 ms (1/2 second) after the commanded profile is complete.

**Damping is critical:** To ensure that a move settles within the distance zone, it must be damped to the point that it will not move out of the zone in an oscillatory manner. This helps ensure that the actual velocity falls within the target velocity zone set with the STRGTV command (see illustrations below).





### Checking the Actual Settling Time

Using the TSTLT command, you can display the actual time it took the last move to settle into the target zone (that is, within the distance zone defined by STRGTD and less than or equal to the velocity defined by STRGTV). The reported value represents milliseconds. **This command is usable whether or not the Target Zone Settling Mode is enabled with the STRGTE command.**



# Index

---

- 6000 code, saving 5
- acceleration feedforward control (SGAF) 32, 38
- acceleration ramp position error, reducing 23
- actual position 29
- address of AT-based controller 8, 14
- algorithm, servo control 34
- ANI input
  - connections, verify polarity 10
  - feedback source 14, 27, 32
  - position 29
  - resolution 19
  - scaling 8
- automatic gain selection 14, 20
- axes, number of 14
- board address 8, 14
- capture data
  - Controller Tuner 15
  - Drive Tuner 8
- chattering servo response 31
- closed-loop operation 27
- COM port, selecting 8, 14
- command polarity test 10
- command, servo output 27
- commanded position 28
- communication link to controller 8, 14
  - Comm display 14
- constant velocity position error, reducing 23
- control signal 27
- controller
  - controller output saturation 28
  - Controller Tuner 14
  - controller tuning procedure 17
- copy protection 4
- critically damped servo response 30
- cursor display 8, 14
- damping 30
- data capture
  - Controller Tuner 15
  - Drive Tuner 8
  - motion during 15
- default update rates 18
- displays, optional
  - Controller Tuner 14
  - Drive Tuner 8
- disturbance 30
  - rejection of 35
- drive
  - commanding a voltage to 8
    - offset 32
  - connections, verify polarity 10
  - drive fault detection, enabling 8
  - drive fault level 8
  - drive gain 8
  - Drive Tuner features 8
  - enable/disable 8
  - fault 8
  - shut down 8
  - tuning procedure 9
- editing the 6000 code 5
- emergency shutdown 2
- encoder
  - connections, verify polarity 10
  - feedback source 14, 27, 32
  - position 29
  - resolution 8, 19
  - scaling 8
- end-of-travel limits, enabling 14
- error, position, *see position, error*
- feedback data 27
- feedback polarity test 10
- feedback source selection and setup 14
- flow diagram for tuning 21

- gains
  - automatic gain selection 14, 20
  - definition of 27
  - gain set
    - definition of 32
    - enable 32
  - tach gain 10
  - tuning
    - controller 17
    - drive 9
    - tuning velocity drives 10
- graph setup, Controller Tuner 16
- hardware requirements 3
- help (on-line help system) 3
- I/O display 14
- instability 30
- installation procedure 4
- integral feedback control (SGI) 32, 35
- integral windup limit (SGILIM) 32, 36
- launching Servo Tuner modules 5
- LDT
  - feedback source 14, 27, 32
  - position 29
  - position sample rate 32
  - read error 32
  - setup (resolution, gradient, recirc, update rate) 19
- limits (end-of-travel), enabling 14
- maximum allowable position error 19
- motion display 14
- motion profile setup, Controller Tuner 15
- motion trajectory update 18
- move completion criteria 39
- number of axes 14
- offset to the command voltage 32
- on-line command reference 3
- on-line Following reference 3
- on-line help 3
- oscillatory servo response 31, 36
- output saturation 28
- over-damped servo response 30
- overshoot 31, 36
- PIV&F gains 34
- polarity of command and feedback response 10
- position
  - actual (based on feedback device) 29
  - commanded 28
  - error 29
    - maximum allowable 19, 32
    - reducing error at constant speed 23
    - reducing error during acceleration 23
    - reducing error during steady state 22
  - overshoot 36
  - position loop tuning 14
  - response (servo) 29
    - types 30
  - setpoint 28
  - tracking error 29
- program files, creating 5
- proportional feedback control (SGP) 32, 35
- read error, LDT 32
- README file 5
- recalling Controller Tuner sessions 14
- reference documentation 3
- reset the controller
  - from Controller Tuner 14
  - from Drive Tuner 8
- resolution
  - ANI 8, 9, 19
  - encoder 8, 9, 19
  - LDT 19
  - resolver 19
- resolver
  - feedback source 27, 32
  - resolution 19
- response – servo 30
- rise time 31
- sampling period & frequency, drive tuning 8
- saturation of the control output 27
- saving 6000 code 5
- saving servo tuner sessions 5
- scaling
  - ANI feedback 8
  - encoder feedback 8
- serial port, selecting 8, 14

- servo
  - control methods/types 34
  - sampling frequency 18, 27
- setpoint 29
- settling time 31
  - actual 41
- shut down in case of emergency 2
- software reference, on-line 3
- software requirements 3
- stability 30
  - verify command-to-response polarity 10
- status display 14
- steady-state 31
  - position error 29
  - reducing 22
- system update rate 18
  
- tach gain 10
- target zone 39
  - timeout error 40
- terminal emulator window 14
- time constant 8
- timeout error 40
- transient 31
- trigger point for data capture 15
- tuning 27
  - Controller Tuner features 14
  - controller tuning procedure 17
  - Drive Tuner features 8
  - drive tuning procedure 9
  - gains, definition 34
  - PIV&F algorithm 34
  - process flow diagram 21
  - related 6000 series commands 32
  - scenario (case example) 24
  - target zone mode 39
  - velocity drive tuning procedure 9
  
- under-damped servo response 30
- uninstall procedure 4
- unstable 30
- update rates 18
- user guide organization 2
  
- velocity feedback control (SGV) 32, 37
- velocity feedforward control (SGVF) 32, 38
  
- windup of the integral action 36