Appendix A

Servo Tuning

In a Hurry?

We strongly recommend tuning the APEX615n before attempting to execute any motion functions. If you **must** execute motion quickly (e.g., for testing purposes), you should at least complete this appendix's *Controller Tuning Procedure* and find a proportional feedback gain that gives a stable response for your system. Then you can proceed to execute your motion functions. Later on, you should read through this entire *Servo Tuning* appendix and follow its procedures to ensure your system is properly tuned.

Servo System Terminology

This section gives you an overall understanding of the principles and the terminology used in tuning the APEX615n Servo Controller/Drive.

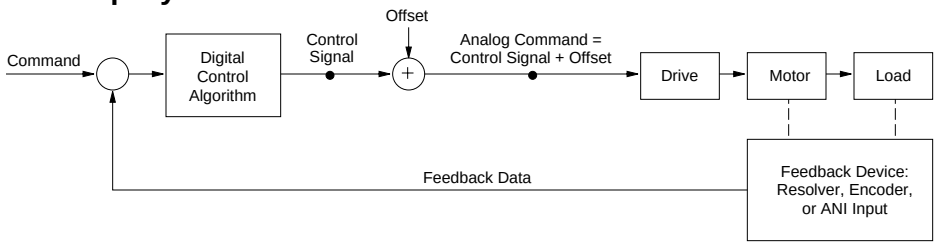
Servo Tuning Terminology

The APEX615n 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 several values from the previous 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 chapter). **Tuning** is the process of selecting and adjusting these gains to achieve optimal servo performance.

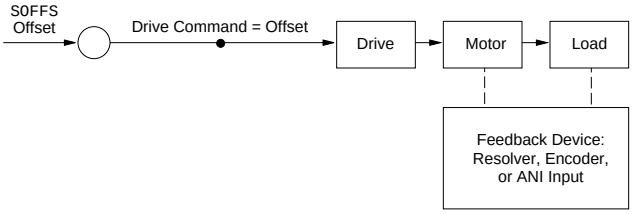
When this control algorithm is used, the whole servo system is a **closed-loop** system (see diagram below). It is called closed loop because the control algorithm accounts for both the **command** (position, velocity, tension, etc.) and the **feedback data** (from the resolver, encoder, or ANI input); 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 may be desirable to disable the algorithm and use the SOFFS command to directly control the motor current; then you can test the drive/motor operation independently from the controller.

Closed Loop System



Servo Algorithm Disabled



Internally, the APEX615n has two main sections—a *controller section* and a *drive section*. The controller accepts a motion command, and uses its digital control algorithm to calculate a *digital* control signal. This digital value is sent from the digital signal processor (DSP) to the digital-to-analog converter (DAC). The DAC has an analog output range of -10V to +10V.

The DAC's output, an *analog* control signal, is sent to the APEX615n's drive section. The drive produces motor current that is proportional to the voltage level of the analog signal.

It is possible that the digital control signal calculated by the control algorithm can exceed the ±10V limit. When this happens, the analog output will stay, or *saturate*, at the maximum limit until the position error changes such that the control algorithm calculates 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.

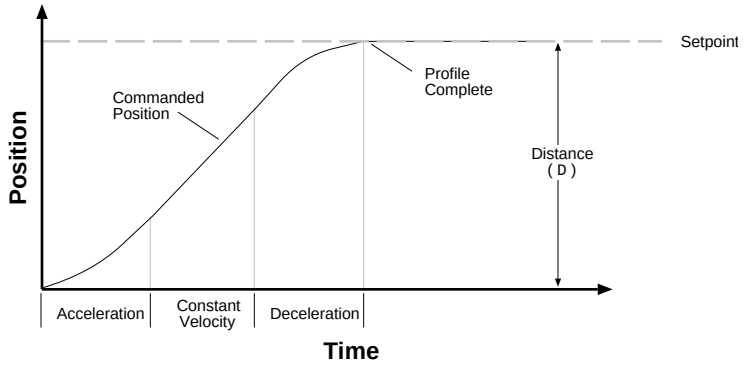
Position Variable Terminology

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 the 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 (resolver, encoder or ANI input). 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 limitations, 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

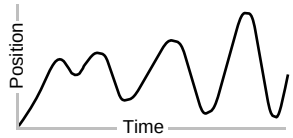
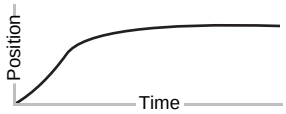
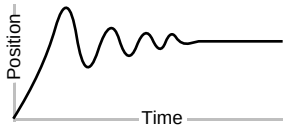
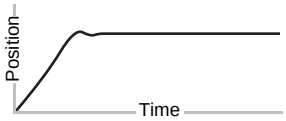
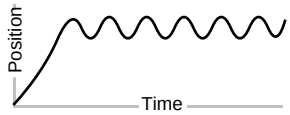
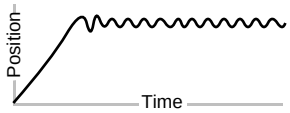
Stability

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 *over-damped*, system gives a smooth but slower response.	
Under-damped	A slightly damped, or *under-damped*, 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 sustained position oscillations of equal amplitude.	
Chattering	Chattering is a high-frequency, low-amplitude oscillation which is usually audible.	

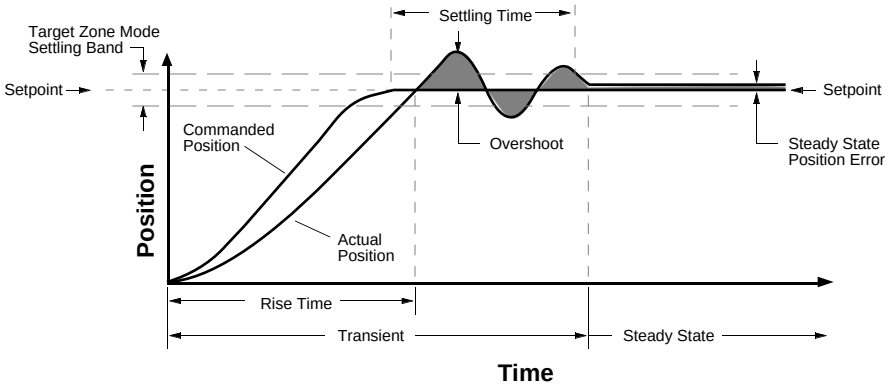
Performance Measurements

When you 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 a period of time the **transient**. When it is not moving, it is defined as **steady-state**.

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



6000 Series Servo Commands

NOTE

The following list briefly describes each servo-related 6000 Series command. More detailed information can be found in the rest of this chapter and within each command's description in the ***6000 Series Software Reference***.

Command	Title	Brief Description (detailed descriptions in *6000 Series Software Reference*)
SFB	*Select Servo Feedback Source*	Selects the servo feedback transducer. You can select resolver, encoder, or ANI feedback. (SFB4, resolver, is the default selection.)
SGAF	*Acceleration Feedforward Gain*	Sets the acceleration feedforward gain in the PIV&**F$_a$** servo algorithm.
SGENB	*Servo Gain Set Enable*	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	*Set Integral Feedback Gain*	Sets the integral gain in the P**I**V&F servo algorithm.
SGILIM	*Set Integral Windup Limit*	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	*Proportional Feedback Gain*	Sets the proportional gain in the **P**IV&F servo algorithm.
SGSET	*Save a Set of Servo Gains*	Saves the presently-defined set of PIV&F gains as a particular *gain set* (specific to the current feedback source). 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	*Set Velocity Feedback Gain*	Sets the velocity gain in the PI**V**&F servo algorithm.
SGVF	*Velocity Feedforward Gain*	Sets the velocity feedforward gain in the PIV&**F$_v$** servo algorithm.
SMPER	*Maximum Allowable Position Error*	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 APEX615n shuts down power output to the motor. 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	*Servo Control Signal Offset*	Sets an offset to the commanded analog output voltage, which is sent to the drive system.
SSFR	*Servo Frequency Ratio*	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 then the servo position correction. This command allows you to optimize this for your application. The default setting (SSF4) is sufficient for most applications.

STRGTE STRGTD STRGTV STRGTT	*Target Zone Mode Enable* *Target Zone Distance* *Target Zone Velocity* *Target Zone Timeout Period*	When using the Target Zone Mode, enabled with the STRGTE command, the actual position and actual velocity must be within the *target zone* (that is, within the distance zone defined by STRGTD and within the velocity zone defined by STRGTV). If the motor/load does not settle into the target zone before the timeout period set by STRGTT, the APEX615n detects an error. To prevent subsequent commands/moves from being executed when this error condition occurs, you must 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. Otherwise, subsequent commands/moves can be executed regardless of the actual position and velocity. This feature is explained in greater detail later in the *Target Zone* section.
TFB and [FB]	*Position of Servo Feedback Devices*	Transfers [or assigns/compares] the actual position of the transducer selected for feedback (see SFB).
TDAC and [DAC]	*Value of DAC Output*	Transfers [or assigns/compares] the output from the APEX615n's digital-to-analog converter. This is the analog control signal output to the APEX615n's internal drive.
TGAIN	*Transfer Servo Gains*	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 (SGI, SGP, SGV, SGAF, and SGVF), or the last gain set enabled with the SGSET command.
TPC and [PC]	*Position Commanded*	Transfers [or assigns/compares] the commanded position (intermediate position setpoint).
TPER and [PER]	*Position Error*	Transfers [or assigns/compares] the error between the commanded position (TPC) and the actual position (TFB, TPE, or TANI) as measured by the feedback device.
TSGSET	*Transfer Servo Gain Set*	Transfers a previously-saved set of servo gain parameters. A gain set is saved with the SGSET command.
TSTLT	*Transfer Servo Settling Time*	Transfers the time it took the last move to settle within the *target zone* (that is, within the distance zone defined by STRGTD and within the velocity zone defined by STRGTV). The Target Zone Mode does not need to be enabled to use this command.

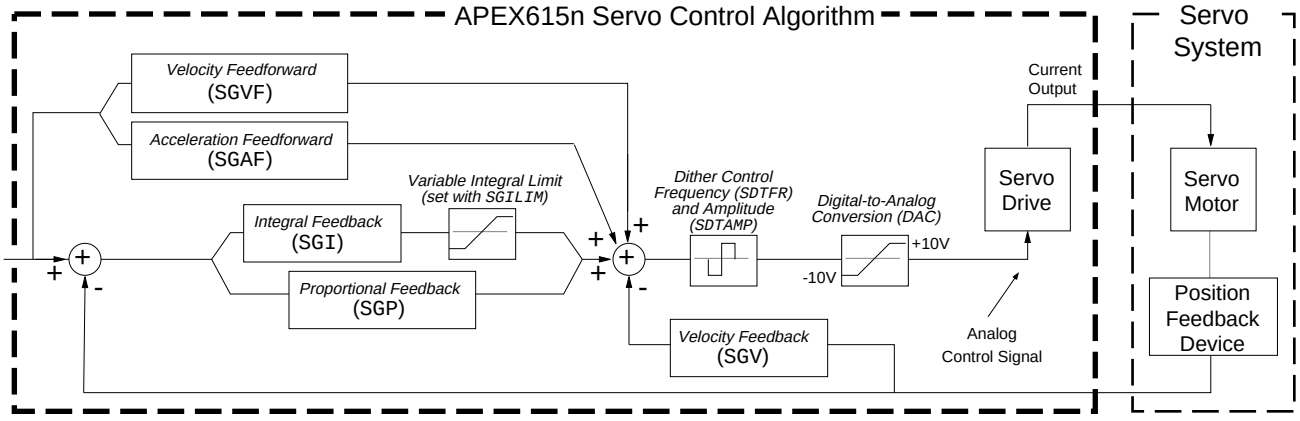
Servo Control Techniques

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

The APEX615n 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 following block diagram shows these control techniques in relation to the servo control algorithm configuration. The 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 APEX615n 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 forcing the load away from its commanded position (such as torque ripple or a spring load), 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.

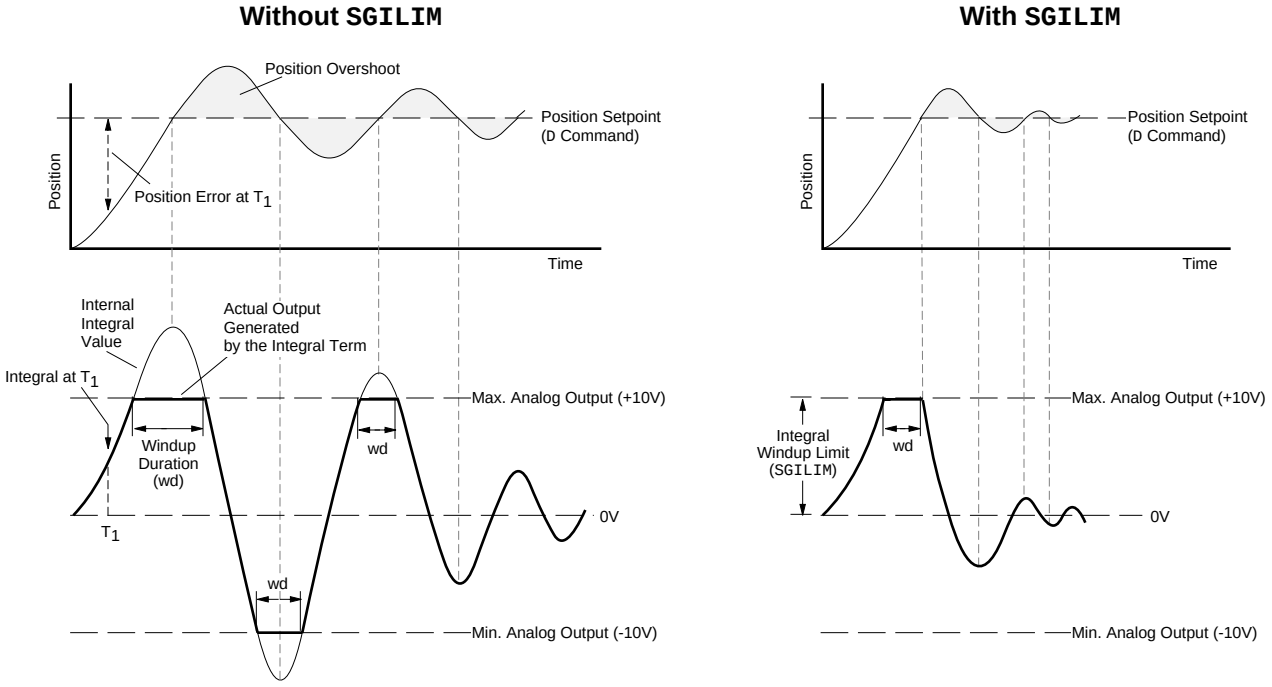
The primary function of the integral control is to overcome friction and/or gravity and to reject disturbances so that steady state position error can be minimized or eliminated. This control action is important for achieving high system accuracy. *However, if you can achieve acceptable position accuracy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

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 *integral 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.



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 based 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 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 APEX615n essentially anticipates the commanded position and initiates a control signal ahead of time to more closely follow (*track*) the commanded position.

Applications requiring contouring or 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 APEX615n's analog control signal range of ±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 APEX615n 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.

Controller Tuning Procedure

The *Controller Tuning Procedure* leads you through the following steps:

1. Turn on AC power to the APEX615n.
2. Setup up for tuning.
3. Select the 615n's servo Sampling Frequency Ratios (SSFR).
4. Set the Maximum Position Error (SMPER).
5. Optimize the Proportional (SGP) and Velocity (SGV) gains.
6. Use the Integral Feedback Gain (SGI) to reduce steady state error.
7. Use the Velocity Feedforward Gain (SGVF) to reduce position error at constant velocity.
8. Use the Acceleration Feedforward Gain (SGAF) to reduce position error during acceleration and deceleration.

Tuning with Servo Tuner™:

Compumotor also offers Servo Tuner™, a Microsoft® Windows™ based program designed to help you tune your motion control servo system. Refer to the Servo Tuner User Guide for tuning procedures.

Before you tune the 615n:

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 with the SFB command and repeat steps 4-8.

EMERGENCY SHUTDOWN

If you need to shutdown the APEX615n during the tuning process (for instance, if the system becomes unstable or experiences a runaway), issue the DRIVEØ command.

Step 1 **Turn on AC power to energize the APEX615n.**

Step 2 Use a computer (with a terminal emulator) or a dumb terminal to enter the commands noted in the steps below. To monitor system performance, you may use visual inspection, or use an analog type position transducer (potentiometer, LVDT, RVDT, etc.) to pick up the load's or motor's position displacement and monitor the transducer output on a digital storage oscilloscope.

Step 3 **Select the Sampling Frequency Ratio (SSFR):**

The APEX615n's control signal is computed by the digital signal processor (DSP). The velocity of the commanded position, the velocity of the feedback device's position, and the integral of the position error are used for various control actions. These measurements are derived by the DSP from the position values sampled periodically at a fixed rate; this sampling rate is called the *servo sampling frequency* (samples/second).

NOTE

The SSFR setting affects the dither frequency ratio (SDTFR setting). Refer to the SSFR and SDTFR command descriptions in the ***6000 Series Software Reference*** for details.

Higher sampling frequencies improve the accuracy of the derived velocity and integral values. A higher sampling frequency can also improve the tracking of a rapidly changing or oscillating position. Therefore, the servo sampling frequency is a key parameter that influences the servo system's stability and closed loop bandwidth.

In addition to computing the APEX615n's control signal, the DSP also computes the commanded position trajectory. When the servo sampling frequency is increased, the motion trajectory update rate has to be decreased, and vice versa. The ratio between the servo sampling frequency and the trajectory update rate, called the *sampling frequency ratio*, depends on the requirements of your application and/or the dynamic characteristics of the system. The Servo Sampling Frequency Ratio (SSFR) command offers four selectable ratio settings. These four ratios and the actual sampling frequencies and sampling periods (reciprocal of sampling frequency) are shown below.

SSFR	Servo Sampling Update		Motion Trajectory Update		System Update	
Command Setting	Frequency (samples/sec.)	Period (μsec)	Frequency (samples/sec.)	Period (μsec)	Frequency (samples/sec.)	Period (μsec)
SSFR1	3030	330	3030	330	757	1320
SSFR2	5405	185	2703	370	675	1480
SSFR4	6250	160	1563	640	520	1920
SSFR8	6667	150	833	1200	417	2400

The general rule for 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 350 Hz, you can use the following guideline to determine the minimum servo sampling frequency: set the servo sampling frequency at least 8 times higher than the bandwidth frequency. The required minimum servo sampling frequency would be 2800 Hz.

The table below provides guidelines for various application requirements.

Application Requirement	SSFR1	SSFR2	SSFR4	SSFR8
XY Linear Interpolation	4	4		
Fast point-to-point motion			4	4
Regulation (speed, torque, etc.)			4	4
High natural frequency system				4

Setting the Sampling Frequency Ratio

Select a sampling ratio (with the SSFR command) appropriate to your system now, before you proceed to tune each gain.

If you change the sampling frequency ratios (SSFR) after the tuning is complete and the new servo sampling frequency is lower than the previous one, the response may change (if your system bandwidth is quite high) and you may have to re-tune the system.

Step 4 **Set the Maximum Position Error (SMPER)**:

The SMPER command allows you to set the maximum position error allowed before an error condition occurs. The position error, monitored once per system update period, is the difference between the commanded position and the actual position as read by the feedback device selected with the last SFB command. *Larger values allow greater oscillations/motion when unstable; therefore, smaller SMPER values are safer.*

When the position error exceeds the value entered by the SMPER command, an error condition is latched (see TAS or AS bit #23) and the 6000 controller issues a shutdown to the faulted axis and sets its analog output command to zero volts. To enable the system again, the appropriate DRIVE1 command must be issued, which also sets the commanded position equal to the actual feedback device position (incremental devices will be zeroed).

If the SMPER value is set to zero, the position error condition is not monitored, allowing the position error to accumulate without causing a fault.

Step 5 **Optimize the Proportional (SGP) and Velocity (SGV) gains (*see illustration for tuning process*)**:

a. Enter the following commands to create a step input profile *(use a comma in the first data field when tuning axis 2—e.g.,* D,1ØØ*)*:

Command	Description
> A999	Set acceleration to 999 units/sec^2
> AD999	Set deceleration to 999 units/sec^2
> V3Ø	Set velocity to 30 units/sec
> D1ØØ	Set distance to 100 units

b. Start with an SGP command value of 0.5 (SGPØ.5 or SGP,Ø.5).

c. Enter the GO1 or GO,1 command depending on which axis is being tuned at the time.

d. Observe the plot of the commanded position versus the actual position on the oscilloscope. If the response is already very oscillatory, lower the gain (SGP); if it is *sluggish* (overdamped), increase the SGP gain.

Repeat Steps 4.c. and 4.d. until the response is slightly under-damped.

e. Start with an SGV command value of 0.1 (SGVØ.1 or SGV,Ø.1).

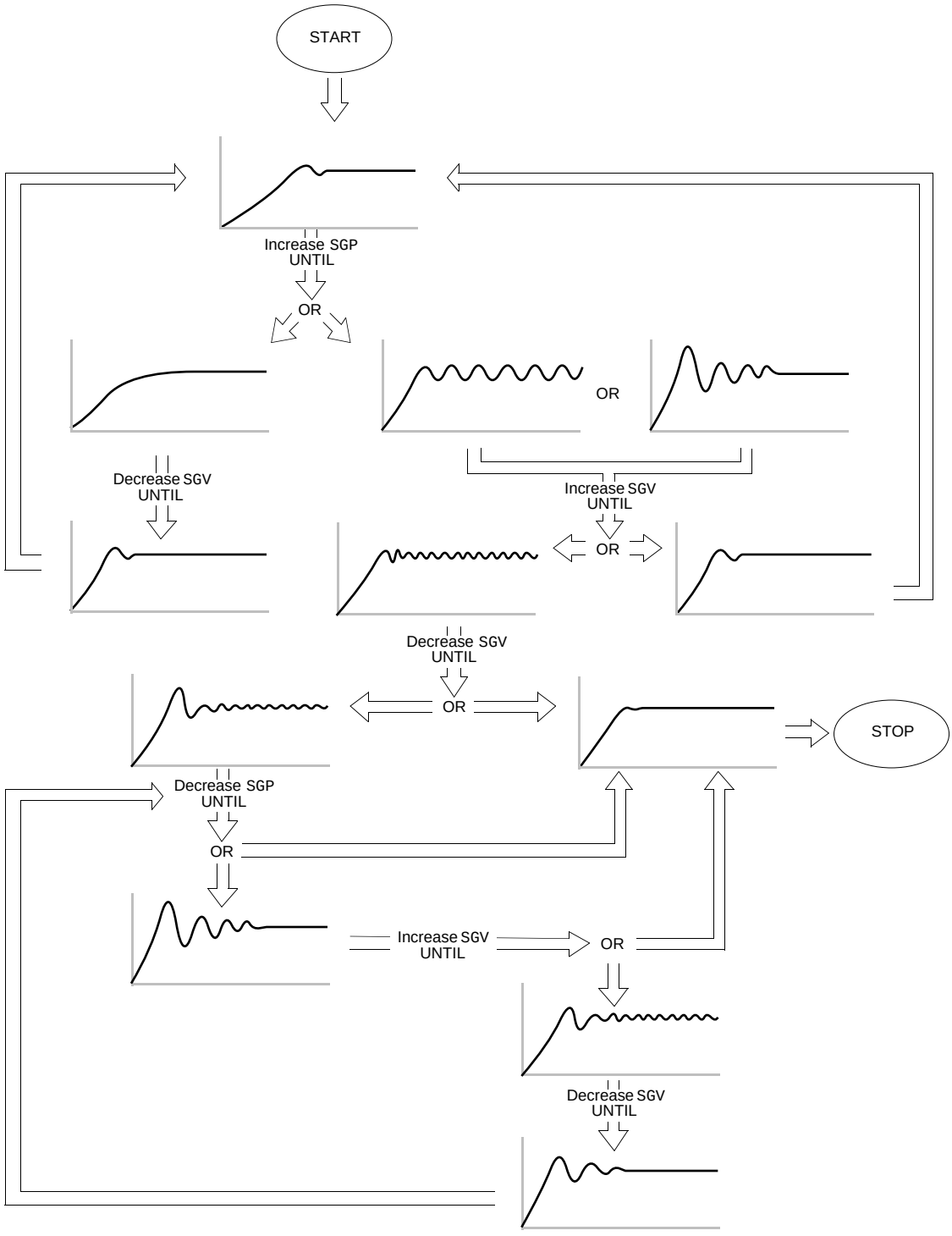
f. As you did in Step 4.c., enter GO1 or GO,1.

g. Observe the plot on the oscilloscope. If the response is *sluggish* (overdamped), reduce the SGV gain. *Repeat Steps 4.f. and 4.g. until the response is slightly under-damped.*

Refer to the Tuning Scenario *section later in this chapter for a case example.*

h. 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. 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 valve, hydraulic cylinder, etc.) to chatter.


START
Increase SGP
UNTIL
OR
OR
Decrease SGV
UNTIL
Increase SGV
UNTIL
OR
Decrease SGV
UNTIL
OR
STOP
Decrease SGP
UNTIL
OR
Increase SGV
UNTIL
OR
Decrease SGV
UNTIL

Step 6 **Use the Integral Feedback Gain (SGI) to reduce steady state error:**

a. Determine the steady state position error (the difference between the commanded position and the actual position). You can determine this error value by the TPER command when the load is not moving.

☞ *Steady state position error is described earlier in the* Performance Measurements *section.*

> **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**. For hydraulic applications, it is usually best to use a small SGI value, or use SGIØ while moving and use SGIn when stopped. The use of the Target Zone Settling Mode (STRGTE) is recommended.

b. If you have to enter the integral feedback gain to reduce the steady error, start out with a small value (e.g., SGIØ.1). After the gain is entered, observe two things from the response:

- Whether or not the magnitude of steady state error reduces
- Whether or not the steady state error reduces to zero at a faster rate

c. Keep increasing the 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.):

1st Lower the integral gain (SGI) value to reduce the overshoot.

☞ *If you are using current control, convert the offset from milliamps to volts and enter the result in the* SGILIM *command.*

2nd Check whether the 615n's analog output saturates the ±10V limit; you can do this by observing the signal from a digital oscilloscope. If it saturates, then lower the integral output limit by using the SGILIM command. 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.*

3rd 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 or valve to chatter).

Step 7 **Use the Velocity Feedforward Gain (SGVF) to reduce position error at constant speed:**

a. Execute a continuous (MC1 command) move, setting the acceleration, deceleration and velocity values appropriate to your application. Set the SGVF value to be the product of SGP ∗ SGV (if SGV = zero, set SGVF equal to SGP).

b. Check the position error at constant velocity by issuing the TPER command.

c. Increase SGVF to reduce the position error (repeat steps a and b as necessary).

Step 8 **Use the Acceleration Feedforward Gain (SGAF) to reduce position error during acceleration:**

a. Execute a continuous (MC1 command) move, setting the acceleration, deceleration and velocity values appropriate to your application. Set SGAF to 0.01 (SGAFØ.Ø1).

b. Check the position error during acceleration by issuing the TPER command.

c. Increase SGAF to reduce the position error (repeat steps a and b as necessary).

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 5 of the *Tuning Procedure*.

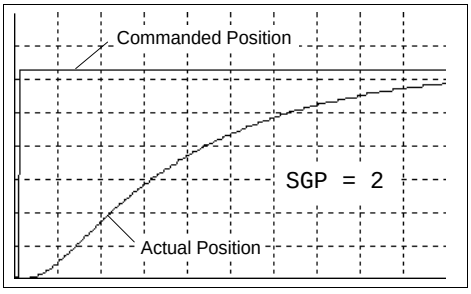
NOTE

The steps shown below (steps 1 - 11) represent the major steps of the process; the actual progression between these steps usually requires 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 Motion Architect's Controller Tuner Module (X axis = time, Y axis = position).

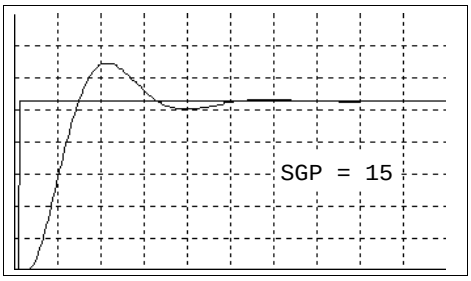
Step 1 For a starting trial, we set the proportional feedback gain (SGP) to 2. As you can see by the plot, the response is slow.

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



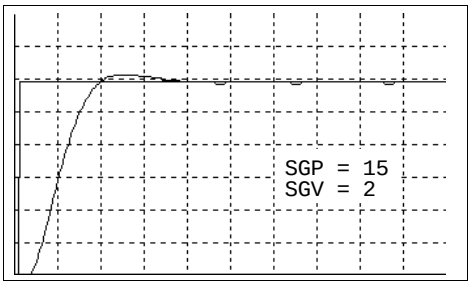
Step 2 With SGP equal to 15, the response becomes slightly underdamped (see plot).

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



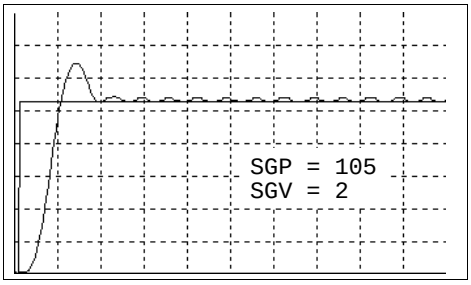
Step 3 With SGV equal to 2, the response turns out fairly well damped (see plot).

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



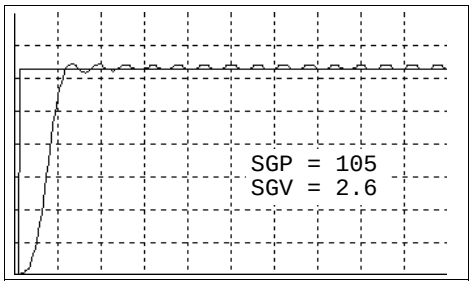
Step 4 As we iteratively increase SGP to 105, overshoot and chattering becomes significant (see plot). This means either the SGV gain is too low and/or the SGP is too high.

Next, we should try raising the SGV gain to see if it could damp out the overshoot and chattering.



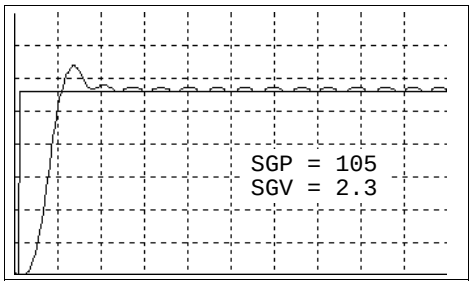
Step 5 After the SGV gain is raised to 2.6, the overshoot was reduced but chattering is still quite pronounced. This means either one or both of the gains is too high.

The next step should be to lower the SGV gain first.



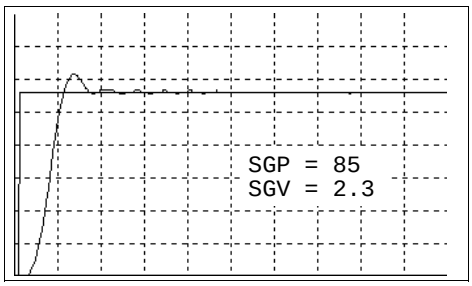
Step 6 Lowering the SGV gain to 2.3 does not help reduce the chattering by much.

Therefore, we should lower the SGP gain until chattering stops.



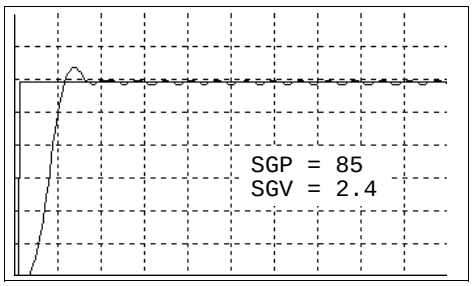
Step 7 Chattering stops after reducing the SGP gain to 85. However, the overshoot is still a little too high.

The next step should be to try raising the SGV to damp out the overshoot.



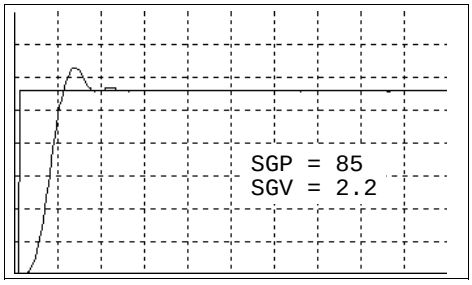
Step 8 After raising the SGV gain to 2.4, overshoot is reduced a little, but chattering reappears. This means the gains are still too high.

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



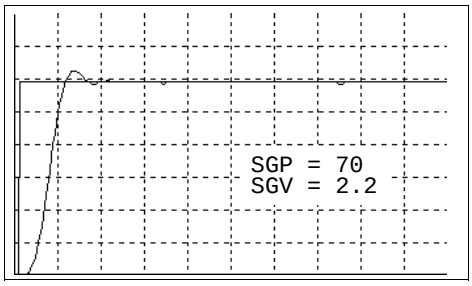
Step 9 After lowering the SGV gain to 2.2 (even less than in Step 7—2.3), chattering stops.

Next we should lower the SGP gain.

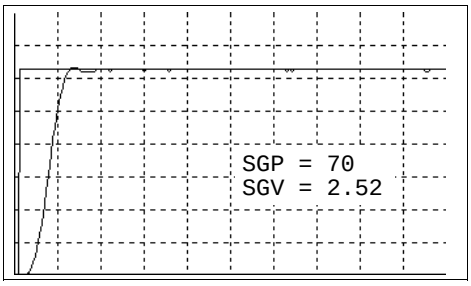


Step 10 Overshoot is reduced very little after lowering the SGP gain to 70. (The SGV gain might have been lowered too much in Step 9.)

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

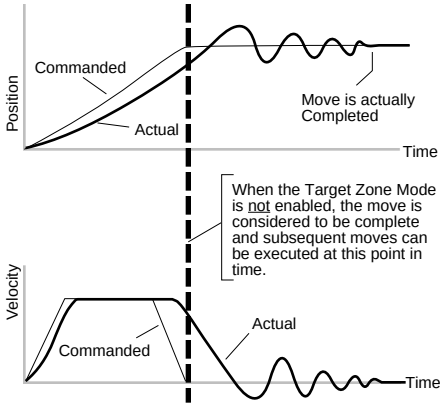


Step 11 When we raised the SGV gain to 2.52, the step response became fast and very stable.



Target Zone (Move Completion Criteria)

Under default operation (Target Zone Mode not enabled), the APEX615n's move completion criteria is simply derived from the move trajectory. The APEX615n 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).



Target Zone Mode

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 STRGTE 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 APEX615n detects a *timeout error* (see illustration below).

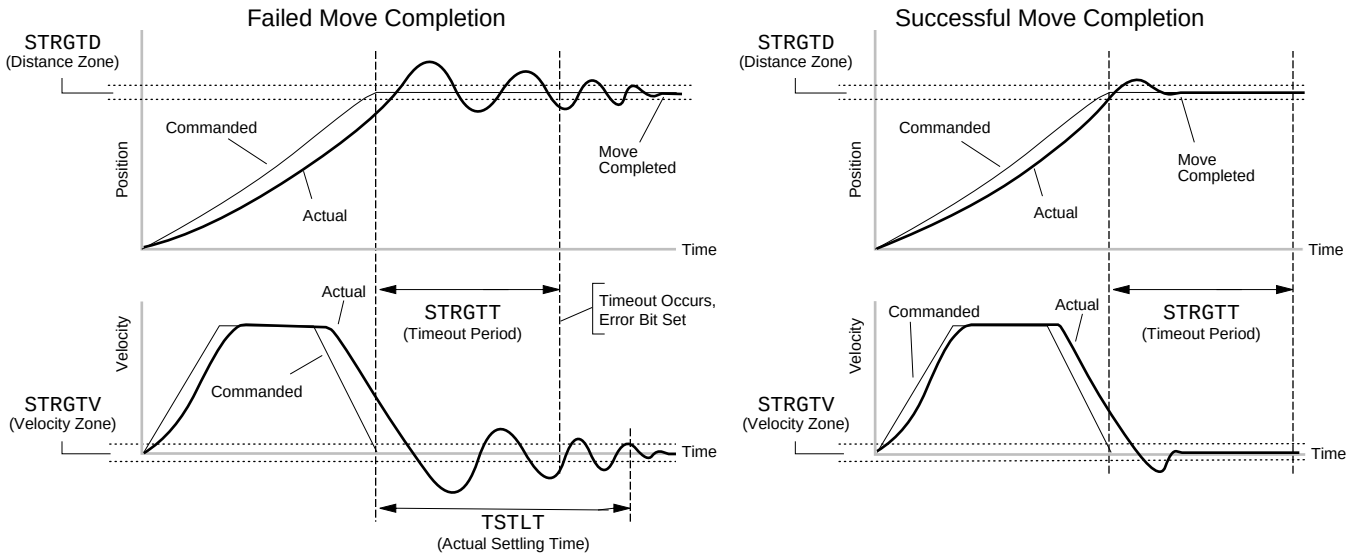
Refer to the Error Handling *section in the* **6000 Series Software Reference** *for error program examples*

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, and when it occurs to branch to a programmed response you can define in the ERRORP program.

As an example, setting the distance zone to ±5 steps (STRGTD5), the velocity zone to ≤0.5 rps (STRGTVØ.5), and the timeout period to 1/2 second (STRGTT5ØØ), a move with a distance of 8,000 steps (D8ØØØ) must end up between position 7,995 and 8,005 and settle down to ≤0.5 rps 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 the actual velocity falls within the target velocity zone set with the STRGTV command (see illustration 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.**