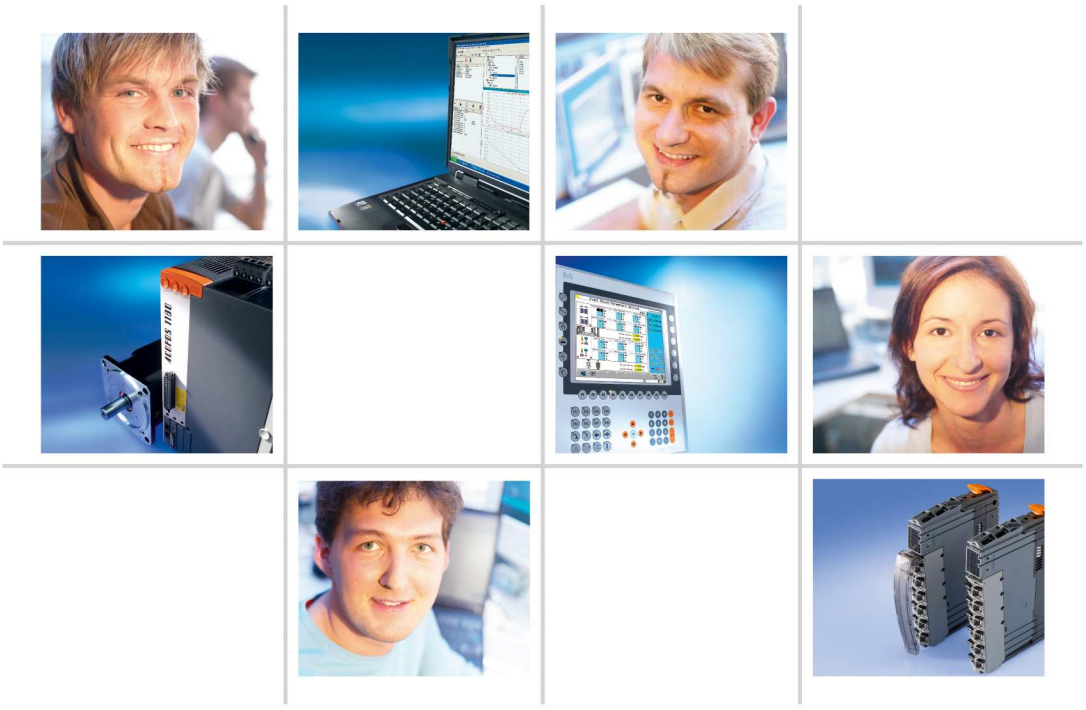
ACOPOS Control Concept and Adjustment

TM450



Perfection in Automation

www.br-automation.com

Prerequisites

Training modules:	TM410 – Motion Components Basis
Software:	Automation Studio 2.5 or higher
	Automation Runtime 2.80 or higher
	ACP10-SW V1.160 or higher
Hardware:	8V1xxx.yy-2

Table of contents

1. INTRODUCTION 4
   1.1 Objectives 5
2. THE BASICS OF CONTROL TECHNOLOGY 6
3. CASCADED CONTROLLER CONCEPT 10
   3.1 Overview 10
   3.2 Set value generator 11
   3.3 Predictive position controller 15
   3.4 Speed controller 18
   3.5 Current controller 22
4. THEORETICALLY DETERMINING THE CONTROL PARAMETERS 23
   4.1 Speed controller 23
   4.2 Position controller 24
5. PROCEDURE FOR SETTING THE CONTROLLER 26
   5.1 General information 26
   5.2 Speed controller 29
   5.3 Position controller 35
   5.4 Limit parameter 41
   5.5 Overview 44
6. SAVING THE CONTROLLER SETTINGS 45
   6.1 NC INIT parameter module 45
   6.2 ACOPOS parameter table 45
7. SUMMARY 46
8. APPENDIX 47

1. INTRODUCTION

The high level of performance of a servo drive's controller is crucial for the quality, precision and dynamic capabilities during a process. This means that the control concept as well as the controller settings are decisive factors.

Fig. 1 ACOPOS servo drive and motor

During this training we will first learn the basics before taking a step-by-step look at the control concept of the ACOPOS servo drive.

We will then take some time learning how to determine the optimum control parameters. Some of the components from Motion Components (training module TM410) will help us throughout this training module.

1.1 Objectives

The course participant will become familiar with the structure and the effectiveness of the ACOPOS control concept.

Each course participant will be able to adjust and optimize the control parameters.

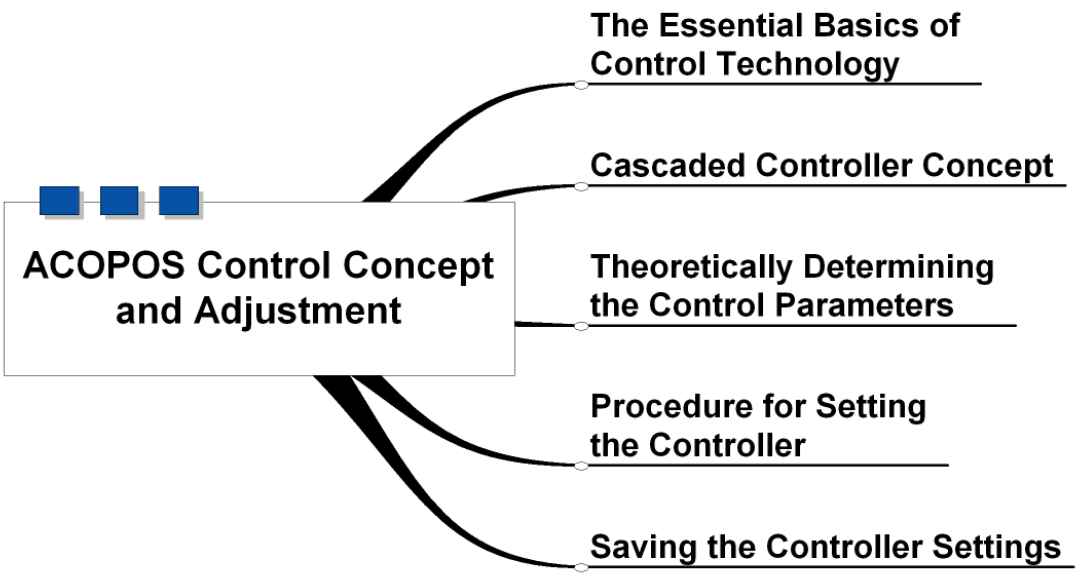


Fig. 2 Overview

2. THE BASICS OF CONTROL TECHNOLOGY

To begin our training, we would like to briefly review a few basics before diving into the details of the ACOPOS control concept.

Let's start by asking ourselves the following question:

Why does a servo drive need a controller?

The goal of a servo drive is to reach a position as fast and as accurately as possible using a motor and various mechanics.

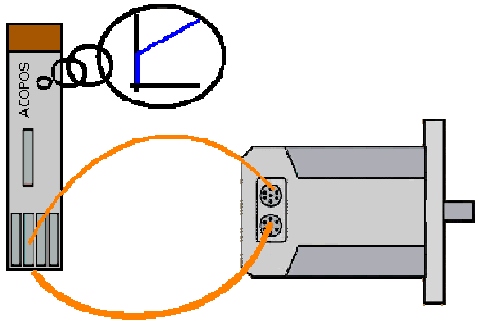


Fig. 3 ACOPOS servo drive and motor

The first thing the controller must know is where to send the motor. This is generally referred to as the **set position**. In control technology, a **reference variable** is used to do this.

The controller must receive some information to find out where the motor is presently located. This information is obtained using an encoder. The encoder is a **measuring element**, which provides the controller with the information (actual position of the motor) via **feedback**.

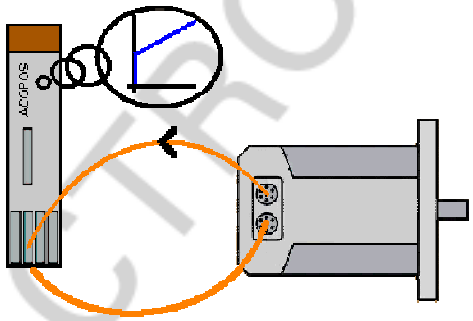


Fig. 4 ACOPOS servo drive, motor and feedback

A **lag error** or **control deviation** occurs when the set position and the actual position do not match.

This lag error is compensated for by the servo drive. This is why it outputs a **manipulated variable**. This manipulated variable affects the **controlled system** (in this case, the motor and the subsequent mechanics) so that the actual position reaches the set position. In our case, the actual position is the **control variable**.

All of this together is called the **closed control loop**.

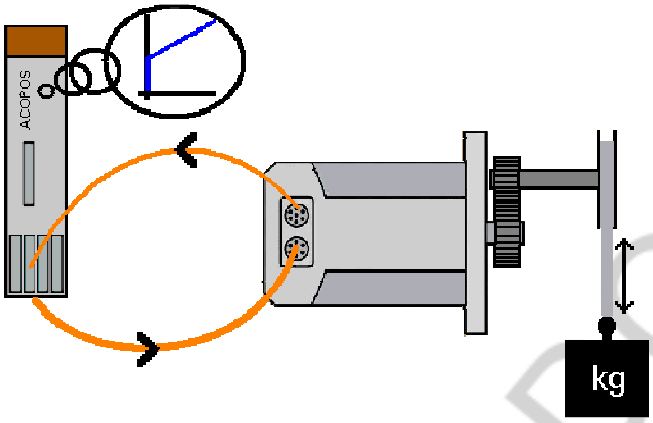


Fig. 5 ACOPOS servo drive, motor, closed control loop and disturbance variables

There are even more external factors that affect this system. These types of factors are called **disturbance variables**, and also must be compensated for by the controller. A suspended load is one example of a disturbance variable.

Differences between closed and open loop controllers:

Unlike a closed loop controller, an open loop controller does not have feedback. This means we cannot determine when, how or whether the goal has been reached. An open loop controller is generally used for conventional frequency converters.

But now what happens within a closed loop controller?

A closed loop controller can be made up of one or more parts of transfer elements. These transfer elements react differently to input values.

The controllers on the ACOPOS servo drive are generally made up of a **proportion element** (P-element) and an **integral element** (I-element).

A P-element immediately reacts to an input value jump with a proportional output value jump. The size of the jump on the output is determined by a factor. This is labeled as the factor "**kv**".

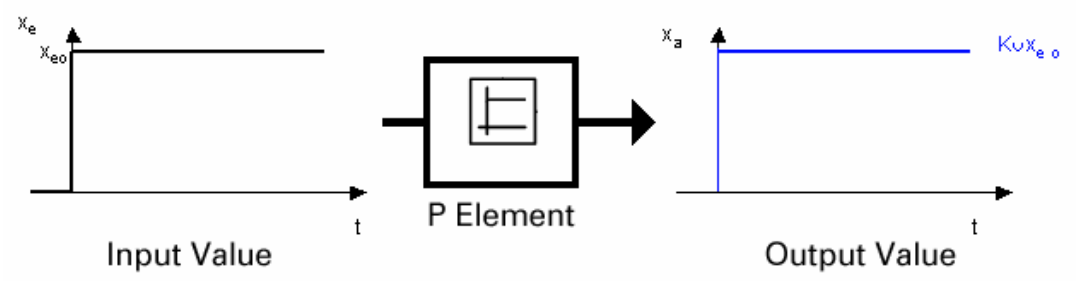


Fig. 6 Reaction of a P-element

The output value increases continuously in the form of a ramp if the input value of an I-element jumps. The slope of this ramp and the rate at which the output value increases depends on the time. We will label the time as integral action time "**tn**".

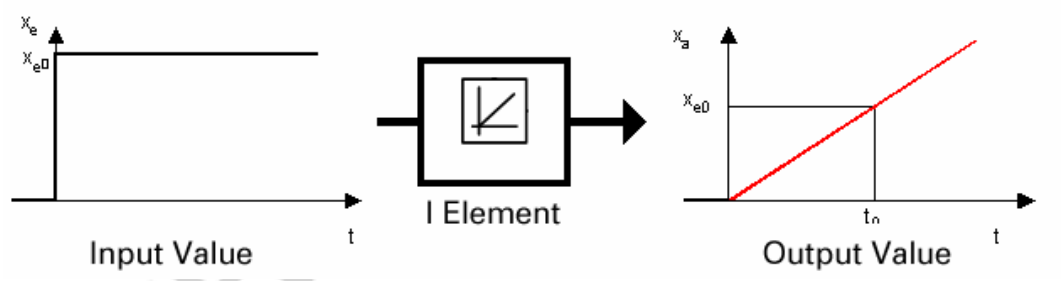


Fig. 7 Reaction of an I-element

A **PI controller** contains a P- and an I- section. The output values of both elements are added together. As a result, the PI-controller reacts to a jumping input value as follows.

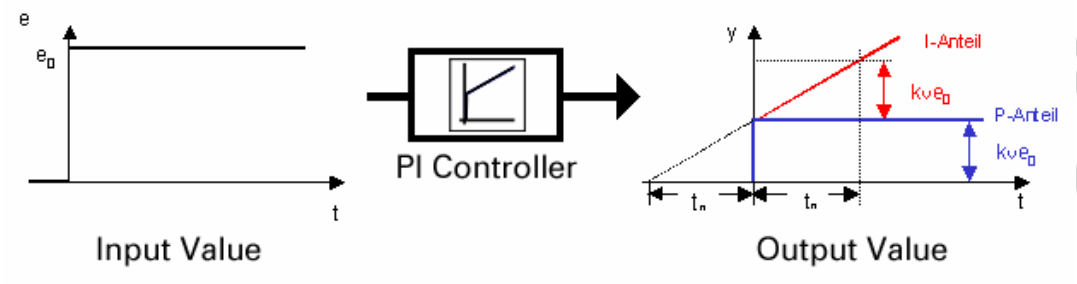


Fig. 8 Reaction of a PI-controller

The P-section allows the controller to react immediately to a change in the input value. A remaining controller deviation would occur if only one P-section was used alone. The I-section integrates this, thereby compensating for the deviation.

The ACOPOS servo drive is equipped with a high-performance processor. Among other tasks, this processor also calculates the controller algorithms. This is why the term "digital control" is used. The difference between an analog (continuous) controller and a digital (discrete time) controller is that the controller deviations is scanned in a corresponding timeframe (cycle). The duration of this cycle should always be the same (= jitter-free), but should also be as short as possible to receive the changes in the controller deviation as fast as possible.

3. CASCADED CONTROLLER CONCEPT

3.1 Overview

The cascaded controller concept is the core of the ACOPOS servo drive. The **position controller**, **speed controller** and **current controller** are cascaded starting with the set value generator. As a result, the manipulated variable of the higher-level controller becomes the reference variable for the lower-level controllers (e.g.: the position controller determines the set speed for the speed controller).

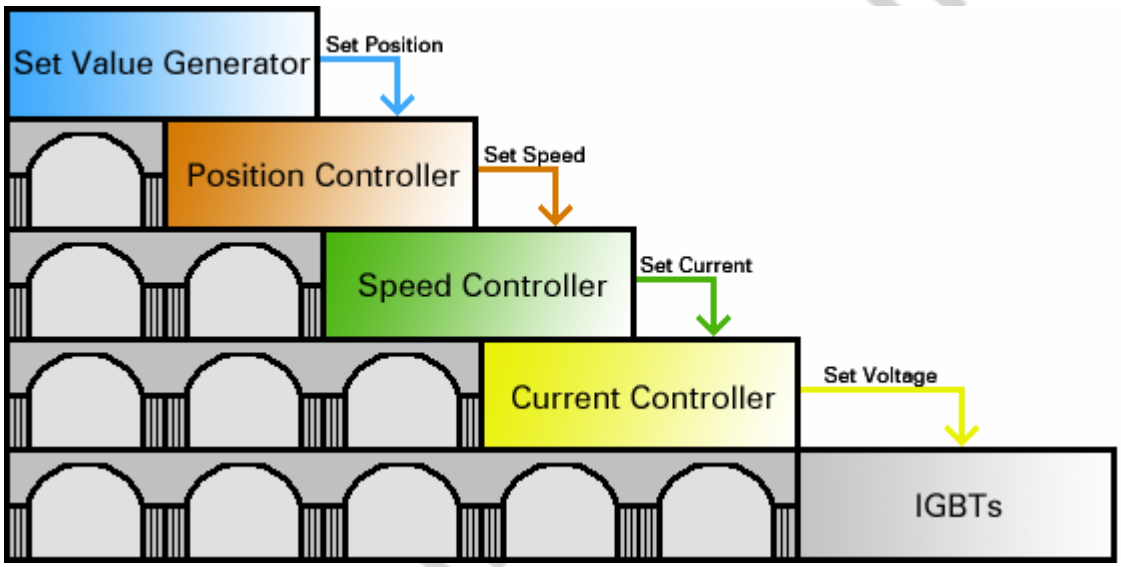


Fig. 9 Cascaded structure

3.2 Set value generator

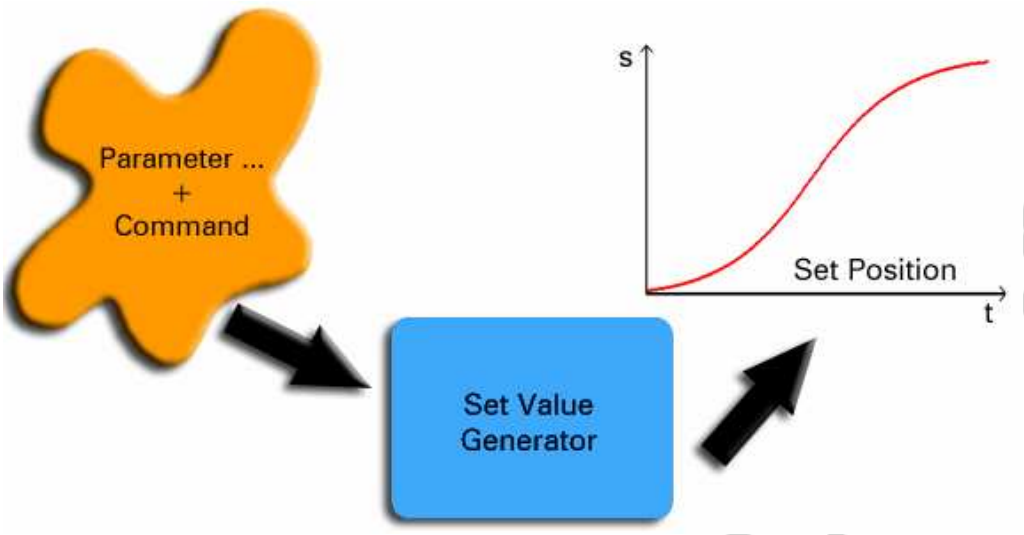


Fig. 10 Principle

3.2.1 Basis movements

As seen earlier, the reference variable is provided for the position controller by a set value generator. This is done with a cycle time of **400µs**.

The job of this set value generator is to create a movement profile after a command for executing a basis movement. The course of this profile depends mostly on the **basis movement parameters** (target position, acceleration, etc.).

Element	Data Type	Description
...		
move		**Movement**
...		
basis		**Basis Movements**
...		
parameter		**Parameters**
s	DINT	Target position or relative move distance [Units]
v_pos	REAL	Speed in positive direction [Units/sec]
v_neg	REAL	Speed in negative direction [Units/sec]
a1_pos	REAL	Acceleration in positive direction [Units/sec^2]
a2_pos	REAL	Deceleration in positive direction [Units/sec^2]
a1_neg	REAL	Acceleration in negative direction [Units/sec^2]
a2_neg	REAL	Deceleration in negative direction [Units/sec^2]

Fig. 11 Automation Studio online help, basis movements

The following figure illustrates this type of profile and the parameter values used:

ax_par1	Value	Unit
s	10000	[Units]
v_pos	10000	[Units/s]
v_neg	10000	[Units/s]
a1_pos	100000	[Units/s²]
a2_pos	100000	[Units/s²]
a1_neg	100000	[Units/s²]
a2_neg	100000	[Units/s²]

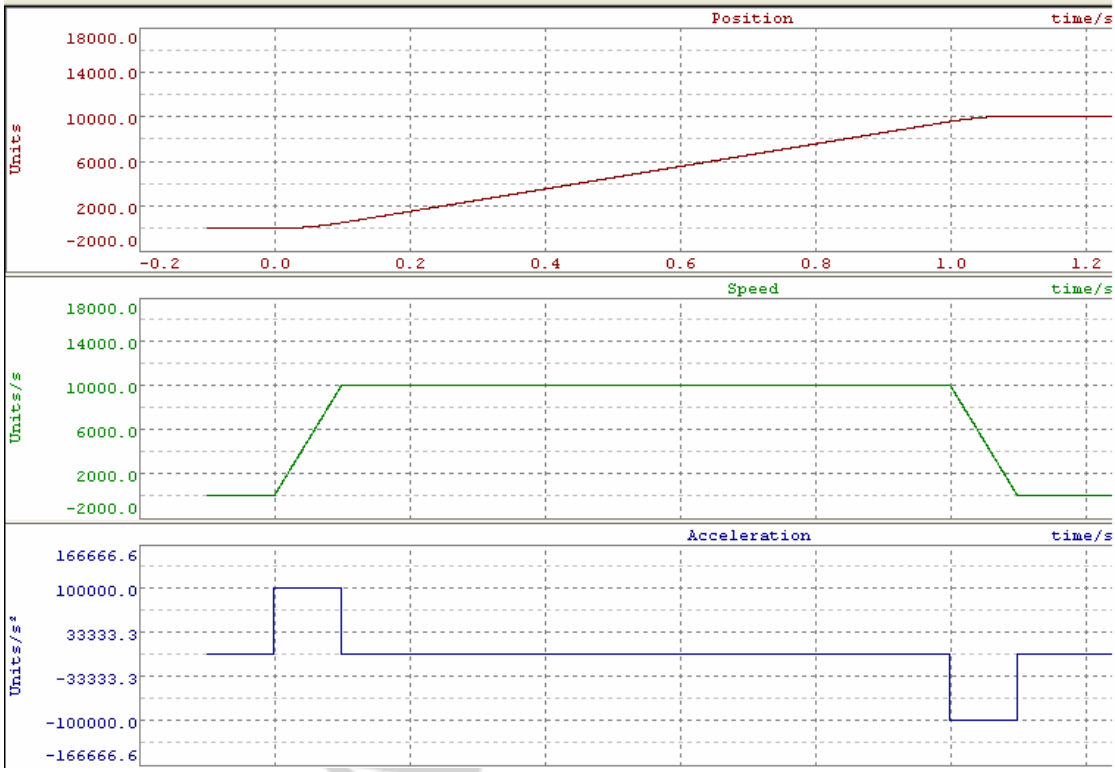


Fig. 12 Trace display of the position, speed and acceleration (derivative of the speed)

We can see from the chart that jumps occur at each end of the acceleration. These jumps are called **jolts**. These occur when there is a bend in the respective speed curve.

Generally this type of behavior is not desired because the motor must generate a higher torque. Furthermore, a high load is placed on the mechanics and the entire system vibrates.

This is why the set value generator in an ACOPOS servo drive is provided with jolt limitation.

3.2.2 Jolt limitation

As we have already learned, the jolt occurs due to a change that causes a bend in the speed curve. If the speed is slowly increased at the beginning or slowly reduced at the end of the acceleration phase, the rectangular acceleration profile becomes a trapezoid. The following chart illustrates this point for us (the basis movement parameters were set the same as in Fig. 12):

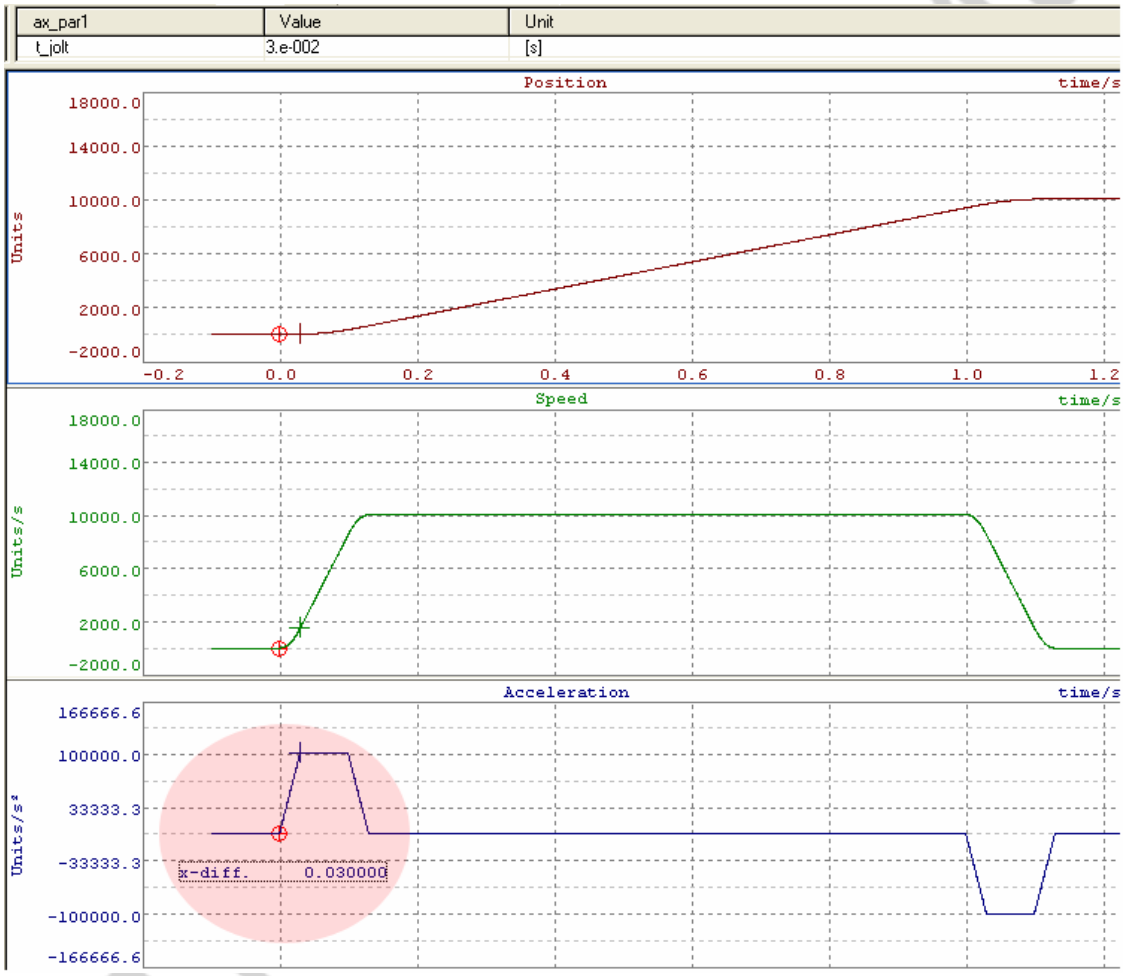


Fig. 13 Trace display of the position, speed and acceleration with active jolt filter

A jolt filter time can be set for the ACOPOS servo drive to generate a movement profile with jolt limitation.

The jolt filter time is the time required for acceleration from zero to the maximum value defined. Jolt limitation uses a linear filter during runtime.

In Fig. 13 we can see that the jolt filter time "**t_jolt**" has been set to 0.03 seconds. The measurement cursors in the lower chart diagram indicate that the rise time of the acceleration is the same as the jolt filter time.

An active jolt limitation extends the time for the set value generation (see Fig. 14). However, in many cases the positioning goal can be reached sooner because the settling time of the system is considerably shortened by the lower jolt load.

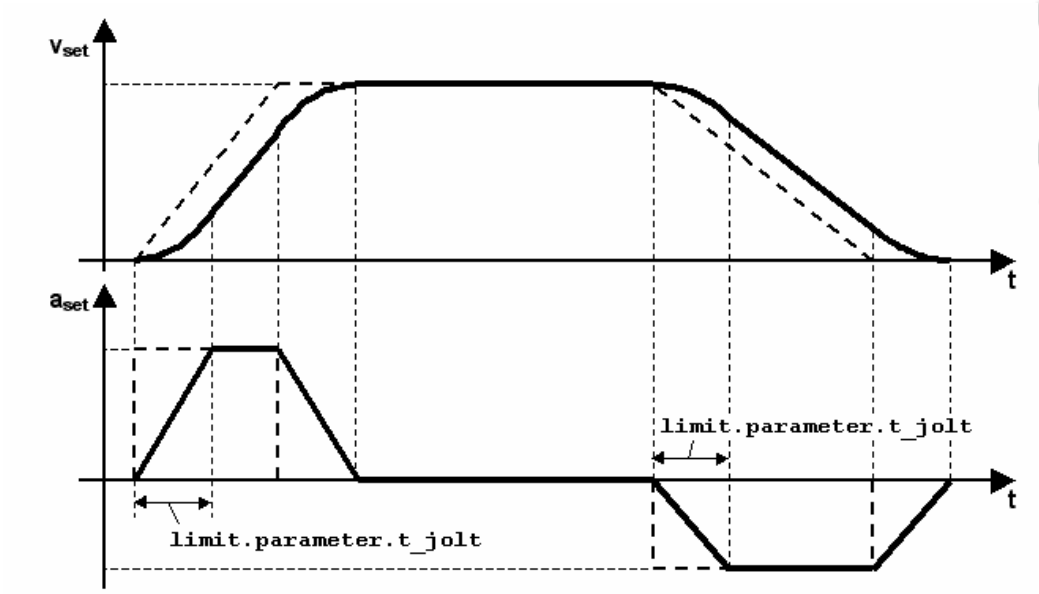


Fig. 14 Timing diagram of a movement with (bold line) and without (dotted line) jolt limitation

Note:

The "t_jolt" parameter is located in the limit values:

Element	Data Type	Description
...		
limit		Limit values
...		
parameter		Parameters
...		
t_jolt	REAL	Jolt time [sec]
...		

Fig. 15 Automation Studio online help system – Limit values

A value between 0.0 and 0.2 seconds can be defined for "t_jolt".

3.3 Predictive position controller

The reference variable of the position controller is created by the set value generator. The ACOPOS servo drive receives the control variable (current motor position) via the motor's encoder system and a corresponding encoder interface card. The control deviation is determined from these two variables, which results in a new manipulated variable for the lower-level speed controller.

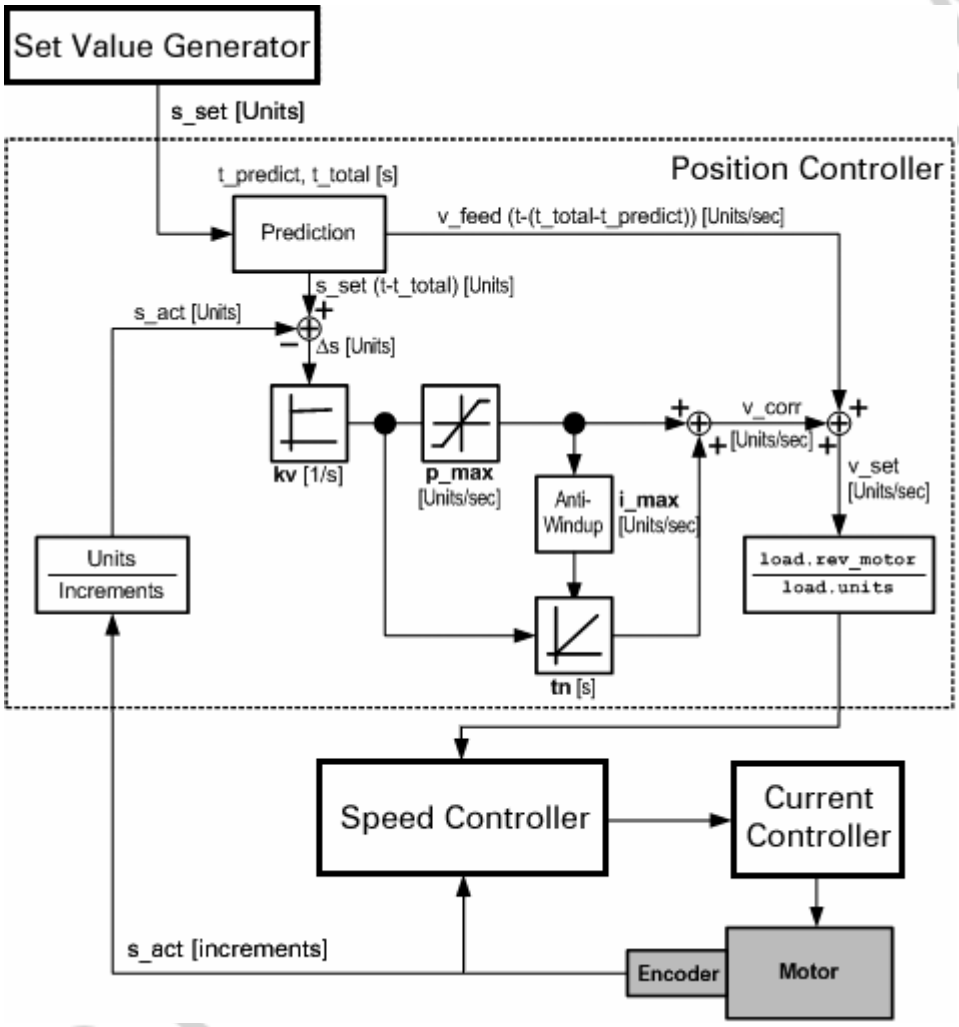


Fig. 16 Block diagram of the position controller

The position controller is implemented as PI controller with anti-windup (manipulated variable limitation) and "predictive" feed forward (input control).

The proportional element with the factor "**kv**" causes an immediate change in the set speed in the event that a lag error occurs. Changes in the set value or disturbance variables can cause these type of control deviations.

The I-element with integral action time "**tn**" is used to compensate for stationary disturbance quantities (e.g. suspended loads).

The manipulated variable limitation is implemented using the parameters "**p_max**" and "**i_max**". These values limit the maximum effect of the P-section and the I-section.

The feed forward is the predictive element of the position controller.

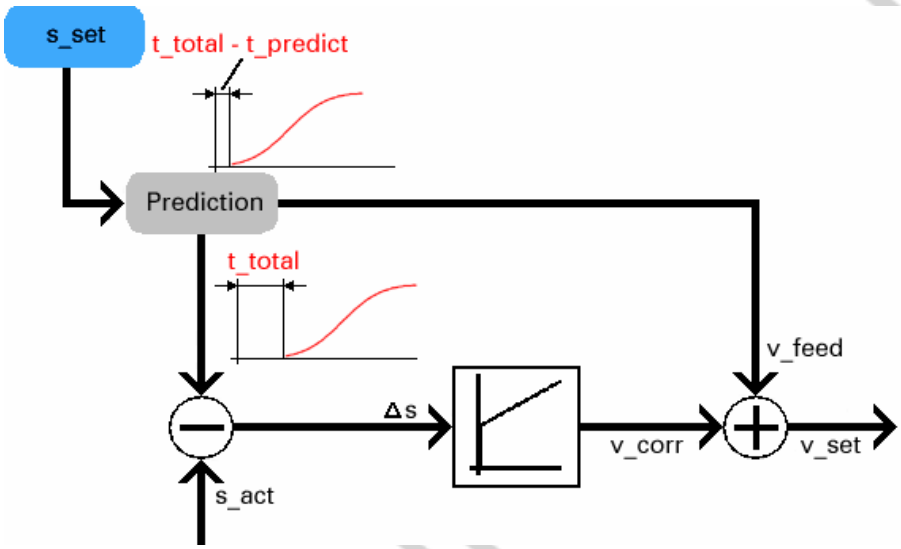


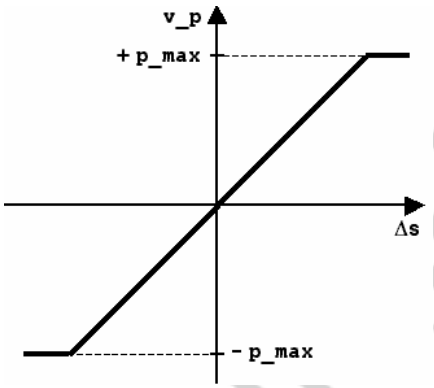
Fig. 17 Prediction, feed forward

A feed forward speed ("v_feed") results from a differentiation of the set position (s_set). The PI-controller establishes a correction speed ("v_corr") from the lag error (Δs). These two speeds are added together to produce the set speed (v_set) for the subsequent speed controller. The feed forward is then predictive if the set position is sent to the PI-controller with a delay (t_total). The set value should be delayed as long as the delay time of the controlled system. This is why the set speed (t_total – t_predict) is introduced to the speed controller first. If the corresponding set position of the PI-controller is then accepted, the control deviation is then smaller as a result of the already fed set speed value. This presents a load on the PI-controller because it still has to compensate for the remaining deviation. Without speed input control, the PI-controller would have to take care of the set speed by itself. This improves the reference behavior and the dynamic properties of the drive.

Calculation of "v_corr" which results from the lag error "Δs":

First, the speed "**v_p**" resulting from the proportional gain is calculated and limited to "**p_max**":

```
v_p = kv * Δs
if ( v_p > p_max )
    v_p = p_max
else if ( v_p < -p_max )
    v_p = -p_max
```



This value and "**i_max**" are used to calculate "**i_limit**" and the speed resulting from the integral gain "**v_i**" is limited to this value:

```
i_limit = i_max - |v_p|
if ( i_limit < 0 )
    i_limit = 0
v_i = f(v_i,Δs,kv,tn)
if ( v_i > i_limit )
    v_i = i_limit
else if ( v_i < -i_limit )
    v_i = -i_limit
```

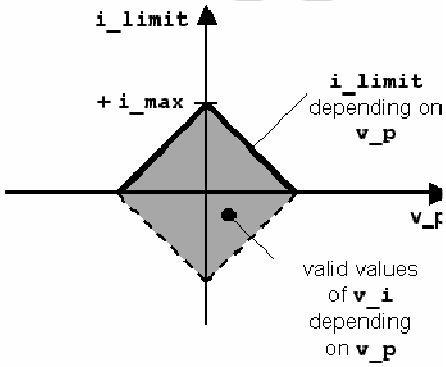


Fig. 18 Automation Studio online help, calculation of "v_corr"

Finally "**v_corr = v_p + v_i**" can be calculated.

The set speed is converted from the configurable measurement system (unit/sec) to the physical motor encoder system (rev./sec) before being passed onto the speed controller.

Lag error monitoring is also performed in the position controller. An E-Stop is executed if the control deviation exceeds a configurable threshold value.

3.4 Speed controller

3.4.1 Speed controller – General function

The speed controller's job is to determine the difference between the manipulated variable of the higher-level position controller and the measured speed. A manipulated variable for the lower-level current controller is then generated using a **PI-controller** with anti-windup.

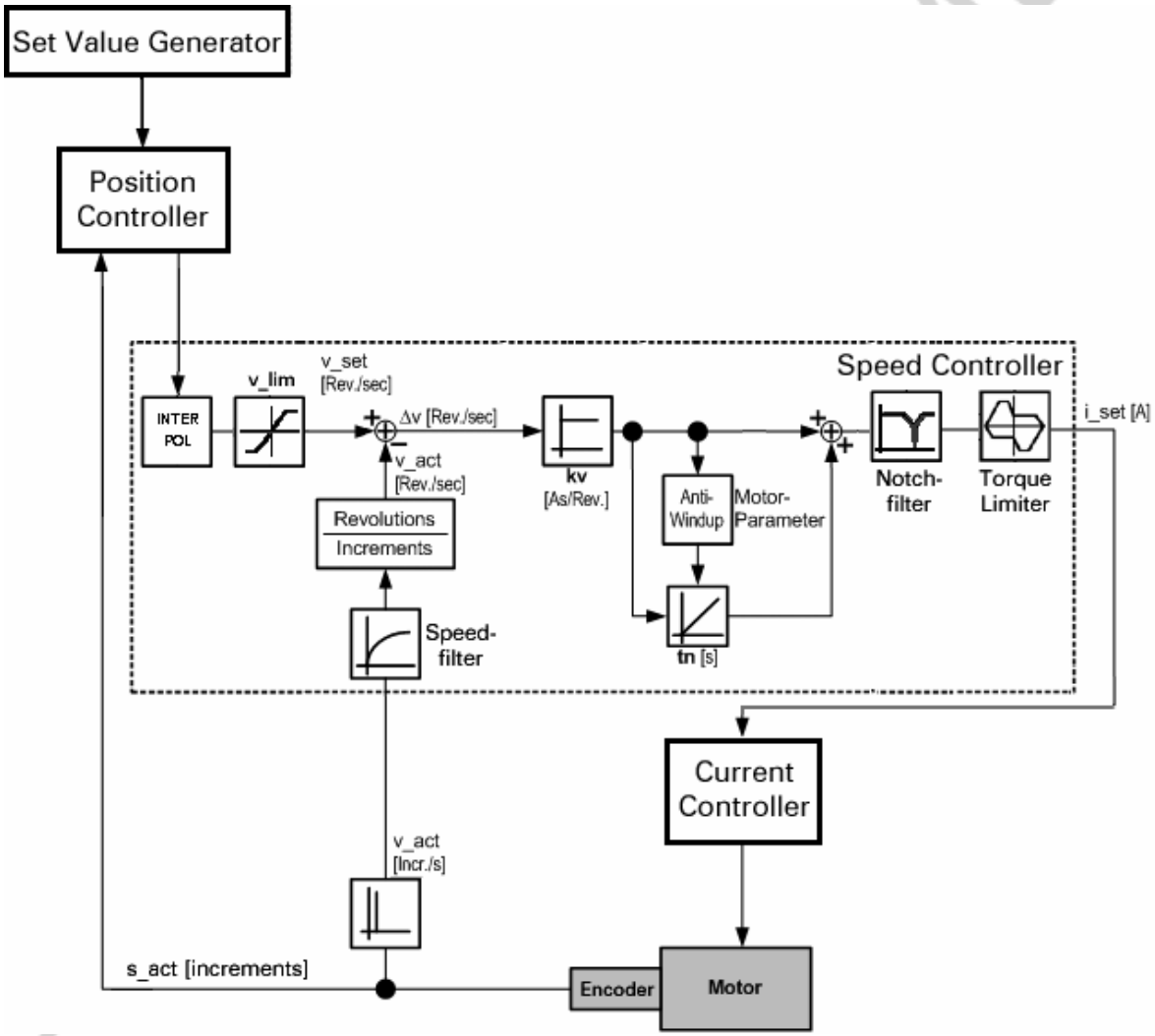


Fig. 19 Block diagram of the speed controller

The set speed value from the position controller is first fed through an interpolator. This is necessary because the position controller runs at a cycle time of 400µs, whereas the speed controller runs at **200µs**. The value is then limited using the maximum motor speed.

The actual speed is determined by differentiating the encoder position. The value is then sent through the **speed filter**. The value is converted from "incr./sec" to the unit "rev./sec" before it can be subtracted from the set speed.

We will take a more detailed look at the functionality of the speed filter a little later.

The P-element with the factor "**kv**" allows the controller to react immediately to any control deviation. This makes this element decisive for the dynamics of the speed controller.

The I-element with integral action time "**tn**" is used to compensate for stationary disturbance variables (e.g. load torque).

The output of the PI-controller can be filtered using a notch filter.

The function of the notch filter will be explained in a later section.

Before the set current can be provided as the reference variable for the current controller, the value is limited using a torque limiter. This torque limitation also determines the anti windup limits for the speed controller I-element.

3.4.2 Torque limiter

The torque limiter is mostly used to protect the motor and the ACOPOS servo drive from the following risks:

- The ACOPOS servo drive cannot output more current than the motor can handle (motor peak current).
- The motor's stator current cannot exceed the ACOPOS peak current.

By default, the torque limiter is initialized using the smaller of the following two values:

- Motor peak current
- ACOPOS peak current

3.4.3 Signal filter

Signal filters can be used on ACOPOS servo drives to separate high frequency disturbances (e.g. signal noise) from a desired signal and to suppress a resonance frequency.

Disturbance in the encoder signal can be caused by one or more of the following possibilities:

- Coupling of disturbances on the communication path (encoder cable).
- Quantization noise when converting the analog signal to a digital form. This mostly occurs when evaluating a resolver signal with low resolution.

With warp-resistant drive mechanics (e.g. direct load coupling to the motor shaft using fastening devices) the mechanical system can be subject to oscillations due to the closed control loop ("two-mass oscillation"). These types of systems generally have a resonance frequency in the range from 700 to 1500Hz. This is further dependent on the following factors:

- Rigidity of the mechanical system
- Mass inertia of the mechanical system
- Physical layout of the system.

Speed filter

As described in section 3.4, the actual speed is filtered, before being processed in the speed controller. This filter is a "speed filter" and functions like a low pass. High-frequency disturbances can be filtered out from the speed signal using this low pass behavior. This allows us to achieve higher controller quality.

Caution:

Parts of the desired signal will also be filtered out if the limit frequency of the low pass filter is set too low!

Notch filter

The frequency range of the set current, which causes the mechanical system to oscillate, can be filtered for the current controller.

Caution:

The notch filter should only be used on mechanics with a rigid coupling (e.g. direct drive). This filter should not be used for connections such as belts or gears!

Furthermore, it can only be used if the existing moments of inertia are always constant!

The resonance frequency of the system could shift as a result of mechanical wear. This means that over time the defined filter can lose its effectiveness.

The notch filter is only effective when the resonance frequency is in the range from 700 to 1500Hz.

The filter has the highest amount of damping at the "notch frequency" entered (= resonance frequency of the mechanical system). There is a range (band width) around this notch frequency in which the damping is lower than 3dB. The smaller the band width is set, the stronger the damping is in the notch frequency.

The notch filter can be used (once all of the requirements for use have been met) to increase the controller gain factors, without causing the entire system to become unstable.

3.5 Current controller

The current controller is made up of PI-controllers (like the position and speed controllers). The corresponding parameters are automatically determined by the servo drive using the motor parameters and the specific ACOPOS parameters. This means that we do not have to spend time in a later section going over how to set these values.

The current controller uses its manipulated variable to control the IGBTs (Insulated Gate Bipolar Transistor). These then output a pulse width modulated (PWM) current signal to the motor.

The current controller function at different cycle times depending on the PWM switching frequency:

PWM frequency	Display unit	Cycle time
5kHz	8V128M.00-2	200µs
10kHz	8V101x.yy-2 8V1090.00-2 8V1180.00-2 8V1320.00-2 8V1640.00-2	100µs
20kHz	8V1022.00-2 8V1045.00-2	50µs

4. THEORETICALLY DETERMINING THE CONTROL PARAMETERS

In the previous sections we learned how the ACOPOS servo drive controllers are structured and inter-related. Now we want to find the corresponding values for the control parameters. These values can be calculated or determined empirically if some of the requirements have not been met.

We can use the following formulas to determine good output values for the control parameters in the event that we don't already know the system's **total moment of inertia** and if the **load is fixed to the motor**. In most cases, we will achieve even better controller behavior by making fine adjustments to the parameter values manually.

4.1 Speed controller

Replacement time constant T_I of the current control loop:

$$T_I = 2 \times \left(0.000075 + \frac{1}{2 \times SwitchingFrequency} \right)$$

Summation of the individual time constants to a replacement time constant T_{σ_v}:

$$T_{\sigma\_v} = T_I + T_{dead\_v} + T_{filter}$$

T_{dead_v} = 0.000175 s (encoder interface dead time, speed determination und scanning)

T_{filter} ("t_filter" parameter) = Filter time constant of the speed filter

Proportional gain of the speed controller:

$$kv = \frac{J \times \sqrt{2} \times \pi}{T_{\sigma\_v} \times k_t}$$

J = total moment of inertia (J_{motor} + J_{brake} + J_{load})

k_t = Torque constant of the motor being used [Nm/A]

Integral action time of the speed controller:

$$tn = 4 \times T_{\sigma\_v}$$

4.2 Position controller

Summation of the individual time constants to a replacement time constant T_{σ_p}:

$$T_{\sigma\_p} = T_{interpol} + 4 \times T_{\sigma\_v} + T_{dead\_p}$$

$T_{interpol}$ = Dead time resulting from interpolator (0.0001s)

4 x T_{σ_v} = Replacement time constant of the speed control loop

T_{dead_p} = Dead time resulting from scanning (0.0002s)

Proportional gain of the position controller:

$$kv = \frac{1}{2 \times T_{\sigma\_p}}$$

Integral action time of the position controller:

$$tn = 4 \times T_{\sigma\_p}$$

Example:

Configuration of the 8MSA2S.E0 motor with ACOPOS 8V1010.00-2 without load:

k_t = 0.46 Nm/A

J = 0.06 kgcm²

Switching frequency = 10000 Hz

Speed controller:

$$T_I = 2 \times \left(0.000075 + \frac{1}{2 \times SwitchingFrequency} \right) =$$

$$2 \times \left(0.000075 + \frac{1}{2 \times 10000} \right) = \underline{0.00025 \text{ sec}}$$

$$T_{\sigma\_v} = T_I + T_{dead\_v} + T_{filter} = 0.00025 + 0.000175 + 0 = \underline{0.000425 \text{ sec}}$$

$$kv = \frac{J \times \sqrt{2} \times \pi}{T_{\sigma\_v} \times k_t} = \frac{0.000006 \times \sqrt{2} \times \pi}{0.000425 \times 0.46} = \underline{0.136 \; As / \text{Re}v.}$$

$$tn = 4 \times T_{\sigma\_v} = 4 \times 0.000425 = \underline{0.0017 \text{ sec}}$$

Position controller:

$$T_{\sigma\_p} = T_{\text{int}erpol} + 4 \times T_{\sigma\_v} + T_{dead\_p} = 0.0001 + 4 \times 0.000425 + 0.0002 = \underline{0.002 \text{ sec}}$$

$$kv = \frac{1}{2 \times T_{\sigma\_p}} = \frac{1}{2 \times 0.002} = \underline{250} \frac{1}{\text{sec}}$$

$$tn = 4 \times T_{\sigma\_p} = 4 \times 0.002 = \underline{0.008 \text{ sec}}$$

5. PROCEDURE FOR SETTING THE CONTROLLER

In this section we will learn about a possibility for determining control parameters, which has been proven time and time again in the field. This will allow you to check and make fine adjustments to values that have already been calculated. If this is not possible, suitable values can be determined empirically. In this case, the values specified in the corresponding notes can be used as start values.

5.1 General information

The control parameters only really have to be determined when the mechanics are already put together. When dealing with a machine where the axis is loaded with different masses, the parameters must be tested both without load and with the highest load. The parameters should also be tested at different speeds and accelerations.

These tests could result in the need for a compromise.

Globally valid control parameters cannot be used because all mechanics have different features.

To determine parameters from cascaded controllers, it is best to start from the bottom (last) controller and work up. In our case, this means that we would start by setting the speed controller first followed by the position controller. The current controller is automatically configured by the ACOPOS servo drive.

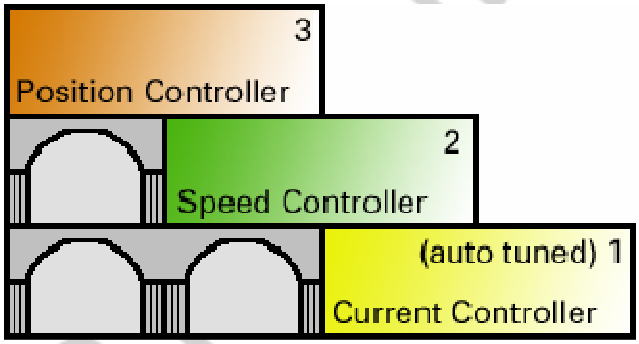


Fig. 20 Order for setting the controllers

It is usually the goal to set the controller as "tough" as possible. A controller can be considered "tough" when a disturbance variable is compensated for as quickly and perfectly as possible.

Example:

We want to manually rotate a flywheel mass which is mounted to the motor shaft.

- The controllers are set "soft" if the flywheel mass can be easily rotated.
- The controllers are set "hard" if the flywheel mass is difficult to rotate or cannot be rotated at all.

However, sometimes it is not the goal to set the controller as tough as possible. This is because a tough controller can cause quick heating, a higher load on the mechanics and therefore more wear. This is the reason why a compromise must often be found when determining the parameters.

We will be using the Motion Components test window to help us set the control parameters. This allows us to change and initialize the control parameters online. This also makes it possible to start positioning movements using any basis movement parameters. We can also configure, start and evaluate the trace in the Motion Components test window.

The behavior of the controller during a typical movement can be closely determined by selecting suitable parameters for tracing. Therefore, a basis movement (e.g. relative or absolute movement) is usually started and the respective parameters are recorded.

Furthermore, the control parameters should be selected so that oscillation of the recorded values is minimal.

Examples:

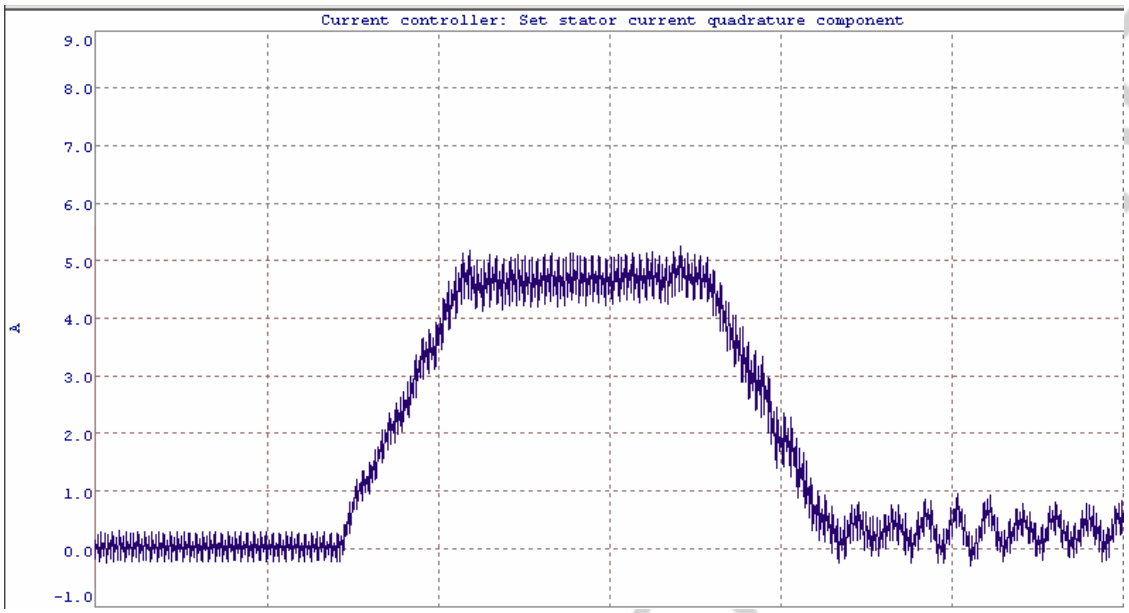


Fig. 21 Strong oscillations

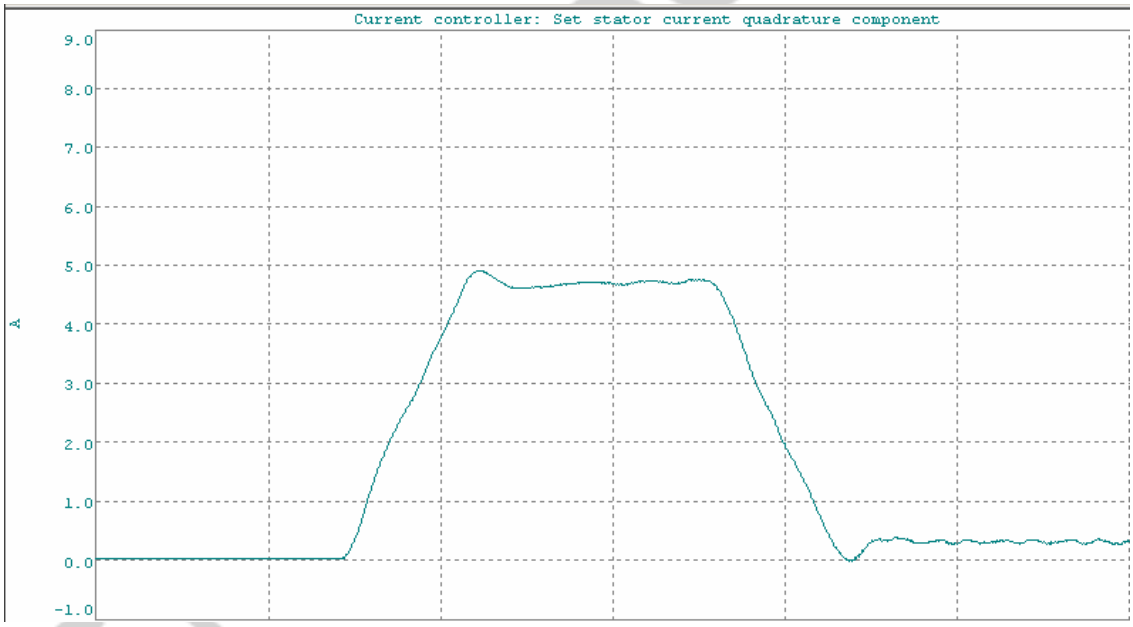


Fig. 22 Almost no oscillations

The user must then check to make sure that all of the requirements have been met (e.g. lag error within the tolerance) once all parameters have been set.

If this is the case, make sure that there are reserves for the gain factor because the system can behave differently due to mechanical wear. Therefore, a corresponding reserve (approx 1/3) should be taken from the determined values.

5.2 Speed controller

The following three parameters can be set for the speed controller. These are located in a subgroup of the control parameters:

Element	Data Type	Description
...		
controller		**Controller**
...		
speed		**Speed Controller**
kv	REAL	Proportional amplification [A sec / Revolutions]
tn	REAL	Integral action time [sec]
t_filter	REAL	Filter time constant [sec] (from V1.12 on)
...		

Fig. 23 Automation Studio online help system – Controller

Note:

The following parameters can be configured for the trace when setting the speed controller:

- Set speed: ACP10PAR_SCTRL_SPEED_REF (ID 250) [1/sec]
- Actual speed: ACP10PAR_SCTRL_SPEED_ACT (ID 251) [1/sec]
- Current controller: Set stator current of quadrature component [A]

The scan rate should be set as low as possible (approx. 0.2 to 4msec). A trigger event can be used to start the trace as accurately as possible (further information can be found in the Automation Studio Online help).

The "kv" and "tn" parameters of the position controller should be initialized with the value "0" so that only the speed controller is active

Note:

The value from "current controller: set stator current of the quadrature component" is the torque generating component of the set current.

The peak value of the current is displayed in the Motion Components trace. This value must be divided by the factor $\sqrt{2}$ ($\approx$1.414) to compare it with the specifications of the motor parameters and the ACOPOS™ servo drive parameters.

5.2.1 Proportional gain "kv"

The most important parameter of the three speed controller parameters is the gain factor "kv". This parameter significantly determines the dynamic properties of this controller. The goal is to set the value as large as possible without causing the system to oscillate.

Note:

1/5 to 1/10 of the nominal motor current (In) can be used as starting value for this factor.

Example:

Difference between the set and actual speed:	1 Rev./sec
Value of the factor "kv":	2 As/Rev.
Output value of the P-element:	2 A

5.2.2 Integral action time "tn"

For most applications it is not necessary to use an integral action time in the speed controller ("tn" = 0s). However, this time value should not be set too low if an application does require an I-section (e.g. to compensate for load-side disturbances, poor (soft) load coupling, high speed precision). Otherwise, the tendency for oscillation is increased in the speed controller.

Note:

100msec (0.1sec) can be used as starting value for the integral action time if you want to use the I-section.The value can then be gradually reduced.

5.2.3 Speed filter

The filter time constant "t_filter" for the speed filter is the last parameter in the speed control parameters. It can be used to set the limit frequency with the unit [sec] (e.g. 1kHz equals 0.001sec).

Improvement to the controller behavior using the speed filter can only be achieved in systems with a high mass moment of inertia and encoder systems with a low resolution (e.g. resolver). Whereas if encoder systems with a high resolution are used, the speed filter generally cannot create any improvements in the controller behavior.

Note:

You can start with a value of 0.8msec (0.0008sec). This value can be then be increased in small steps until the controller behavior (less oscillation) has improved. Generally, the usable values are in the range from 0.8msec to 2msec.

5.2.4 Notch filter

Note:

As we already know, the notch filter also works in the speed controller. However, there is no entry for this in the axis structure because it is not used too often. Therefore, it must be directly configured using parameter IDs:

Speed controller – notch filter: Frequency: ACP10PAR_FILTER_F0 (ID 226)
Speed controller – notch filter: Bandwidth: ACP10PAR_FILTER_B0 (ID227)

The resonance frequency of the system can be determined using the following steps:

- Initialize the position controller parameters with the value "0", deactivate the speed filter (t_filter = 0) and deactivate the notch filter (set both parameters to "0").
- Switch on the controller
- The proportional gain of the speed controller is increased until the mechanics oscillate.
- Record the actual speed using the trace (scan rate = 200µs). Switch the controller off again when the trace has finished.
- Analyze the frequency spectrum in the trace using FFT (Fast Fourier Transformation).
- Set the most notable frequency occurring in the trace as notch frequency for the notch filter.
- Enter a minimum band width (e.g. 25Hz).
- Switch the controller on again and determine a critical proportional gain for the speed controller.
- If the critical value has been increased, determine if the behavior has further improved due to the variation of the band width and the notch frequency. Otherwise, determine a characteristic natural frequency again.
- If further improvement is no longer possible, the determined values can be entered in an ACOPOS parameter table for example.

Note:

The notch filter cannot be used if a distinct resonance frequency cannot be determined.

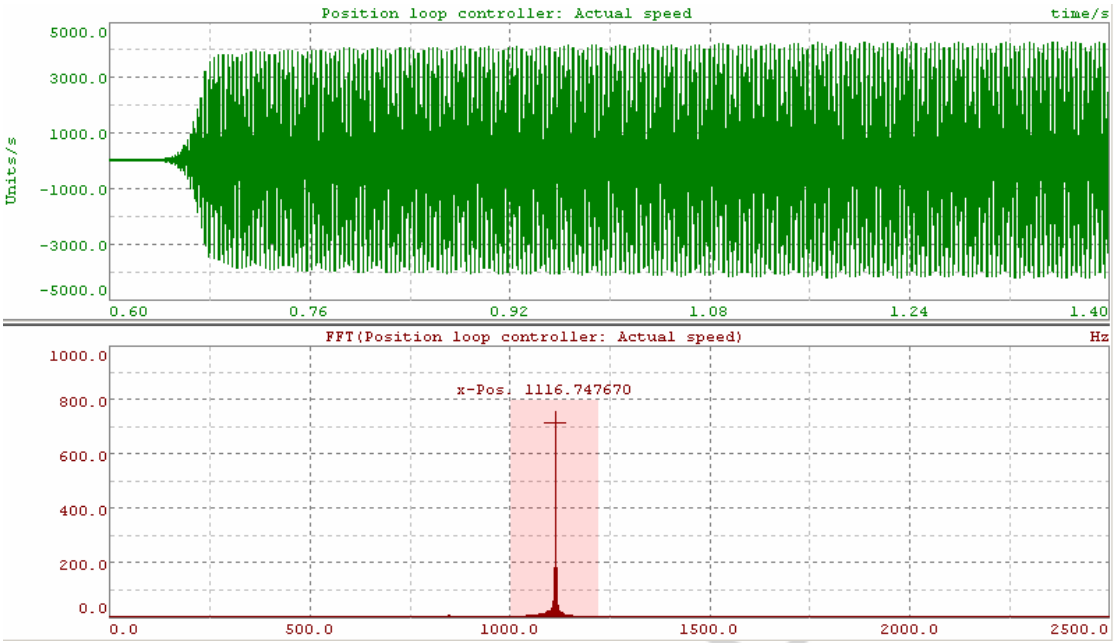


Fig. 24 FFT analysis.

Exercise:

Project: Name

Hardware: Test rack, flywheel mass mounted to the motor shaft.

Specifications:

The basis movement parameters to be used are already defined in the NC INIT parameter module.

Positioning path "s" = ± 50000 Units

A stationary manipulated variable is simulated in the example project using an additive set torque.

Parameters for the trace:

Max. trace duration: 2 seconds

Scan rate: 0.0008 seconds

Task:

Use the flow chart (Fig. 32) to set the speed controller in steps.

- What happens in this case if the value "0" is entered for the position control parameter "kv"?
- Can the notch filter be used and defined?
- Can the speed filter be used and defined?
- Would you define an integral action time "tn" for the speed controller?

5.3 Position controller

The parameters for setting the position controller are located in a subgroup of the control parameters:

Element	Data Type	Description
...		
controller		**Controller**
...		
position		**Position Controller**
kv	REAL	Proportional amplification [1/sec]
tn	REAL	Integral action time [sec]
t_predict	REAL	Prediction time [sec]
t_total	REAL	Total delay time [s]
p_max	REAL	Max. proportional action [Units/sec]
i_max	REAL	Max. integral action [Units/sec]
...		

Fig. 25 Automation Studio online help system – Controller

Note:

The following parameters can be configured for the trace when setting the position controller:

- Position controller: Actual speed [Units/sec]
- Position controller: Lag error [Units]
- Current controller: Set stator current of quadrature component [A]

The scan rate should be set as low as possible in this case as well.

5.3.1 Proportional gain "kv"

The value from the "kv" factor should also be set as large as possible for the position controller without causing the controller to oscillate.

Note:

You can start with a value of 50 /sec and increase it gradually.

Example:

Lag error:	15 units
Value of the factor "kv":	100/sec
Output value of the P-element:	1500 Units/sec

5.3.2 Integral action time "tn"

The similar conditions also apply for this value as for the integral action time of the speed controller. For some applications it is sufficient to just use one true P-controller. An I-section must be used if an axis has to compensate for a stationary manipulated variable (e.g. hanging load). Furthermore, it may be necessary to use the I-section of the position controller if the speed controller was only able to be set soft.

The integral action time does not have to be set for the position controller if already set for the speed controller.

Note:

You can also use a starting value of 100msec if necessary in this case.

An I-section in the position controller causes an overshoot when the target position is reached and therefore should only be used in special cases.

5.3.3 Total delay time "t_total"

In a single-axis application, this parameter should be initialized with the same value as the prediction time.

The delay via the network can be compensated using the "t_total" in multi-axis applications.

The potential value range is also as large as in the parameter "t_predict".

5.3.4 Prediction time "t_predict"

This parameter is required for the feed forward. A value "t_predict = 0s" disables the offset.

The prediction time makes it possible to compensate for the lag error during the acceleration and deceleration phase to nearly "0". This parameter is generally set after correct values have been determined for "kv" and "tn".

Note:

The potential value range of the "t_predict" parameter is 0.0 to 0.06 seconds.

Generally, a large prediction time is not really necessary if the speed controller could be set hard.

Definition rule:

$$t\_predict = \frac{4 \times \pi \times J}{(kv_{SpeedController} \times k_t)} + 0.0002$$

J = Moment of inertia on the motor [kgm²]
$kv_{speed\ controller}$ = Proportional gain of the speed controller [As/Rev.]
k_t = Torque constant of the motor being used [Nm/A]

Note:

The position controller runs at a cycle time of **400µs**. This is why the values for "t_predict" and "t_total" should always be selected in a way so that they are equal to a multiple of 0.0004 seconds. Otherwise, they are rounded off by the ACOPOS™ servo drive.

The following charts display the lag error for different prediction time values.

Corresponding speed profile:

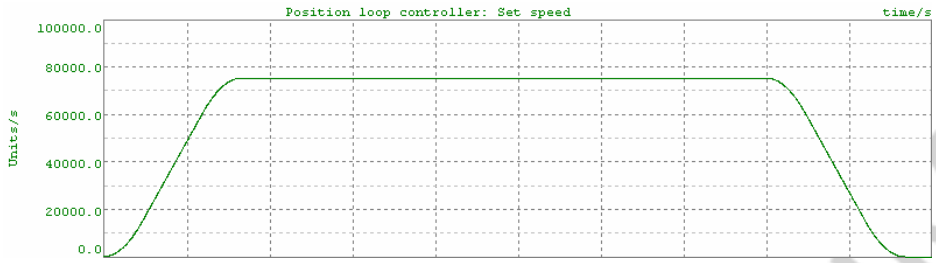


Fig. 26 Corresponding speed curve for the following lag error curves

"t_predict" set too low:

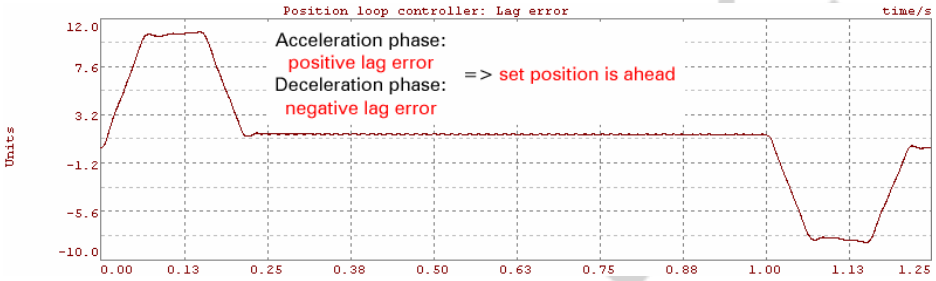


Fig. 27 Prediction time set too low

"t_predict" set too high:

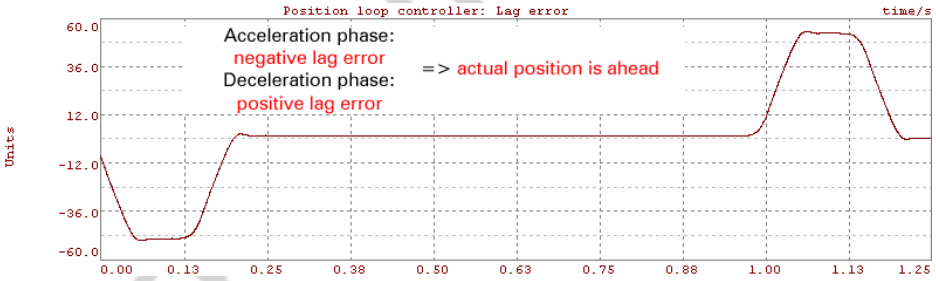


Fig. 28 Prediction time set too high

"t_predict" set correctly:

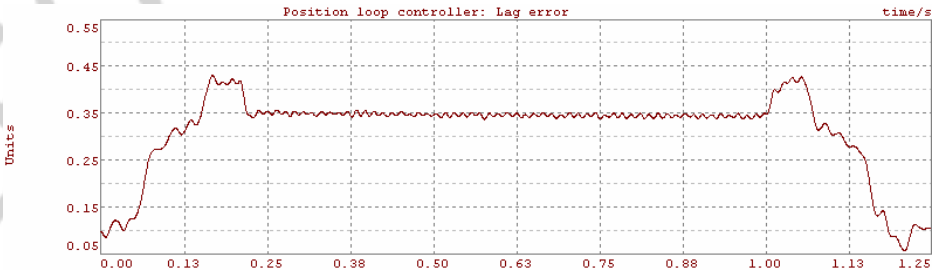


Fig. 29 Prediction time set correctly

5.3.5 Maximum proportional action "p_max"

The influence of the proportional gain can be limited using the "p_max" parameter. This can be done to prevent manipulated variables that are too large.

Note:

The value for these parameters can be calculated using the following formulas:

$$p\_max = \frac{I_{max} \times 2}{kv_{SpeedController}} \times UnitFactor$$

I_{max} = Motor peak current [A]
$kv_{speed\ controller}$ = Proportional gain of the speed controller [As/Rev.]
Unit factor = Unit scaling [units/rev.]

5.3.6 Maximum integral action "i_max"

The maximum influence of the integral section can be limited using the "i_max" parameter. This can be done to prevent a "windup".

Note:

The value for these parameters can be calculated using the following formula to achieve a required holding torque:

$$i\_max = \frac{\frac{M}{k_t} \times 1.1}{kv_{SpeedController}} \times UnitFactor$$

M = Required holding torque [Nm]
k_t = Torque constant of the motor being used [Nm/A]
$kv_{speed\ controller}$ = Proportional gain of the speed controller [As/Rev.]
Unit factor = Unit scaling [units/rev.]

Exercise:

Same project and hardware as before.

Specifications:

The basis movement parameters to be used are already defined in the NC INIT parameter module.

Positioning path "s" = ± 50000 Units

A stationary manipulated variable is simulated in the example project using an additive set torque.

Parameters for the trace:

Max. trace duration: 2 seconds

Scan rate: 0.0008 seconds

Task:

Use the flow chart (Fig. 32) to set the position controller in increments. This setting will be based on the values determined for the speed controller in the previous exercise.

- What happens if you are not using an integral action time "tn" for the position controller?

5.4 Limit parameter

When the control parameters are optimized, two parameters must still be set for the limit values.

5.4.1 Jolt filter time "t_jolt"

The value for the parameter can be determined by recording the lag errors during a positioning movement without jolt filter time. At the end of this movement, it will become evident that the system must first "settle down". After being determined from the trace, the settling time (time until the oscillations level out) can now be used as jolt filter time "t_jolt".

Fig. 30 Determining the jolt time

5.4.2 Lag error stop limit "ds_stop"

Element	Data Type	Description
...		
limit		Limit values
...		
parameter		Parameters
...		
ds_warning	REAL	If the lag error exceeds "ds_warning", a warning will be indicated
ds_stop	REAL	If the lag error exceeds "ds_stop", an active movement will be stopped
...		

Fig. 31 Automation Studio online help system – Limit values

A warning is output if the current lag error value set for the "ds_warning" parameter is exceeded. An emergency stop is executed if the value of the "ds_stop" parameter is also exceeded.

Note:

A controlled e-stop ramp is generated when an e-stop occurs. The set speed is decelerated to "0" using the current initialized limit values.The controller is then switched off.

Note:

The value for "ds_stop" can be determined using the following calculation:

$$\frac{I_{max}}{kv_{SpeedController} \times kv_{PositionController}} \times 2 \times UnitFactor$$

I_{max} = Motor peak current [A]
$kv_{speed\ controller}$ = Proportional gain of the speed controller [As/rev.]
$kv_{position\ controller}$ = Proportional gain of the position controller [1/sec]
Unit factor = Unit scaling [units/rev.]

Task:

Same project and hardware as before.

Specifications:

The basis movement parameters to be used are already defined in the NC INIT parameter module.

Positioning path "s" = ± 50000 Units

A stationary manipulated variable is simulated in the example project using an additive set torque.

Parameters for the trace:

Max. trace duration: 2 seconds

Scan rate: 0.0008 seconds

Task:

Set the limit value correctly and check the result.

5.5 Overview

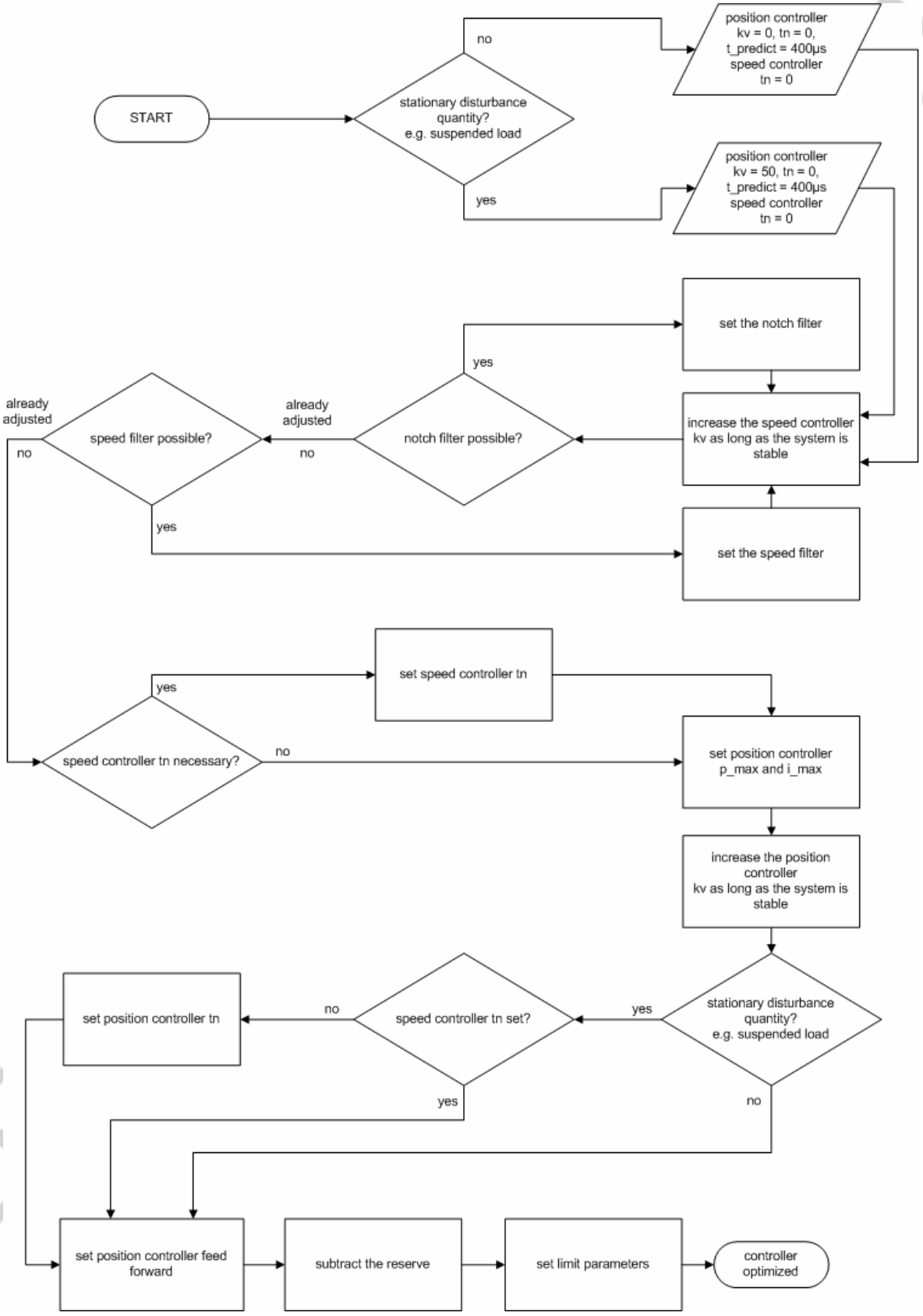


Fig. 32 Overview of the process for setting the control parameters

6. SAVING THE CONTROLLER SETTINGS

The data just determined must be saved to the controller so that it is available to the ACOPOS servo drive each time the machine is started. All parameters available in the axis structure can be saved in an NC INIT parameter module. The remaining values can be entered in a parameter table.

6.1 NC INIT parameter module

The following options are available for saving the values in the NC INIT parameter module.

- Entering the values directly in the corresponding module.

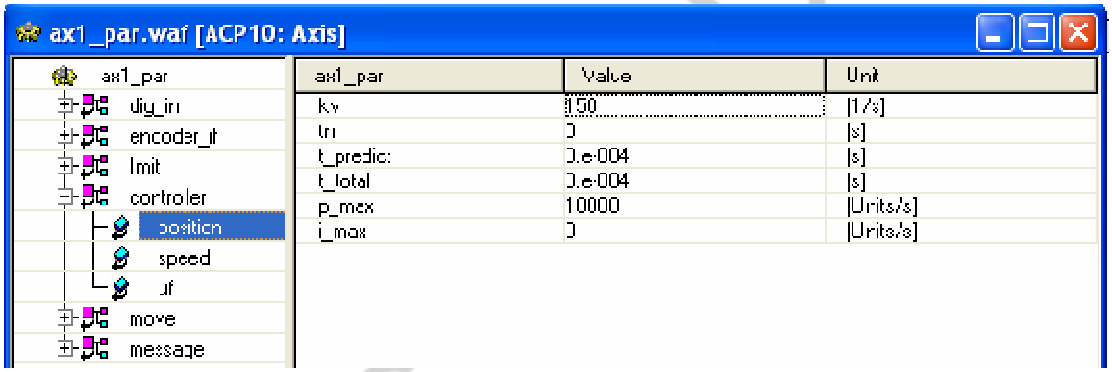


Fig. 33 NC INIT parameter module

- Saving the determined values immediately in the NC test window.

Once everything has been saved, the project should be compiled and the current version should be transferred to the controller.

6.2 ACOPOS parameter table

The parameters and values (e.g. for the notch filter) can be entered in an ACOPOS parameter table.

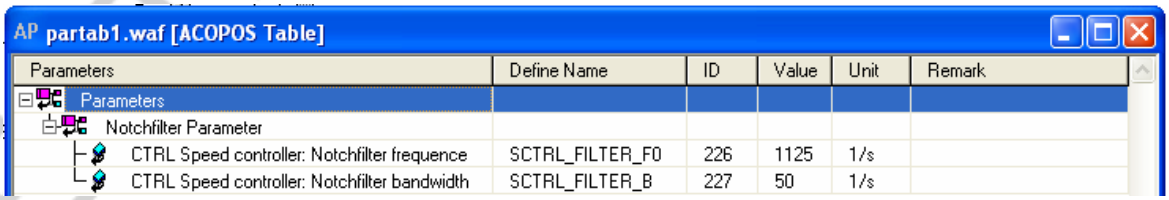


Fig. 34 ACOPOS parameter table

7. SUMMARY

By understanding how the parameters that we will be using work, we are now able to determine correct values.

The cascaded control concept allows us to optimize the control parameters and filter parameters step-by-step.

The parameters determined are usually different. However, the procedure for obtaining these values is always the same.

8. APPENDIX

k_t = [redacted] Nm/A

J = [redacted] kgm²

Switching frequency = [redacted] Hz

Speed controller:

$$= 2 \times \left(0.000075 + \frac{1}{2 \times SwitchingFrequency} \right) = 2 \times \left(0.000075 + \frac{1}{2 \times [redacted]} \right) = [redacted]$$

$$T_{\sigma\_v} = T_I + T_{dead\_v} + T_{filter} = [redacted] + 0.000175 + [redacted] = [redacted]$$

$$kv = \frac{J \times \sqrt{2} \times \pi}{T_{\sigma\_v} \times k_t} = \frac{[redacted] \times \sqrt{2} \times \pi}{[redacted] \times [redacted]} = [redacted]$$

$$tn = 4 \times T_{\sigma\_v} = 4 \times [redacted] = [redacted]$$

Position controller:

$$T_{\sigma\_p} = T_{interpol} + 4 \times T_{\sigma\_v} + T_{dead\_p} = 0.0001 + 4 \times [redacted] + 0.0002 = [redacted]$$

$$kv = \frac{1}{2 \times T_{\sigma\_p}} = \frac{1}{2 \times [redacted]} = [redacted]$$

$$tn = 4 \times T_{\sigma\_p} = 4 \times [redacted] = [redacted]$$

Notes

Overview of training modules

TM200 – B&R Company Presentation **
TM201 – B&R Product Spectrum **
TM210 – The Basics of Automation Studio
TM211 – Automation Studio Online Communication
TM212 – Automation Target **
TM213 – Automation Runtime
TM220 – The Service Technician on the Job
TM223 – Automation Studio Diagnostics
TM230 – Structured Software Generation
TM240 – Ladder Diagram (LAD)
TM241 – Function Block Diagram (FBD)
TM246 – Structured Text (ST)
TM247 – Automation Basic (AB)
TM248 – ANSI C
TM250 – Memory Management and Data Storage
TM260 – Automation Studio Libraries I
TM261 – Closed Loop Control with LOOPCONR

TM400 – The Basics of Motion Control
TM410 – The Basics of ASiM
TM440 – ASiM Basic Functions
TM441 – ASiM Multi-Axis Functions
TM445 – ACOPOS ACP10 Software
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Starting up Motors

TM500 – The Basics of Integrated Safety Technology
TM510 – ASiST SafeDESIGNER

TM600 – The Basics of Visualization
TM610 – The Basics of ASiV
TM630 – Visualization Programming Guide
TM640 – ASiV Alarm System
TM650 – ASiV Internationalization
TM660 – ASiV Remote
TM670 – ASiV Advanced

TM700 – Automation Net PVI
TM710 – PVI Communication
TM711 – PVI DLL Programming
TM712 – PVIServices
TM730 – PVI OPC

TM800 – APROL System Concept
TM810 – APROL Setup, Configuration and Recovery
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM840 – APROL Parameter Management and Recipes
TM850 – APROL Controller Configuration and INA
TM860 – APROL Library Engineering
TM865 – APROL Library Guide Book
TM870 – APROL Python Programming
TM890 – The Basics of LINUX

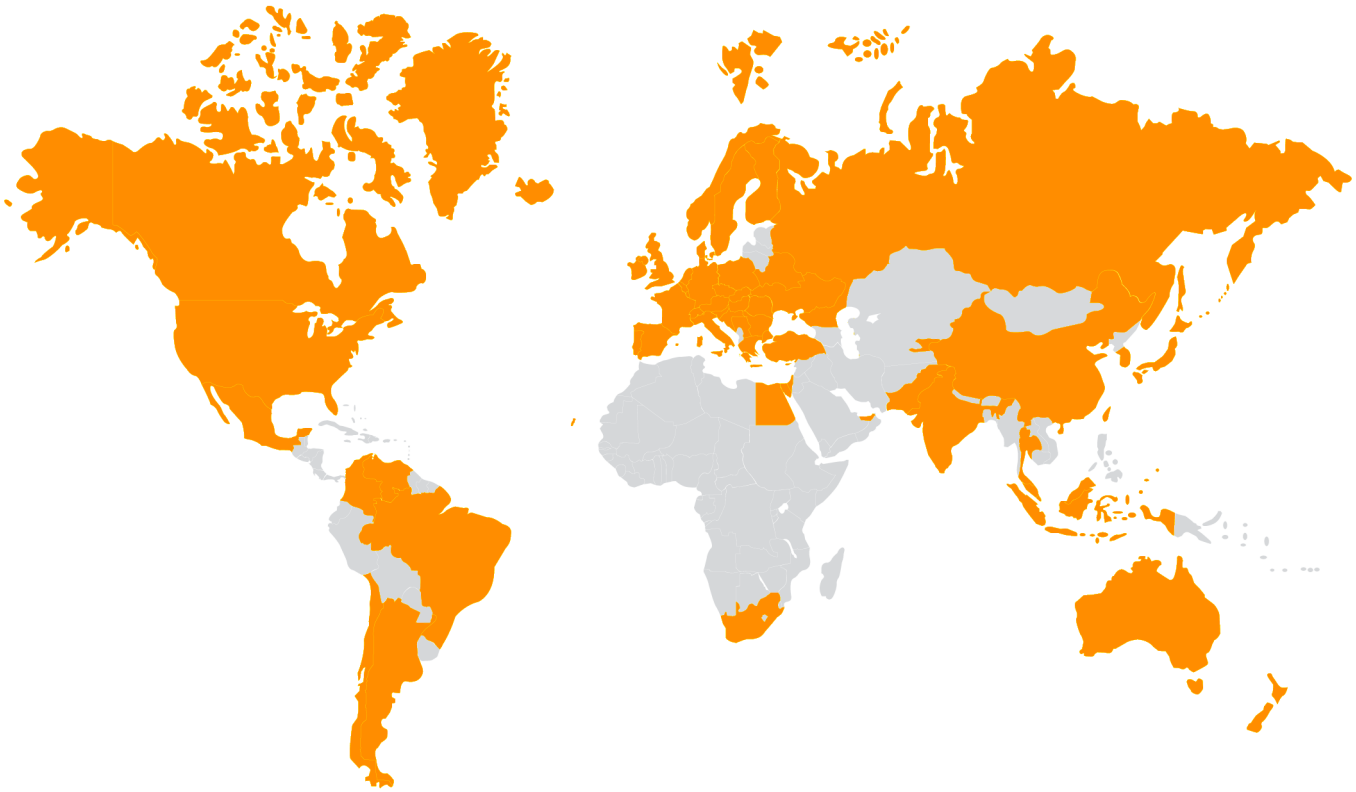
**) see Product Catalog

CORPORATE HEADQUARTERS

Bernecker + Rainer Industrie-Elektronik Ges.m.b.H.
B&R Strasse 1
5142 Eggelsberg
Austria
Tel.: +43 (0) 77 48 / 65 86 - 0
Fax: +43 (0) 77 48 / 65 86 - 26
info@br-automation.com
www.br-automation.com

TM450TRE.25-ENG 0907

140 offices in more than 55 countries - **www.br-automation.com/contact**



Australia • Argentina • Austria • Belarus • Belgium • Brazil • Bulgaria • Canada • Chile • China • Colombia • Croatia • Cyprus
Czech Republic • Denmark • Egypt • Emirates • Finland • France • Germany • Greece • Hungary • India • Indonesia
Ireland • Israel • Italy • Japan • Korea • Luxemburg • Kyrgyzstan • Malaysia • Mexico • The Netherlands • New Zealand
Norway • Pakistan • Poland • Portugal • Romania • Russia • Serbia • Singapore • Slovakia • Slovenia • South Africa
Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • USA • Venezuela • Vietnam