

Servo Drive Setup Guide

TSD and TSP Series

This user guide describes how to set up the *TSD and TSP* servo drive series.

Table of Contents

Abbreviations	2	3.10 Global Keys	30
1 Introduction	3	4 Plug-In Modules	31
2 Hardware and Software Setup	4	4.1 Axis Module	31
2.1 Hardware Prerequisites	4	4.2 Frequency Response Module	33
2.2 Software Installation	4	5 Drive Configuration	42
2.3 Connect Drive to PC	6	5.1 Preparation	42
3 How to use TAM System Explorer	7	5.2 Initial Setup of the Register Tree	44
3.1 Topology Tree	8	5.3 Verification of the Encoder	56
3.2 Axis Monitor	11	5.4 Controller Tuning	56
3.3 Scope	13	5.5 Verification of the Setup	72
3.4 Triamec Workspace	21	6 Advanced Topics	77
3.5 TAM Configuration	22	6.1 Commutation with Absolute Encoders	77
3.6 TAM Snapshot	26	7 Flow Charts	81
3.7 Firmware Update	28	References	83
3.8 Tama Program	29	Revision History	84
3.9 Setup Protection	30		

Abbreviations

AC	Alternating Current
DC	Direct Current
HMI	Human-Machine Interface
LED	Light Emitting Diode
PC	Personal Computer
STO	Safe Torque Off
TAM	Triamec Advanced Motion
TAM API	TAM Application Programming Interface

1 Introduction

This user guide describes how to set up the *TSD* and *TSP* servo drive series.

After a short description of the required hardware and software setup in chapter 2, the user guide explains the usage of the *TAM System Explorer* software in chapter 3 and 4. **Chapter 5 describes the commissioning and tuning of a servo axis.** Chapter 6 handles several advanced topics.

To control Triamec drives with *Beckhoff TwinCAT*, first setup the drives as described in this document. Then proceed to document [2] for a *Tria-Link* setup, or [3] for an *EtherCAT* setup.

Names in italics, like *Triamec*, are general keywords. Register names and paths are indicated according to the following example: Axes[0].Parameter.PositionController.OutputLimit. Bold expressions indicate a HMI item, i.e. **File > Open**. Keyboard keys are formatted like Enter.

This version of the user guide requires the use of a TAM System Explorer version ≥ 7.29 and a drive firmware version $\geq 4.24.0$

2 Hardware and Software Setup

This chapter describes the hardware prerequisites and the installation of the required software used for commissioning of the drive.

2.1 Hardware Prerequisites

For the commissioning of the Triamec drive with this user guide the following hardware setup is required.

- A motor with an encoder connected to the drive.
- The DC-bus is supplied with an appropriate DC voltage using a Triamec power supply.
- The drive logic is supplied with 24VDC.
- Make sure the STO channels are connected correctly to fulfill the safety requirements.
- Connect the drive with the PC (see section 2.3 for further information).

See also the corresponding hardware manual for further information:

- www.triamec.com > **Support** > **Documents** > **Manuals**
- or **Help** > **Documentation** > **Hardware** in *TAM System Explorer*.

2.2 Software Installation

For the commissioning of *Triamec* drives *TAM System Explorer* is required. *TAM System Explorer* provides access to the drive parameters, commands and signals. Additional tools are available for measuring signals, identifying the dynamic behavior and tuning the controllers. *TAM System Explorer* is an application based on *TAM API* and is installed as part of the *TAM Software* package. Administrator rights are required during the installation process. The following steps describe how to run the installation:

1. Download the current release of *TAM Software* from www.triamec.com > **Support** > **TAM Software**.
2. Extract the downloaded ZIP file and run the installer `TAM Software *.* Setup.exe`.
3. The **TAM Software Setup** dialog is opened (Figure 1). Click **Install** to start the installation process.
4. If a **User Account Control** dialog is opened check if the verified publisher is correctly identified as

Triamec Motion AG and acknowledge the installation process.

5. The setup process will also install the required drivers. Additional prompts might appear during this process which need to be approved.
6. In some cases a restart of the *PC* is required to complete the installation process. In this case, a prompt will pop up with the request to restart the *PC*.

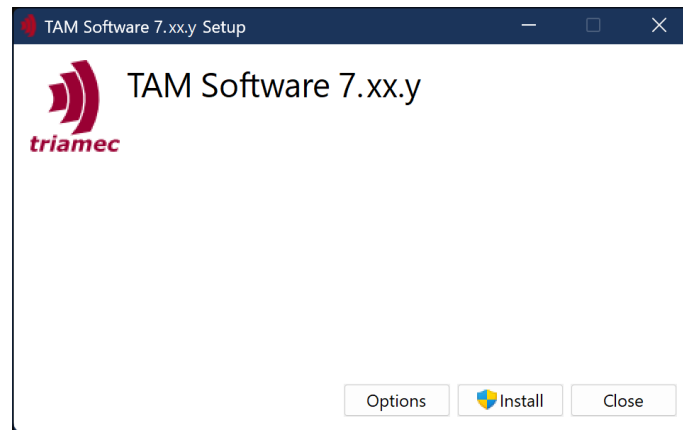


Figure 1: TAM Software Setup

2.2.1 Deinstallation

Deinstall the *TAM Software* by means of the operating system's controls.

In some side-by-side installation scenarios, uninstalling the *TAM Software* might fail. In such a case, first *repair* the software, then *uninstall*. Repair is accessible via the uninstall routine, as shown in Figure 2.

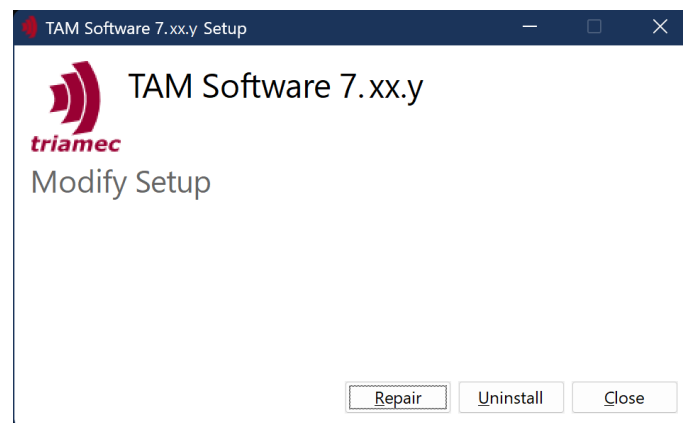


Figure 2: Deinstall the TAM Software

Note Uninstalling the *TAM Software* might also remove the associated device drivers. In this case all programs accessing Triamec devices may fail.

2.3 Connect Drive to PC

For the commissioning of the drive, the following options are available to connect the drive with the PC:

2.3.1 USB

- Connect the drive to the PC with a *USB* cable. Multiple drives can be connected together to the PC by using a *USB* hub.
- An existing *Tria-Link* network can be connected with an (external) *PC* via *USB* by connecting the *PC* to the adapter card as described in chapter Error: Reference source not found.

In case a Windows Explorer window opens when the drive is rebooted or when USB is connected to the PC, this can be suppressed by changing the following Windows settings:

- **Windows Start Button – Settings** (gear wheel) – **Auto Play**
- Under **Removable drive** select **Take no action** from the pull-down menu.

2.3.2 Ethernet

Up to 4 drives may be directly connected to a PC over Ethernet. Leave the IP configuration of the drive on the default values (DHCP). For more connection types with Ethernet see document [4].

2.3.3 Tria-Link

Please refer to document [12] for more information about how to connect a PC to a *Tria-Link* system.

3 How to use TAM System Explorer

This chapter gives an introduction into the usage of *TAM System Explorer*.

If *TAM Software* is installed correctly *TAM System Explorer* can be executed from the **Windows Start Menu**. Figure 3 shows the window after the start, containing the following panels:

1. *Topology Tree*: The topology tree allows to navigate between different devices and registers.
2. *Axis Monitor*: The axis monitor shows the state for each axis and allows to acknowledge pending errors. Additionally, emergency buttons are provided.
3. *Scope Area*: The scope allows to plot the values of all registers provided by the topology tree.
4. *Tab Panel*: The tab panels are used to write and read to registers from the *Topology Tree*, to set the scope properties and to log messages.

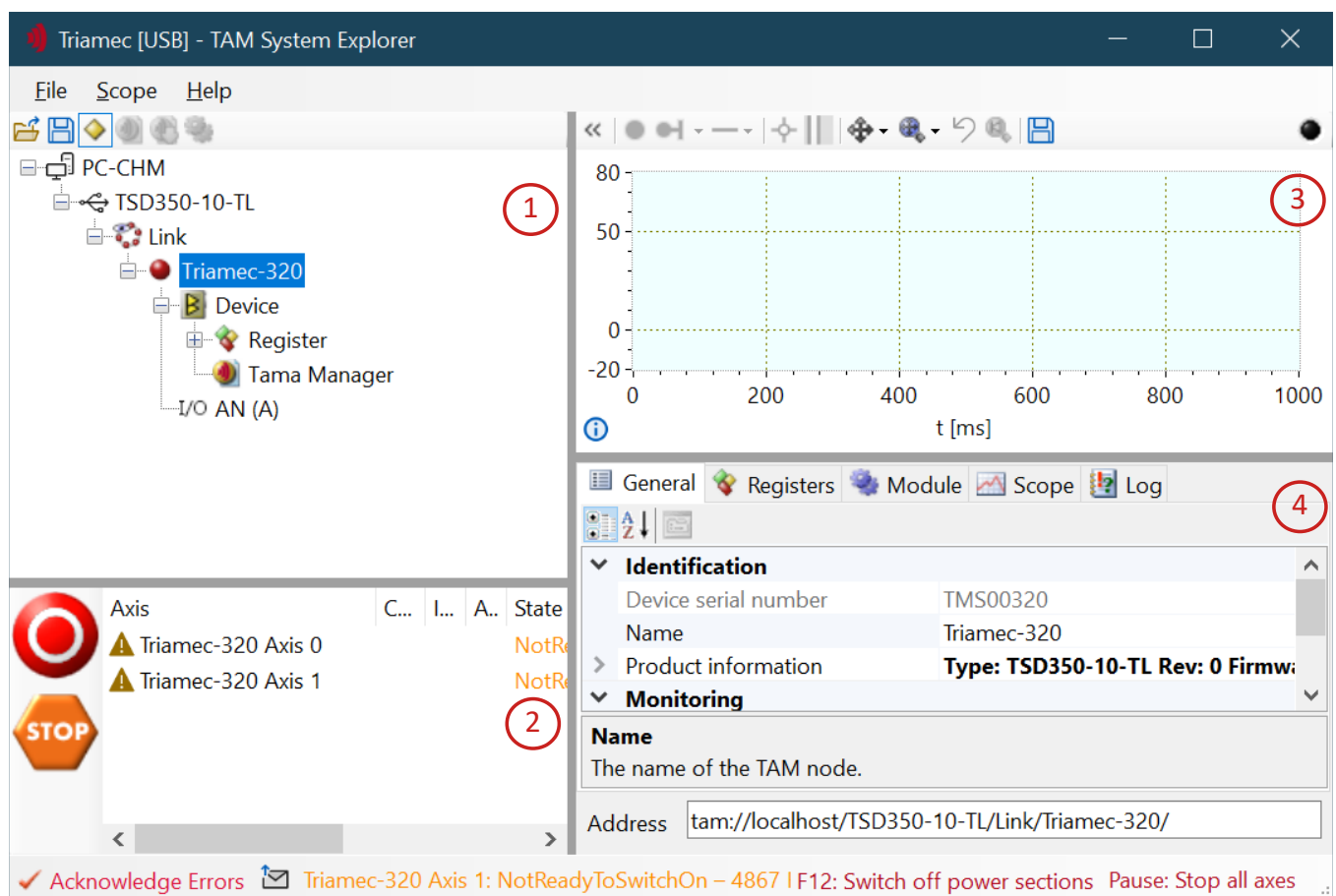

















Figure 3: *TAM System Explorer* after start up. 1: *Topology Tree*; 2: *Axis Monitor*; 3: *Scope Area*; 4: *Tab Panel*


3.1 Topology Tree

After an initializing phase, the *Topology Tree* shows the actual hardware content of the system. The *Topology Tree* is built out of the following objects:

	Local Computer	Computer on which the <i>TAM System Explorer</i> is running
	Tria-Link Adapter Card	<i>PCIe</i> adapter card used to communicate over <i>Tria-Link</i> .
	USB Adapter	Device connected via <i>USB</i> . Drives and <i>TL</i> cards are equipped with a <i>USB</i> connector.
	Ethernet Adapter	Used to communicate over <i>Ethernet</i>
	Offline Adapter	Examine a system while not being connected
	Tria-Link	<i>Tria-Link</i> communication channel used to connect the adapter with devices
	Tria-Link Observer	<i>Tria-Link</i> communication channel in observer mode
	Station	Addressable node within the link
	Device	Logical unit providing a register tree, for example a servo drive
	Register Node	Collection of registers with similar purpose (array of similar structures)
	Register	Representing a parameter, a command, a signal or information
	Register Array	An array of registers of the same type
	Tama Manager	Allows to assign, download and enable <i>Tama</i> programs
	Module	User interfaces which simplify the register access e.g. for the tuning of the axes.
	Option Module	Option module for additional input and output.


Note If a connected device is not listed in the *Topology Tree*, execute a restart (**File > Restart**).

It is recommended to give the stations a meaningful name e.g. *Station XY*, *Station Z*. The naming of the station can be changed by selecting the station and then clicking the station name or pressing **F2**. The station name is stored in register General.Parameters.DeviceName in the *Topology Tree*. The hardware device related to the station can be identified either

- by the serial number which is displayed in register General.Signals.Identifications.SerialNumber.
- or by activating the status *LED* (red blinking) of the drive by setting General.Commands.ExternalError to TRUE. To reset the status *LED* set ExternalError to FALSE and press the  **Acknowledge Errors** button.

Each device in the *Topology Tree* provides a *Register Tree*. The *Register Tree* is the basic interface between *TAM System Explorer* and the device. The registers are divided into three or four classes:

- Parameters
- Commands
- Signals
- Information (*Axes only*)

In order to interact with a register unfold the *Topology Tree* until the desired register gets visible. The registers may be hidden, to reduce vertical amount of space occupied by the tree. In order to show them, click on the **Show/Hide individual registers** Button .

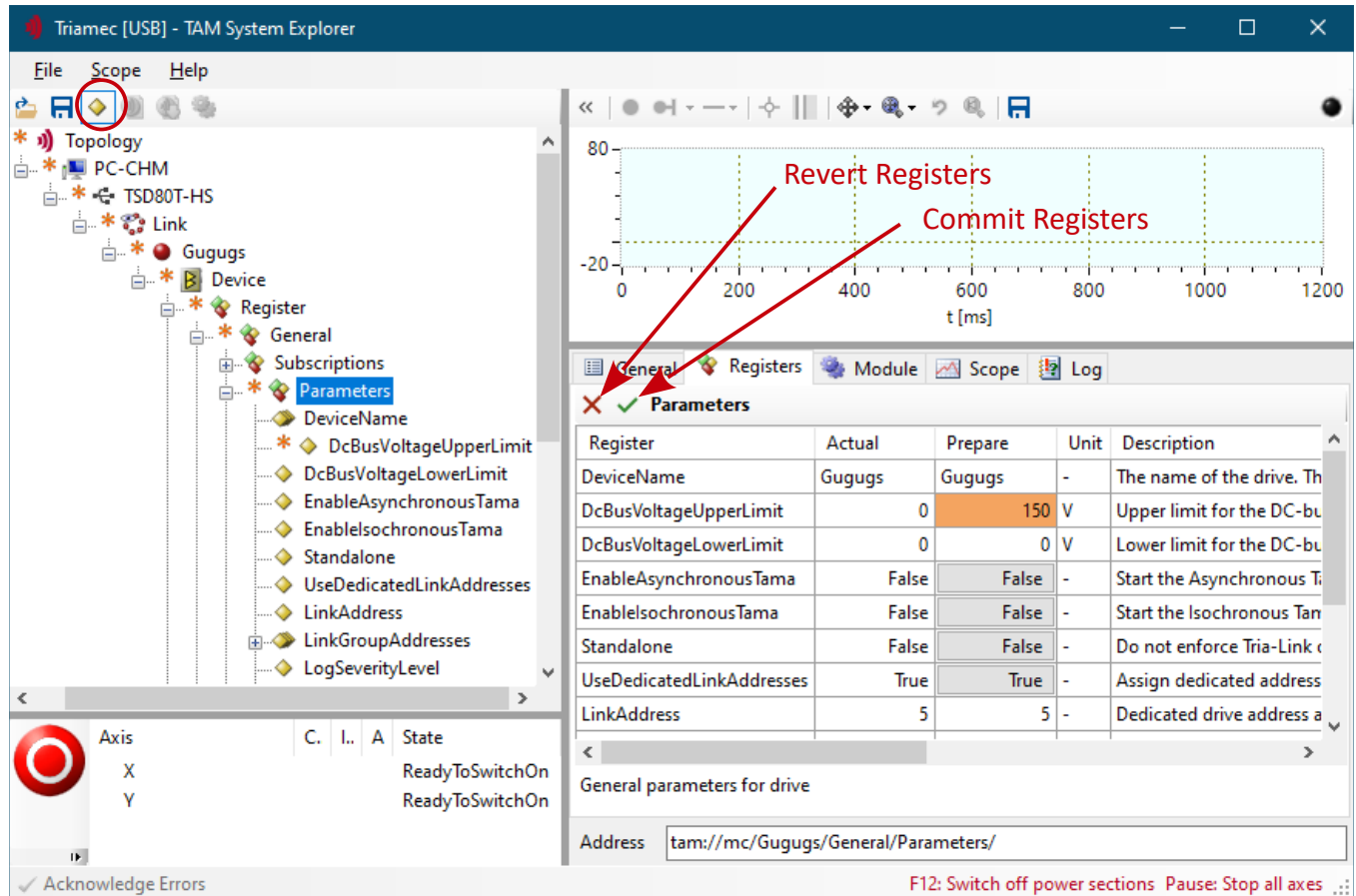


Figure 4: TAM Topology Tree and Registers

A click on a register node or a register opens the **Registers** tab in the *Tab Panel*. All registers contained in the selected register node are listed in the **Registers Tab Panel**.

3.1.1 Parameters

Parameters are used to configure the drive for the intended application, e.g. to set up the position controller or the encoder. Typically, a parameter value is written when commissioning the drive and remains unchanged, except if the system is changed. Parameters can be stored in a configuration file on the PC or persisted on the drive (see sections 3.5).

In the **Registers** tab (see Figure 4) the *Parameters* are listed with the following columns:

- Register → the name of the register, corresponding to the name in the tree
- Actual → the actual register value stored on the drive
- Prepare → the register value which will be written to the drive after clicking **Commit**
- Unit → physical unit of the value
- Description → short description of the parameter

Only the prepare field is editable. If the prepare value has a gray background, the register is read only. A changed value gets an orange background. This state remains until it gets committed.

Parameter handling buttons:

- ✓ **Commit** Updates the drive with the values of the prepare fields.
- ✗ **Revert** Reverts the prepare fields with the values from the actual fields.

Setting a parameter value:

1. Enter the desired value into the prepare field of the appropriate register
2. Press Enter or click into another register. The background of the edited field gets orange.
3. Click the **Commit** button to transmit the value to the drive. After a successful commit, the background of the prepare field turns back to white.

A star (*) in front of a tree item indicates an uncompleted commit.

3.1.2 Commands

Commands are used to change the state of the device, e.g. to set a digital output or to execute a move. The state of a command can not be stored and will be reset after a restart of the device.

In the **Registers** tab (see Figure 4) *Commands* are listed with the following columns:

- Register → the name of the register, corresponding to the name of the tree view
- Value → the actual value of the command
- Unit → physical unit of the value
- Description → short description of the command

Command values can be edited directly, no commit command is required to execute the command:

1. Enter the desired value into the value field of the appropriate register.
2. Press Enter or click into another register to execute the command.

3.1.3 Signals

Signals allow real time observation of the state of the drive.

In the **Registers** tab (see Figure 4) *Signals* are listed with the following columns:

- Register → the name of the register, corresponding to the name of the tree view
- Value → the actual value of the signal
- Unit → physical unit of the value
- Description → short description of the signal

All signals are read only.

3.1.4 Information

For each axis registers `Axes[]`.Information can be used to document the properties of the axis. Information registers do not affect the behavior of the drive in any way. Information registers are also stored with the TAM configuration file on the PC or persisted on the drive (see sections 3.5).

Note The register `Axes[].Information.AxisName` is used by the axis monitor, the scope and the modules to display the axis identification.

3.2 Axis Monitor

Below the Topology Tree the axes are listed in the *Axis Monitor* as a flat list.

The axes are identified by the register `Axes[].Information.AxisName`. Initially the station name is added in front of `AxisName`, but will be removed when the axis name is changed the first time.

The axis name can be changed by selecting the axis name in the axis monitor and clicking it again or by pressing **F2**. The modified axis name is also stored in the register `Axes[].Information.AxisName`.

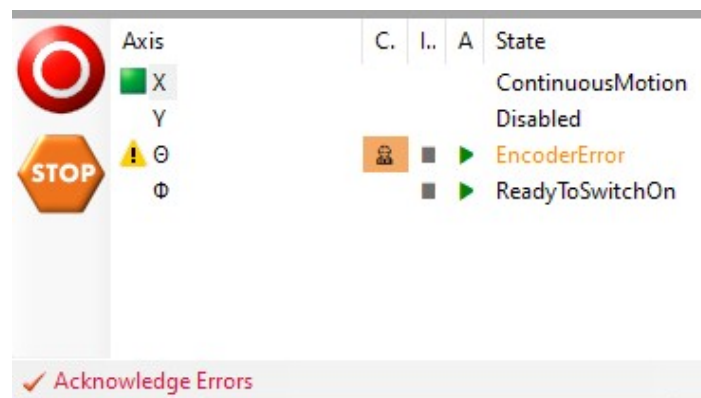




Figure 5: The axis monitor shows the state for each axis and allows to acknowledge pending errors. Additionally, the emergency buttons are provided.

The axes are highlighted according to their state:

- **Enabled:** Enabled axes are marked with a green square.
- **Error:** If an axis has a problem, the axis is marked with an error icon. The error type is shown in the **State** column, highlighted in red. Resolved errors don't simply disappear. They need to be acknowledged using the  **Acknowledge Errors** button. If no errors are pending for acknowledgment, this button is grayed out.
- **Warning:** Orange state color indicates a warning.

Warnings as well as errors which can't yet be acknowledged need some interaction, for example providing the required DC bus voltage in case of a `DcBusVoltageOutOfRange` warning.

The **Control system overridden** column shows a slowly blinking  icon on orange background as long as an axis module is attached (see chapter 4.1).

The **Asy VM** and **Iso VM** columns reflect the states of the asynchronous and isochronous *Tama* virtual machines (see chapter 3.8). Different characters are shown depending on the state.

Character	Meaning
(empty)	There is no Tama program on the device
■	The virtual machine is not operational
▶	The virtual machine is operational
-	The state is not retrieved and therefore unknown

In case of a communication failure, the state of a device can be shown as Unresponsive, or even as Link down, if the entire link is broken.

Caution In some cases¹ the communication between the *Axis Monitor* and the drive could be suspended. In these cases, the axis state is probably not reflected correctly. Check the registers General.Signals.DriveState and Axes[].Signals.General.AxisState instead or restart *TAM System Explorer* (**File > Restart**).

The axis monitor also provides two emergency buttons:



Emergency switch off: All power sections will switch off immediately, if this button is pressed. The axes are then released and may move until the kinetic or potential energy is dissipated. Pressing the emergency button will cause an external error which has to be reset e.g. by pressing the ✓ **Acknowledge Errors** button.



Emergency Stop: If emergency stop is pressed, the path planner will issue a stop move based on the emergency parameters. This brings all active axes to a halt. The axes will remain enabled.

Caution Switching off the power section of a drive during motion may damage the mechanical axis.


¹ Communication is suspended after a soft reboot of a drive with persistent link address observed via Tria-Link, or after an initialize of the Tria-Link executed by another control system.

3.3 Scope

The scope is a very powerful tool with a lot of functionalities. It allows plotting all register items (*Signals, Parameters, Commands*) with a sampling rate of up to 100kHz. To add a register item to the scope, simply drag it from the *Topology Tree* and drop it into the *Scope Area* (see also Figure 6).

Remarks Drag and drop of registers is not possible while the scope is running.

Registers from a *second* drive can only be added (drag and drop) when the two drives are synchronized (e.g. if they use the same link).

If the *Topology Tree* doesn't show the registers, click the **Show/Hide individual registers** Button .

Scope settings can be set by opening the **Scope** tab in the *Tab Panel* (see section 3.3.1). Controls related to the scope can be found either

- in the scope menu in the main menu bar,
- in the scope tool bar
- or by opening the context menu with a right click on the *Scope Area*.

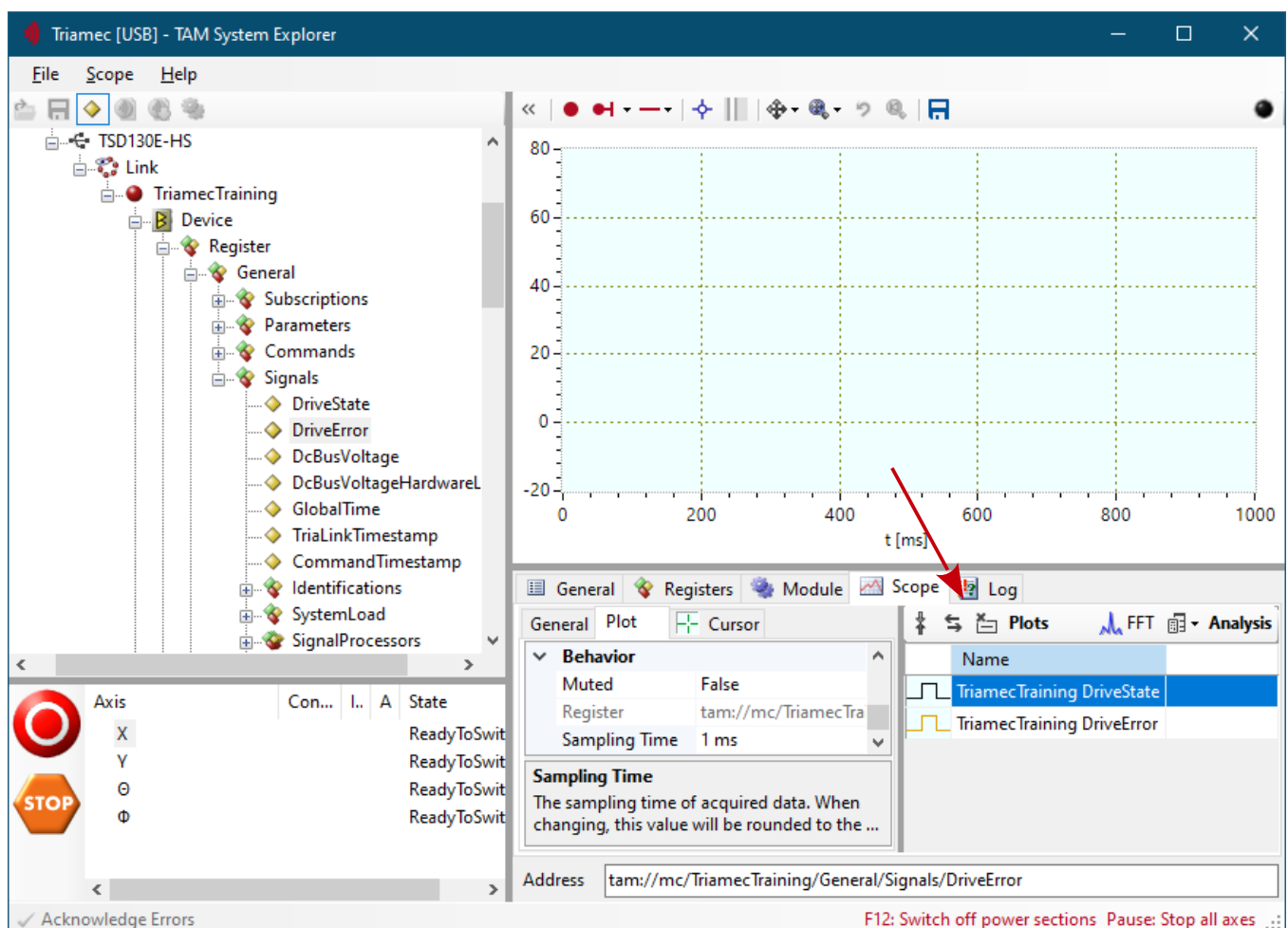



Figure 6: Scope actions: Drag and drop the registers you want to visualize into the scope.

3.3.1 Scope Settings

To modify the scope settings, click the **Scope** tab in the *Tab Panel* or click the *Scope Area* to activate the **Scope** tab. Its right area contains the list of registers (called *Plots*) assigned to the scope. The left area displays the tabs with the scope settings, which are divided into three groups:

- The **General** tab specifies scope properties valid for all plots.
- The **Plot** tab specifies the properties of the currently selected plot.
- The **Cursor1** or **Cursor2** tab owns the properties of the two cursors and the trigger.

To remove a plot from the scope, select the plot entry in the **Plots** list and press Delete or click the **Delete the selected plot(s)** button .

Remark Scope settings can not be modified while the scope is running.

3.3.2 Sampling Time

The default sampling time is 0.1 millisecond. The sampling time can be increased or decreased by changing the **Default Sampling Time** in the **General** tab for all signals or by changing the **Sampling Time** in the **Plot** tab for individual signals (Figure 12 and Figure 13). The scope supports the following sampling times: 0.01ms, 0.02ms, 0.1ms and multiples of 0.1ms. Unsupported values will be rounded to the next valid sampling time.

The maximal number of values which can be received from a drive depends on the sampling time and the data type of the register. Most registers contain 32-bit values, only absolute positions without a 'Float' identifier are 64-bit values. The following table shows the maximum number of plots depending on the sampling time and the register type:

Sampling Time [ms]	32-bit values	64-bit values
0.01	8	4
0.02	16	8
≥ 0.1	80	40

When the maximal number of plots is exceeded, an error message will pop up (Figure 7). In this case the number of plots must be reduced or the sampling time increased.

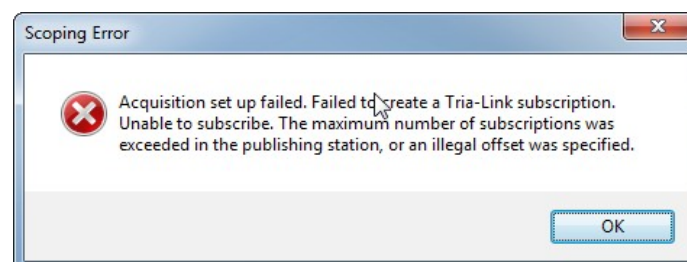


Figure 7: Scoping Error

Remarks A large number of plots with small sampling time will cause a huge data stream. Depending on the performance characteristic of the PC this could severely reduce the responsiveness of *TAM System Explorer*, especially if expensive plot styles are used e.g. if **Plot > Appearance > PointStyle** is different from **None**.

Other applications using the same link (e.g TwinCat) can reduce the maximum amount of registers which can be plotted.

3.3.3 Sampling Duration

The sampling duration can be set by clicking on the minimum/maximum label of the time axis in the scope (Figure 8). The unit of the time axis is millisecond.

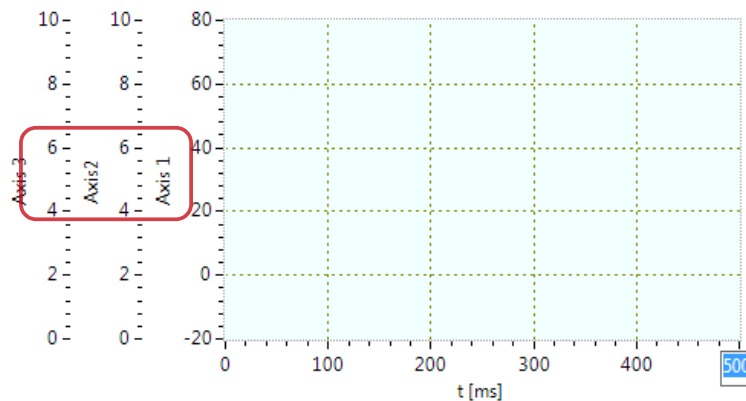


Figure 8: Sampling duration and multiple Y axes.

Remark Long sampling duration and small sampling time with a large number of plots will cause a huge amount of data. Depending on the characteristic of the PC this could cause memory issues, especially on 32 bit PCs.

3.3.4 Y Axes


Y axes can be created, edited and removed by opening the **YAxis Collection Editor** by clicking **General > YAxis > [icon]** in the **Scope** tab (see also Figure 12).

- Use **Appearance > Caption** to name the axis.

To assign a plot to an existing axis, select the plot in the **Plots** list and assign the axis with the pull-down menu from **Plot > Axes > YAxis**.

Y range:

- The min/max range of an axis can also be changed by clicking on the min/max axis label (Figure 9).
- Clicking on the upper part of a Y axis will scale into the range, clicking on the lower part scales out.
- The Y axis has a context menu (click right on the axis) with entries to achieve the same effect.

- You can also move plots up and down by dragging their Y axis.
- Use the center button  to vertically center the selected plot.

The scope always activates the Y grid of the Y axis that the mouse hovered over last. The Y grid is also adjusted when hovering over a plot in the **Plots** list. This can be disabled by making a Y grid sticky with the context menu of the axis as shown in Figure 10.

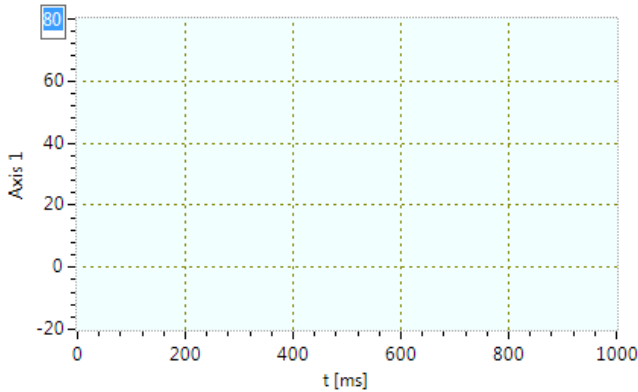


Figure 9: Changing the Y axis range

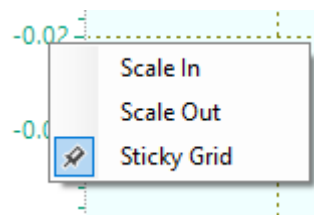







Figure 10: Y axis context menu items

3.3.5 Triggering

To start scoping press the **Start** button . By default the scope is in single trigger mode . For automatic triggering click  to toggle to repeat mode  or set the parameter **General > ScopeCharting** to **True**.

Signal trigger: It is also possible to use a signal from the **Plots** list as a trigger source. The following steps are required to setup a signal trigger:

1. Select a trigger mode from the **Select triggers** pull-down menu (Figure 11):
2. Assign the trigger signal to the **Plot** property in the **Cursor** tab by selecting the desired plot from pull-down menu (see also Figure 14).
3. Set the desired trigger level to the **Level** property in the **Cursor** tab.
4. To define the pre-trigger and the post-trigger duration set **Axes > TimeAxis > Range > Minimum/Maximum** in the **Plot** tab or click on the min/max label of the time axis (Figure 9).
5. The scope shows a red draggable cross-hair which can be used to set the trigger level and the trigger time.
6. Press the **Start** button  to activate the trigger.

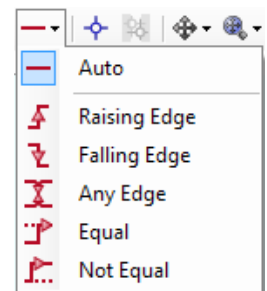


Figure 11: Trigger mode.

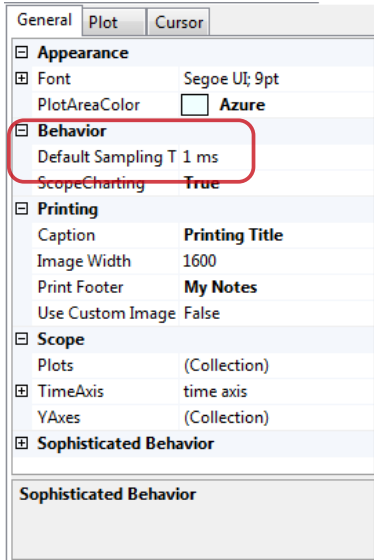


Figure 12: General tab of the scope. The most commonly used parameters are Default Sampling and YAxes.

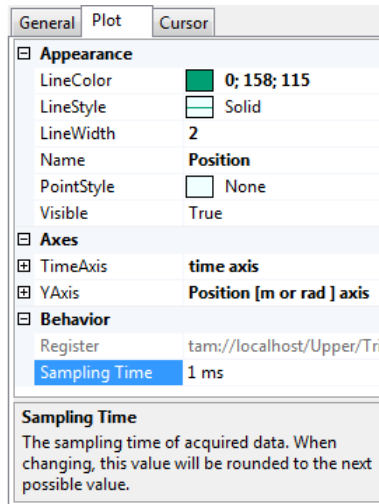


Figure 13: Plot tab of the scope. The most commonly used parameters are LineColor, Name and Sampling Time. Selecting several plots allows for configuring them altogether.

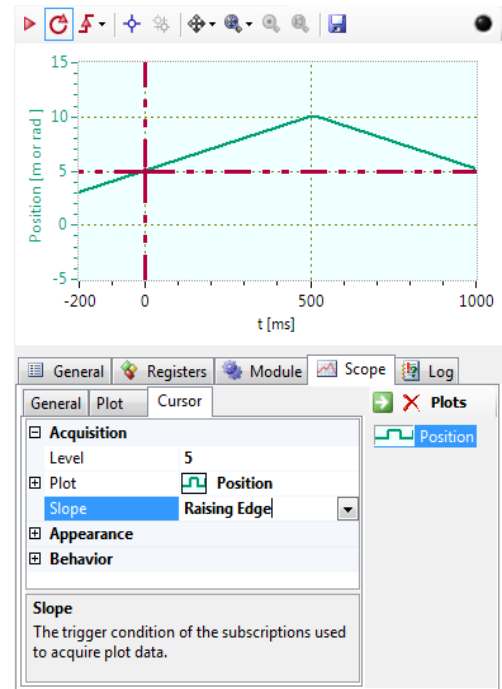




Figure 14: Repeat and trigger buttons. A horizontal draggable line marks the trigger level, the vertical draggable line defines the time of the trigger event. The Plot property defines the trigger source.

It is possible to store the data of each trigger event automatically to a file on the PC with **Auto-Save**. Auto-save mode can be activated by clicking the pull-down-menu to the right of the repeat icon  and select **Auto-Save**. When auto-save is active, the repeat icon shows a small blue disc symbol . The auto-saved data from the trigger events can be accessed from TAM System Explorer by clicking **File > Open Workspace Folder > Measurements\autosave*.csv** (see also [1]).

3.3.6 Zoom and Pan

The scope supports different zoom and pan operations.

Important For zooming press Shift while defining the zoom area with the mouse or rolling the mouse wheel.

For panning press Ctrl while dragging the *Scope Area* with the mouse.

Zooming or panning are only enabled when the scope is not running.

The zoom pull-down menu (Figure 15) allows to set the desired zoom behavior (horizontal/vertical/both) and provides additional commands related to zooming:

- **Zoom to plots:** Sets the plot axes to the actual limit values of the curves.
- **Zoom to cursors:** Restrict horizontal view to the range between the cursors (only available when both cursors are applied).
- **Arrange Plots:** Automatically assign similar types of signals to one axis and arrange axes without overlap of the signals.
- **Fix zoom range:** Set the default min/max range of the axes according the current zoom range.
- **Undo zoom:** Undo the last zoom/pan operation.
- **Reset zoom range:** Resets the min/max range to the default values.

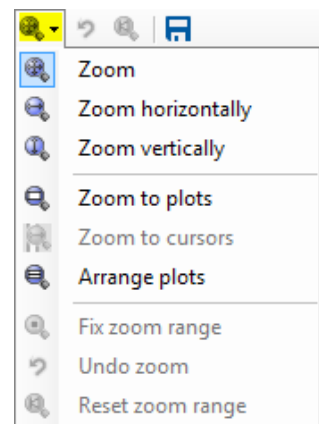


Figure 15: Zoom menu items.

3.3.7 Saving and Loading Scope Data

Use the **Scope** menu to save and load plot data (Figure 16):

- The scope data can be stored as a *.csv file by using **Scope > Save Plot Data....**
- To open an existing data file in the scope, use **Scope > Load Plot Data....**
- Further information about saving and loading plot data can be found in the application note about data exchange [1].

Scope settings can be saved to avoid time consuming reconfiguration:

- The plot settings, ranges, color and more, but without data, can be saved by executing **Scope > Save Configuration...**
- A saved configuration can be loaded by executing **Scope > Load Configuration....** Already configured plots are not affected by loading a stored configuration.

To print or save an image of the scope including labels, use **Scope > Print Plots..., Copy Image to Clipboard** or **Save Plots as Image....** It is useful to customize the prints in advance by changing properties of the **Printing** category in the **Scope > General** tab.

3.3.8 Loading Templates

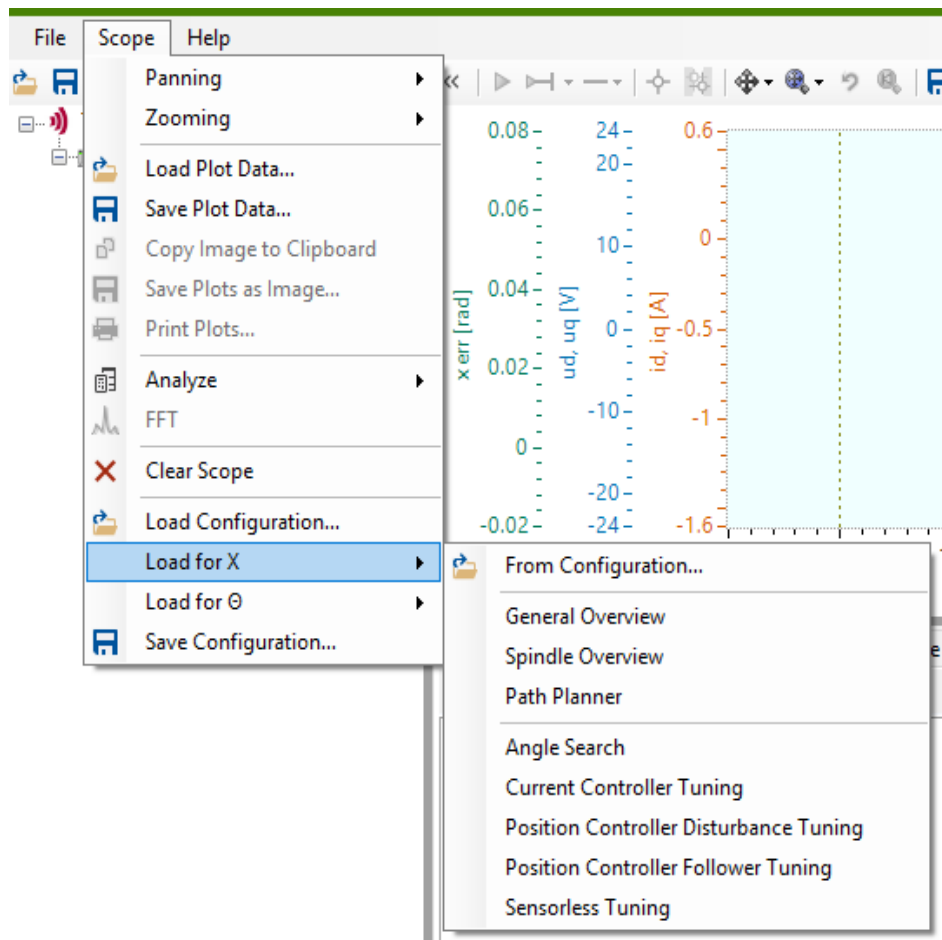


Figure 16: The Scope Menu

The first section of the scope menu is related to zooming and panning. The next section is used to load and save plot data followed by an analysis section and a section to reset the scope. The last section handles the configuration of the scope.

For each axis in the *Topology Tree*, the scope menu offers a **Load for <axis name>** entry. When expanded, a number of templates are available as shown in Figure 16. Each template will set up the scope for a specific use with respect to the selected axis.

Several templates may be loaded in sequence for different axes. Use the **Clear Scope** menu to reset the scope to an initial state.

The first sub-menu entry **From Configuration...**, allows to treat any previously saved scope configuration as a template for the currently selected axis. This is mainly used for configurations where all plots refer to the same axis.

3.3.9 Analysis

The scope incorporates the most frequently used analysis functions.

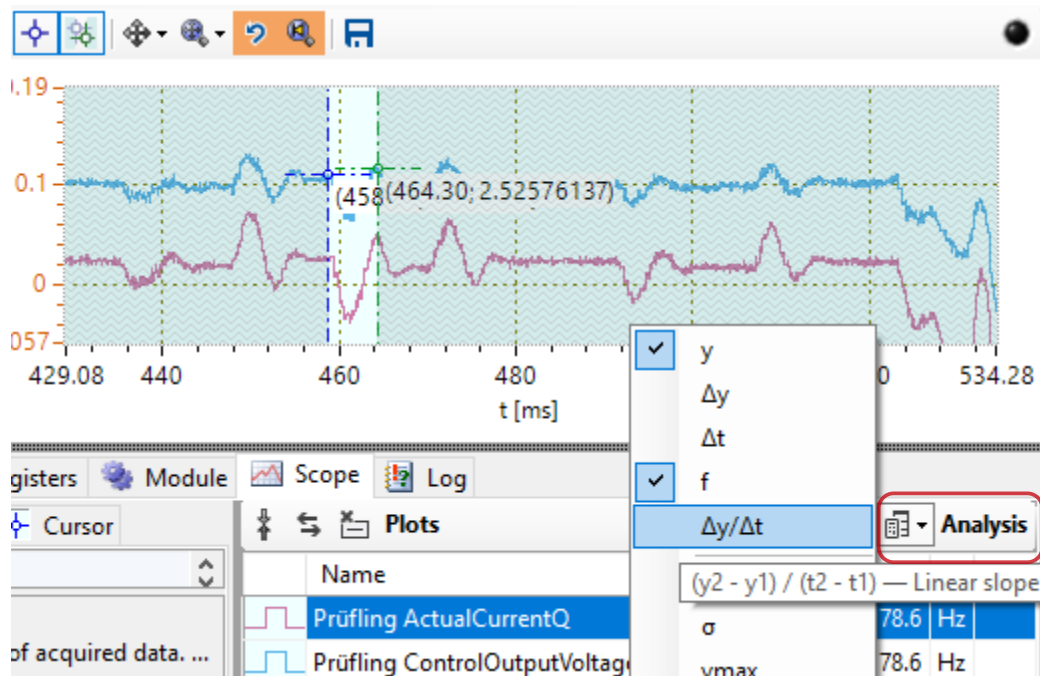


Figure 17: Use the primary cursor to inspect single points in time. Add the secondary cursor to restrict analysis to an interesting range. Now pull in statistical measures from the Analysis drop down.

As shown in Figure 17, the two cursors can be used to select and analyze data. Each measure is shortly explained by its tool-tip. The first group of measurements only examine the two points at the cursors, while the second group considers all the data in the range defined by the two cursors. A transparent shading is shown to indicate parts of the scope *not* taken into account for calculations.

When the secondary cursor is inactive, the whole visible range is considered for calculation.

Analysis also features an *FFT* calculation. When invoked, another scope view is opened with the selected data plotted against the frequency domain.

3.3.10 Formatting

The formatting and behavior of the scope, individual plots and cursors can be configured in the respective tabs.

3.4 Triamec Workspace

When working with *TAM System Explorer*, different files are used or generated e.g. configuration files, measurement data or firmware files. The *Triamec Workspace* helps to manage and organize these files. Basically the workspace is a directory where all files are stored in dedicated folders. Using the workspace simplifies file handling, backup and support. For different applications and machines it is recommended to use separate workspaces.

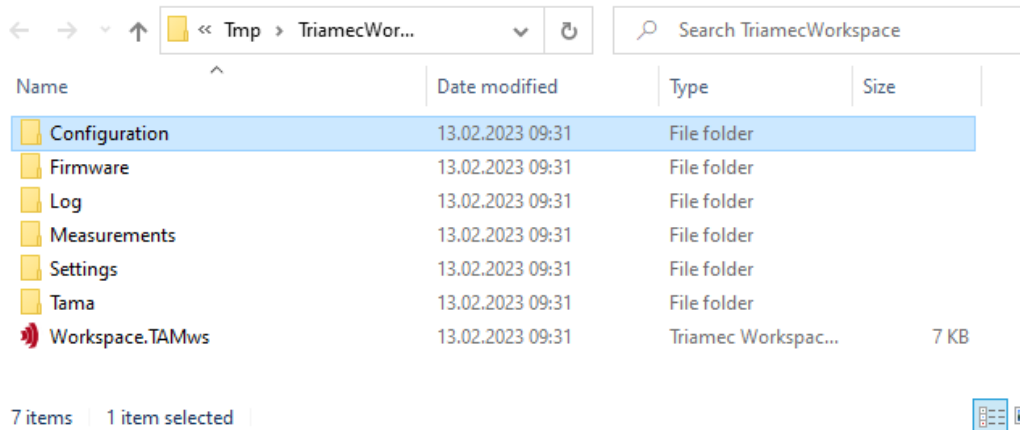


Figure 18: Triamec Workspace in File Explorer

The following table shows the structure of the workspace.

Path		Contents
'WspName'	Configuration	TAM and scope configurations
	Doc	Documentation, readme, checklists
	Firmware	Place device firmware files here
	Log	Application log files
	Measurements	Frequency Response measurements, plot data and pictures
	Tama	Tama programs
	Projects	Visual Studio projects, i.e. for creating Tama programs

A default workspace is generated when the *TAM Software* is installed. This default workspace folder is named **Triamec** and it is located in the **PublicDocuments** folder. When *TAM System Explorer* is started the first time, this workspace will be loaded. After a restart of *TAM System Explorer*, the last used workspace will be loaded automatically. Use the **File** menu to handle the different workspaces:

- **File > New > Workspace:** creates a new workspace.
- **File > Open Workspace:** loads an existing workspace.
- **File > Open Workspace Folder:** opens the currently loaded workspace folder in Windows File Explorer.
- **File > Recent Workspace > ... :** select and load a recently opened workspace.

The currently active workspace folder is shown in the title bar of *TAM System Explorer*.

3.5 TAM Configuration

The parameterization of a *Topology Tree* is called *TAM Configuration*. The *TAM configuration* contains the parameterization of the drives, the names of the nodes in the topology and other information.

The *TAM Configuration* can be saved

- **persistent on the drive:** If the *TAM Configuration* is saved persistent on the drive, this configuration becomes active directly after restarting the drive. A drive without a persistent *TAM Configuration* will load an initial configuration on start-up with default values for all registers.
- or as a ***.TAMcfg file on the PC:** The *.TAMcfg file is useful to exchange a configuration between machines of the same type or to save and archive the configuration status of a machine. It is recommended to save the configuration during setup of the configuration occasionally to have a backup in case of a problem.

The *.TAMcfg file is an *XML* file and can also be opened with a suitable editor (e.g. *Notepad++*) or with a file comparison tool to compare two different states of the configuration (e.g. *WinMerge*). Scope settings are *not* part of the *TAM Topology Tree* and have to be stored separately (see section 3.3.7).

3.5.1 Saving the TAM Configuration Persistent on the Device

Conclude the following steps to save the *TAM Configuration* persistent on the drive (see also Figure 19):

1. Right-click the station of the drive and select **Manage Persistence ...**.
2. Click **Save parameter permanently** to store the setting to the drive.

The **Manage Persistence ...** dialog also allows reverting the current configuration or to disable the persistence.

Remarks During the activation of the persistence it is possible that the drive state changes to Un-responsive for a short moment.

Persistence has to be set for each drive separately.

To reset the drive to default values, disable persistence and power cycle the drive.

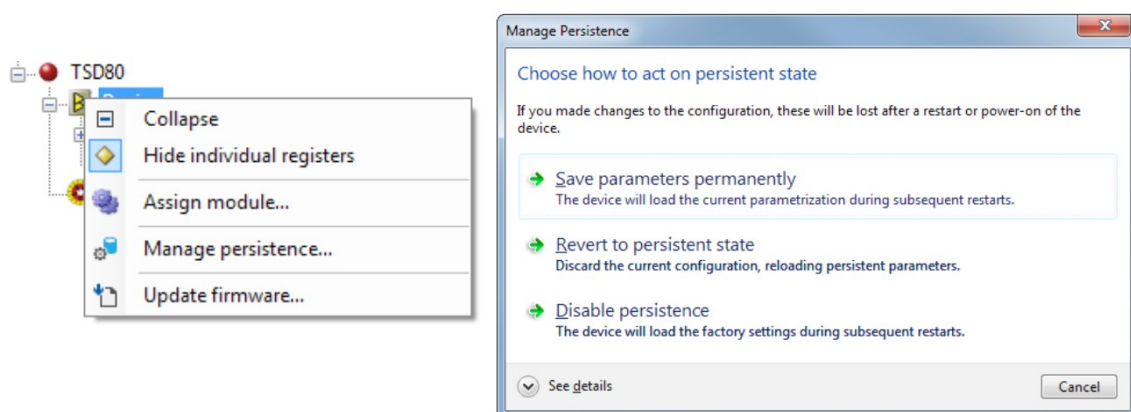


Figure 19: Persistent TAM Configuration

3.5.2 Saving the TAM Configuration on a PC

To save the *TAM Configuration* on the PC the following steps are required (see also Figure 20):

1. Open the save configuration dialog by clicking **File > Save TAM Configuration...**
2. The **Includes** panel allows to select/unselect items, which should be stored with the *TAM Configuration*. It is recommended to use the default settings to save the configuration.
 - ♦ **Registers:** All parameter and information registers will be stored.
 - ♦ **Tama assemblies:** If a *Tama* program is loaded to the drive either the path to the Tama program or the Tama code itself will be saved to the *TAM Configuration* (see also section 3.8).
 - ♦ **Module assignments:** If a module is assigned to the *Topology Tree*, it will be saved to the *TAM Configuration*.
 - ♦ **Module parameters:** Parameters related to the module will be saved.
3. The **Coverage** panel allows to select which part of the *Topology Tree* should be saved:
 - ♦ If **Starting from topology** root is checked (default), the whole *Topology Tree* with all devices assigned to it will be saved.
 - ♦ If **Starting from tam://...** is checked, only items below the currently selected item in the *Topology Tree* will be saved. This is for example helpful, if only the configuration of a particular device should be saved.
4. Clicking the **Save** button will open the **Save As** dialog.
5. After choosing a file name and clicking the **Save** button the progress window will pop up which shows the progress.
6. In case of an unexpected exception, the process may be started again by using the **Restart** button.

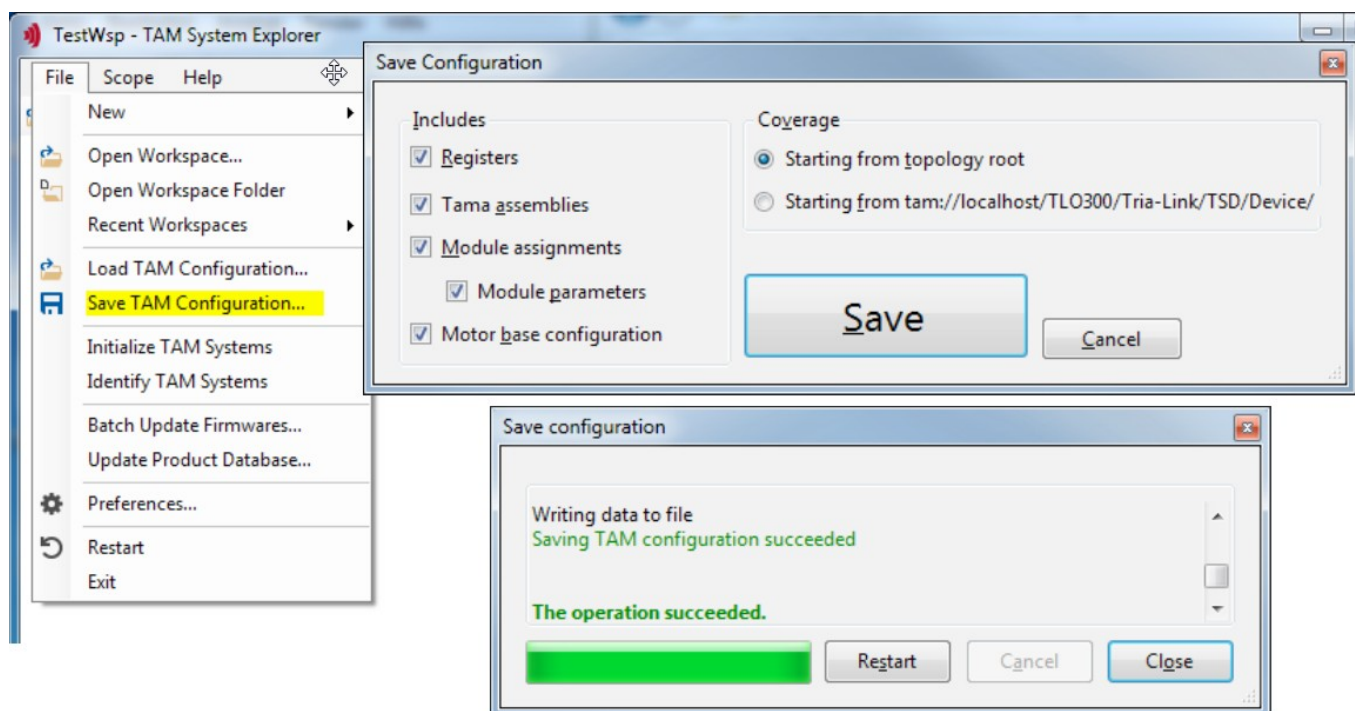


Figure 20: Save TAM Configuration on PC

3.5.3 Load TAM Configuration

The following steps are required to load a *TAM Configuration*:

1. Open the **Load Configuration** dialog by clicking **File > Load TAM Configuration....**
2. Select a matching **TAM Configuration** and click the **Open** button. A progress window will pop up.
3. If issues are detected with the configuration, you will need to manually **Start** the process after having reviewed the warnings carefully (Figure 21).
4. In case of an unexpected exception, the process may be started again by using the **Start** button.

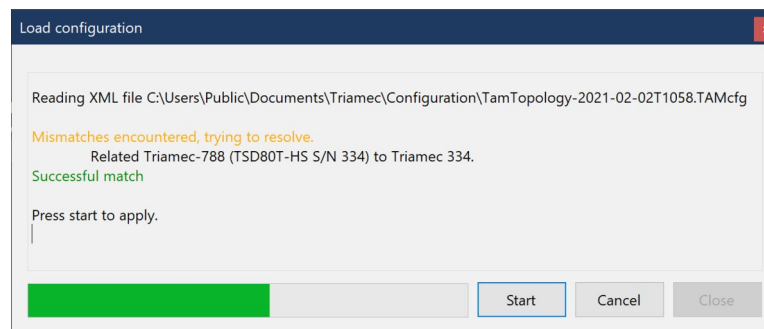
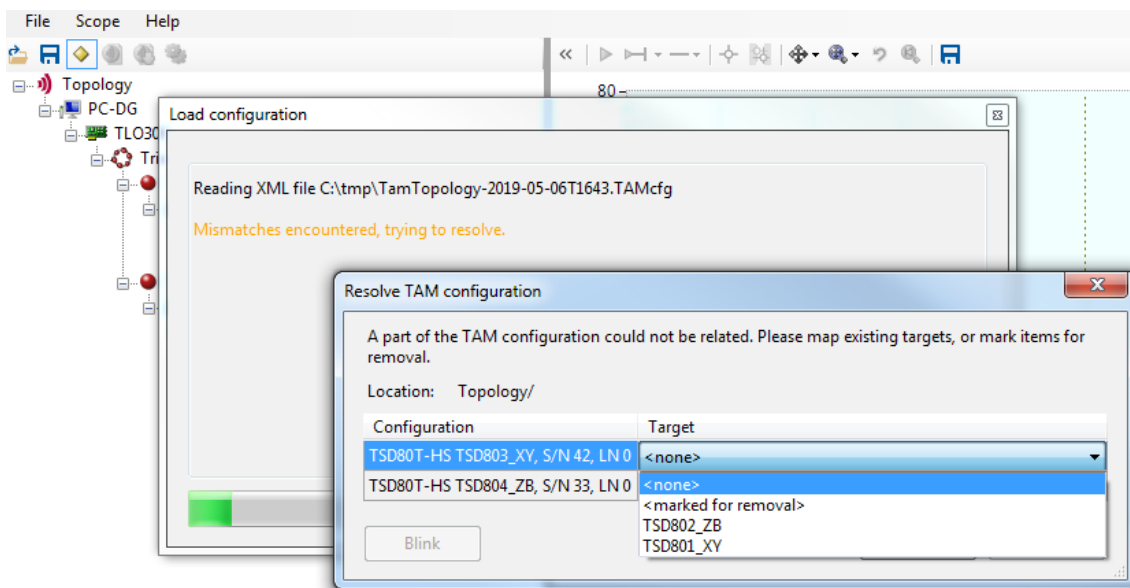


Figure 21: Configuration download progress window

When a *TAM Configuration* is loaded, the configuration is assigned to the physically existing stations according to the following scheme:

- If the name of a station in the tree does uniquely match with the name of a station in the configuration, the configuration of the station is assigned by name.
- If at this point there are still devices which are not assigned, the **Resolve TAM configuration** dialog is opened (Figure 22).
 - ♦ The dialog lists all stations available in the configuration at the left column of the table.
 - ♦ With the pull down menu in the right column a (not yet assigned) station from the *TAM Topology Tree* can be assigned to the configuration.
 - ♦ If the station does not exist in the current tree, select **<mark for removal>**.
 - ♦ To identify the target station, use the serial number or use the blink button. The blink button will cause a red blink of the axes *LED* and system status *LED*.
 - ♦ Press **Start** in the **Load configuration** dialog to apply the assignment of the configuration.
 - ♦ After a new assignment of the stations it is recommended to save the *TAM Configuration*.



- If the station name does not match or is not unique, and the serial number of the device does match, the configuration is assigned by serial number.
- Stations that can't be assigned by name or serial number are assigned by station type, if a unique assignment according to the station type is possible.

3.5.4 Working Offline

Even when you don't have access to a drive or the machine, you can open a TAM Configuration offline:

- By opening the TAM Configuration in the Windows Explorer.
- Using the menu **File > Open Configuration Offline...**

The application then simulates the topology tree as specified in the configuration. The window title is set to the name of the TAM configuration file.

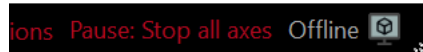


Figure 23: In order to make it clear for everyone, the status bar changes to dark mode in offline mode, and an indicator is visible on the right.

The simulation only reproduces a few aspects of the real system. Among other features, this is a convenient way to edit the configuration, including Controller Tuning.

3.5.5 Transferring a TAM Configuration

It is often required to transfer a *TAM Configuration* from one machine A (*MA*) to another machine B (*MB*) of the same type. This can be done by executing the following steps:

1. Give the stations of *MA* a meaningful name e.g. Station1_XY, Station2_ZB etc.
2. Save the configuration of *MA* on the PC.
3. Connect to *MB* and load the configuration.
4. Solve the configuration mismatch by using the serial number or the blink button.
5. Save the configuration for *MB* on the PC and persistent on the drive.

3.6 TAM Snapshot

A *TAM Snapshot* represents the state of a drive at a specific time instance. The state is represented by register values. A *TAM Snapshot* is thus closely related to a *TAM Configuration*, since it also contains all *Parameters* of a drive, but holds additional information such as *Signals* or *Commands*. A *TAM Snapshot* is generated alongside *TAM Dump Files* in a *Dump Event*, which occurs either due to an error or when manually triggered by the user. A *TAM Snapshot* can be opened offline with the *TAM System Explorer*. When a *TAM Snapshot* is opened, the binary file is additionally saved as a JSON to make the file readable and to allow differentiating *TAM Snapshots* more easily. It is possible to generate a *TAM Configuration* from a *TAM Snapshot* that has been loaded to the *TAM System Explorer* by saving the current configuration as described in 3.5.2.

3.6.1 Trigger a TAM Snapshot

In order to trigger a *TAM Snapshot* manually, a *Dump Event* event has to be triggered, which can be done by clicking on the camera symbol depicted in Figure 24 or by pressing F7.

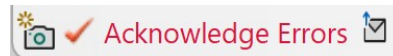


Figure 24: Bottom menu with “Record Dump” (left) and “Create Report” (right) button

3.6.2 Save TAM Snapshot to PC

A TAM Snapshot can be saved in two ways:

1. Generate a Report. This can be done by clicking the “Create Report” button shown in Figure 24.
2. Download directly from the device’s web interface. You can open the web interface of a device by right-clicking it and select Browse, refer to Figure 25. After having opened the device’s start page, go to directory in order to find all TAM Snapshots available. Select the most recent one by checking the timestamp.

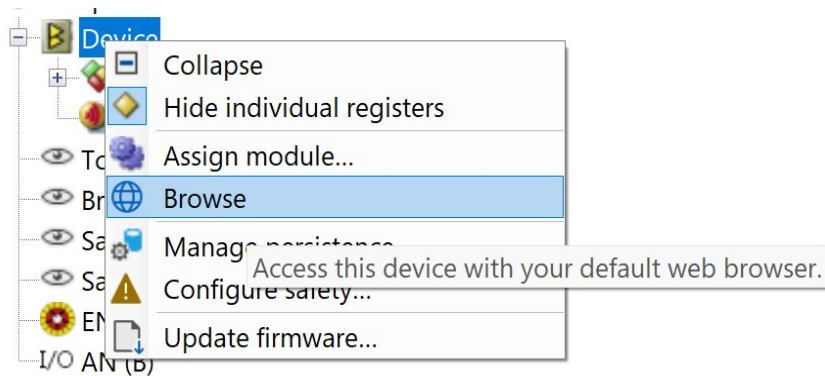


Figure 25: Browse a device’s web interface

3.6.3 Open a TAM Snapshot

A TAM Snapshot can be opened in two ways:

1. Double-click a .TAMsnap file
2. Load a .TAMsnap file via the context menu shown in Figure 26.

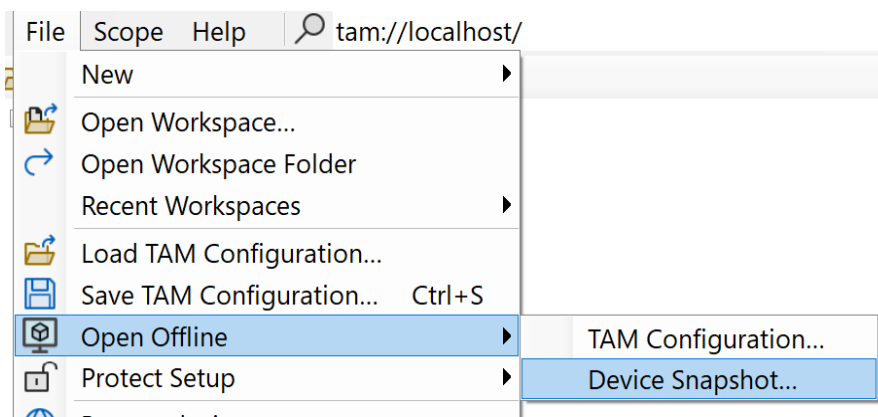


Figure 26: Open a TAM Snapshot via the context menu

3.7 Firmware Update

This section explains how to find a suitable firmware and how to download the firmware to the device.

The newest firmware release can be downloaded from the *Triamec* web page:

1. Go to www.triamec.com > **Support** > **TAM Firmware**:
2. Open the web-page matching the product type of your device (e.g. > **TSD Series**). If the product type of your device is not known, you can identify it by clicking the station of the device in the *Topology Tree*. This opens the property page in the **General** tab as shown in Figure 27. You will find the type of the device under **Product Information**. The property page also shows the currently installed firmware version and the product revision which could also be relevant for the selection of the firmware.
3. Check the release notes for relevant changes of the firmware.
4. Download the *.zip file with the firmware package and unzip the file.

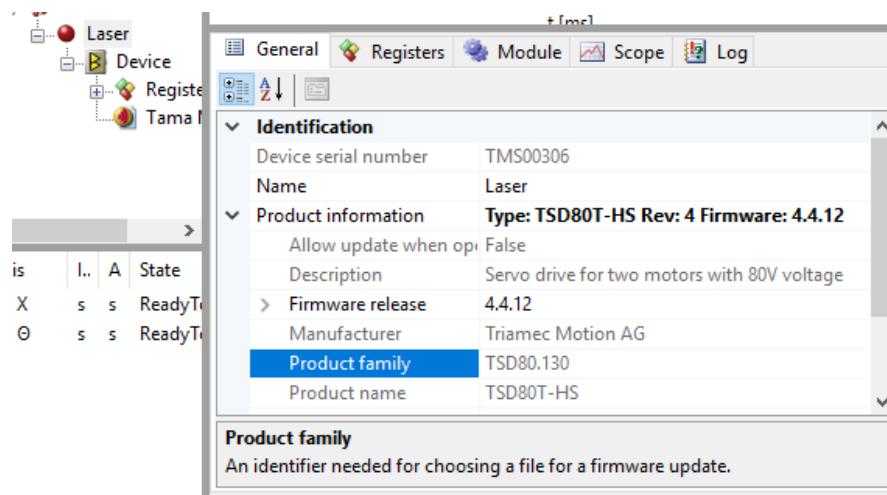


Figure 27: Station Properties.

The product information shows the type, the product revision and the firmware version.

Execute the following steps to download the firmware:

- Make sure that all axes are disabled and STO is not active.
- Right click the device to open the context menu (Figure 28).
- From the context menu choose **Update Firmware**
- Select the desired firmware and click open. A filter assures, that only compatible files are selectable. If no firmware file is selectable, make sure you have downloaded the correct firmware package.
- The **Firmware File Download** window pops up. Click **Start** to initiate the download. The green bar will show the progress of the download.
- After the download succeeded, press **Close** to close the window.

Tri-Link adapters don't have a station node, they display the **Update Firmware ...** context menu entry in their adapter node.

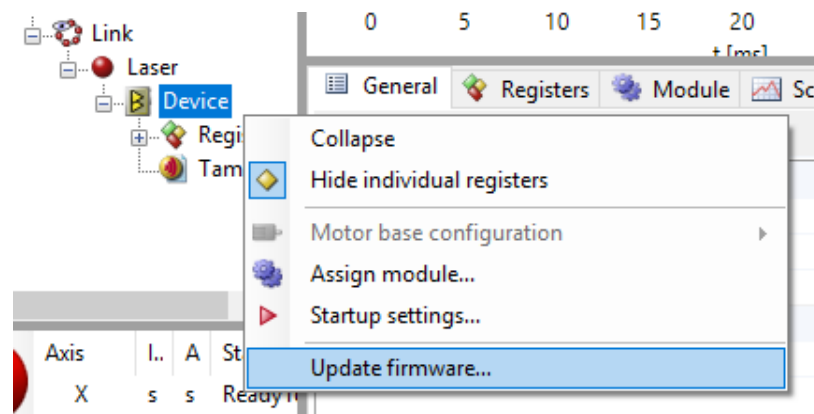


Figure 28: Firmware download via the station node.

Important If the replaced firmware version is older than 4.7 or the *TAM System Explorer* version is older than 7.10 a power cycle of the drive is required to activate the new firmware!

Warning *Safe Torque Off* must not be activated before and during a firmware download.

3.8 Tama Program

A *Tama Program* is a piece of software which can be loaded to a *Triamec Servo Drive* during run time and which is executed with up to 10kHz in real time. With a Tama Program it is possible to extend the drive firmware with user specific implementations. Typical applications with *Tama Programs* are:


- Stand alone control of a drive
- Adaptive gain controller
- Adaptive feed forward
- Compensation algorithms
- etc.

The *Tama Program* has access to all registers in the *TAM* register tree. This allows full control over the axis and the drive. The *Tama* code is typically written in C#. See [6] for a detailed documentation of how to implement and use *Tama Programs*.

3.9 Setup Protection

As a system supplier, you can protect your setup with a password. This can reduce support incidents due to modifications by unauthorized personnel.

Using the menus under **File > Protect Setup**, you can enable and disable protection. For the purpose of servicing, you can also disable protection temporarily, without changing device state, so you don't need to remember to enable protection in the end. This state is maintained only until the application is closed.

Menus and areas not available due to the protection receive a  protection icon and an explanatory tool-tip or label, respectively.




Remark This is not an all-purpose solution against deliberate tampering. Only the TAM System Explorer protects the setup using this mechanism. Customer written applications may consider setup protection, but don't need to.

Caution You must manage the password such that it doesn't get lost. If you forget the password, you will need to contact Triamec support.


Caution Since the password is hashed with a weak algorithm, don't reuse a sensible password.

3.10 Global Keys

The following keys are always active when TAM System Explorer is active.

Key	Effect
<u>F5</u>	Start/Stop Scope. This corresponds to the  button in the Scope toolbar.
<u>F6</u>	Arrange plots in the Scope. This corresponds to the  button in the Scope toolbar.
<u>F12</u>	Emergency switch off all power sections. You may also click the  button located on the left of the axis monitor.

Caution Switching off the power section of a drive during motion may damage the axis.

Pause/Break Emergency Stop of all active axes. You may also click the  button located on the left of the axis monitor.

4 Plug-In Modules

The whole functionality of a *Triamec* device can be accessed directly by interacting with the registers of *Topology Tree*. However, this can be rather extensive and complicated. To simplify dedicated tasks, plug-in modules can be assigned to the *Topology Tree*. These modules provide a graphical user interface which allows to interact with the device while reading and writing of the registers is handled by the module.



Figure 29: Axis module assignment.

With the following steps, an **Axis** or **AxisGroup** module can be assigned to a device (Figure 29):

1. Right click the device to open the context menu of the device.
2. Click **Assign module...**
3. The module is assigned to the *Topology Tree*. To access sub modules, unfold the module node in the *Topology Tree*. The user interface is shown in the *Tab Panel* in **Module > View**. Parameters assigned to the module are shown in **Module > Parameter**.

To remove a plug-in module open the context menu of the module, then click **Remove module**.

4.1 Axis Module

The axis module can be used to execute basic movements of the axis. For dual-axis drives an **Axis Group**-module should be added to the *Topology Tree*. Unfold the module-node and select the desired axis. The user interface of the axis is then shown in the *Tab Panel* in **Module > View** (Figure 31). If the **Axis** node is expanded further, additional tools like *Frequency Response Tuning* are accessible.

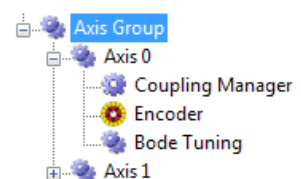


Figure 30: Unfolded Axis Group

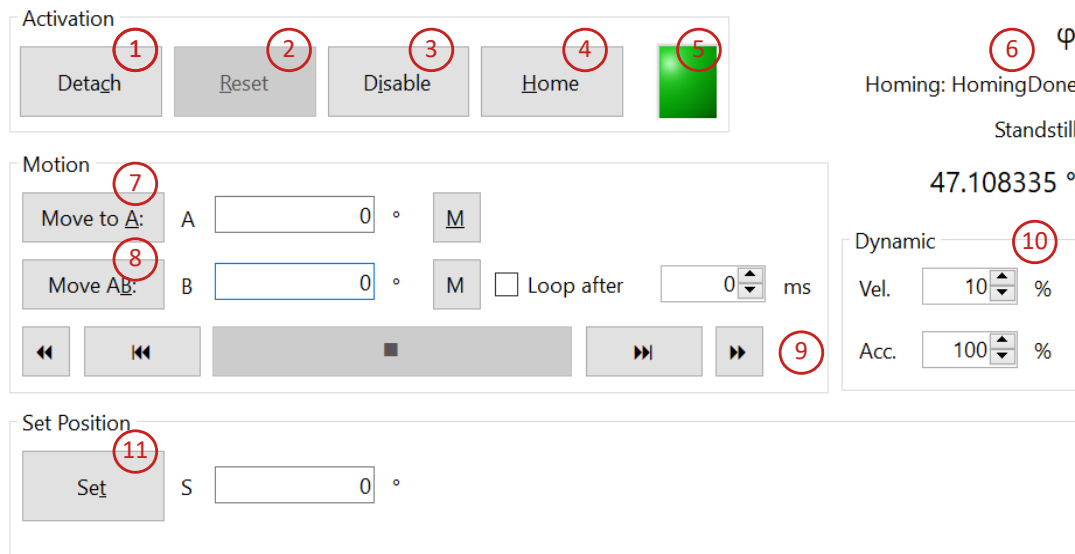


Figure 31: User interface of the axis module

The axis module offers the following controls to interact with the axis (see also Figure 31):

1. **Attach/Detach:** In case the **Attach** button is shown, this button needs to be pressed in order to enable the controls of the module. This button prohibits commands from the superior control system (e.g., EtherCAT) to allow the module to control the axis. Pressing the button sets `Axes[].Commands.General.OverrideControlSystem` to 1. Release the button (**Detach**) to return control to the superior control system.
2. **Reset:** The **Reset** button can be used to reset pending errors or warnings. If the reset was successful, the drive returns to the Disabled state. If the reset was not successful, additional actions are needed by the user depending on the type of the error/warning.
3. **Enable/Disable:** This button is used to enable or disable the axis. After the axis is successfully enabled, the axis state is Standstill. If **Disable** is pressed, the axis-state will change to Ready-ToSwitchOn or Disabled.
4. **Home:** If a homing sequence is configured (see chapter 5.2.11), you can initiate it with this button in the Standstill state.
5. Status indicator.
6. At the top right corner the following information is displayed:
 - ♦ The axis name (`Axes[].Information.AxisName`)
 - ♦ The homing state (`Axes[].Signals.Homing.State`), if homing is configured
 - ♦ The axis state (`Axes[].Signals.General.AxisState`)
 - ♦ The actual position (`Axes[].Commands.PositionController.MasterPosition`)
7. **Move to A:** Pressing this button causes the axis to move to the position specified in field **A**. Use the corresponding **M**-button to load the actual position to field **A**.
8. **Move AB:** Pressing this button causes the axis to move towards position **A** and then to position **B**. Use the corresponding **M**-button to load the actual position to fields **A** or **B**. If the axis is already located at A, only a move to B is executed. If the **Loop After**-checkbox is checked, the move sequence A B will be repeated until another move button or the stop button ■ is pressed. The wait time field at the right is used to set the delay at A and B before the next move starts.
9. Jog and Stop buttons:

- ♦ Press the stop button ■ to terminate an ongoing move, or abort an ongoing homing sequence, respectively.
- ♦ Continuous jog negative ◀◀ / positive ▶▶: Pressing this button starts a continuous move. The move will run until another move button or the stop button ■ is pressed.
- ♦ Jog negative ◀◀ / positive ▶▶: While pressing this button the axis moves in the desired direction. The move will stop when the button is released.

10. **Dynamics:** These two fields can be used to reduce the dynamics of the move.

- ♦ **Vel.:** This field specifies the velocity used for the move relative to `Axes[].Parameters.PathPlanner.VelocityMaximum`.
- ♦ **Acc.:** This field specifies the acceleration used for the move relative to `Axes[].Parameters.PathPlanner.AccelerationMaximum`.

11. **Set Position:** Pressing **Set Position** sets the actual position to the value specified in field S. If applicable, homing will be invalidated.

Important If the **Activation** panel shows an **Attach** button, this button needs to be pressed in order to enable the controls of the module. Release the button (**Detach**) to return control to the superior control system.

Sub-Modules

- **Frequency Response Tuning:** See chapter 4.2 *Frequency Response*.

4.2 Frequency Response Module

The **Frequency Response** module is used to measure the dynamic characteristic of an axis in frequency domain and to tune the controller based on this measurement. The following sections describe the steps required.

Find the **Frequency Response** module in the **Axis Group** module and open the **View** tab. The **View** tab contains the following buttons:

- **Measure...** opens the **Frequency Response Measurement** application. The application allows to configure and to execute the measurement.
- **Tune...** opens a *Frequency Response* measurement in the **Frequency Response Tuning** editor. The editor allows to modify the controller parameter and directly visualizes the effect of the modification.

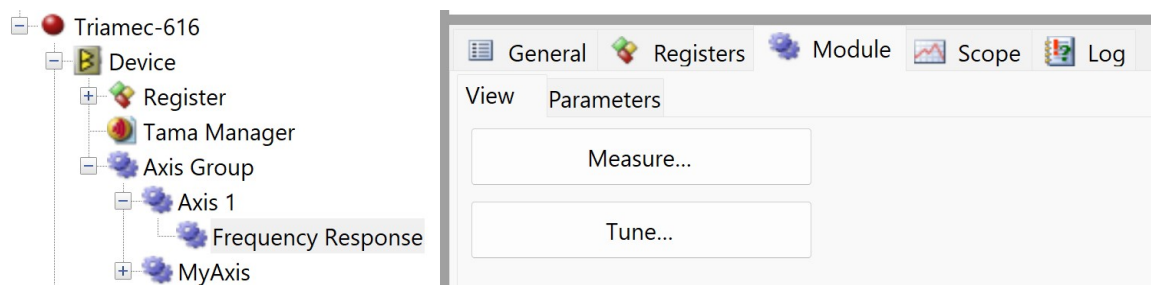


Figure 32: Frequency Response module.

4.2.1 Frequency Response Measurement

The Frequency Response measurement allows to acquire the frequency response of the axis. This measurement is required for *Frequency Response* tuning and therefore has to be executed first.

4.2.1.1 Preparation:

Warning The *Frequency Response* measurement causes the axis to move. Therefore a safe environment and setup is required and the drive has to be configured correctly.

The following parameters must be set correctly. An incorrect parameterization could influence the measurement results or damage the motor. See also section 5.2 on how to configure these parameters.

- Axes[].Parameters.Motor.Type
- Axes[].Parameters.Motor.InvertDirection
- Axes[].Parameters.Motor.PolePairs
- Axes[].Parameters.Motor.EncoderCountsPerMotorRevolution
- Axes[].Parameters.Motor.PeakCurrent
- Axes[].Parameters.Motor.NominalCurrent
- Axes[].Parameters.PositionController.Encoders[].Type
- Axes[].Parameters.PositionController.Encoders[].InvertDirection
- Axes[].Parameters.PositionController.Encoders[].Pitch
- Axes[].Parameters.PositionController.OutputLimit
- Axes[].Parameters.Commutation. ...
- Axes[].Parameters.CurrentController.OutputLimit
- General.Parameters.EncoderTopology
- General.Parameters.DcBusVoltageUpperLimit
- General.Parameters.DcBusVoltageLowerLimit

Important Make sure that the axis is moving freely during the measurement. If the axis is touching an obstacle or an end stop, the measurement will not reflect the correct behavior of the axis.

To avoid moving against an end stop e.g. if the direction of the axis is vertical, a soft material like foamed plastic can be used to underlay the axis. If the axis is equipped with a brake, the brake has to be released during the measurement. In case of air bearings the correct air pressure must be applied.

4.2.1.2 Measurement Setup

To start the measurement click the **Measure...** button (Figure 32). This opens the window with the application as shown in Figure 33.

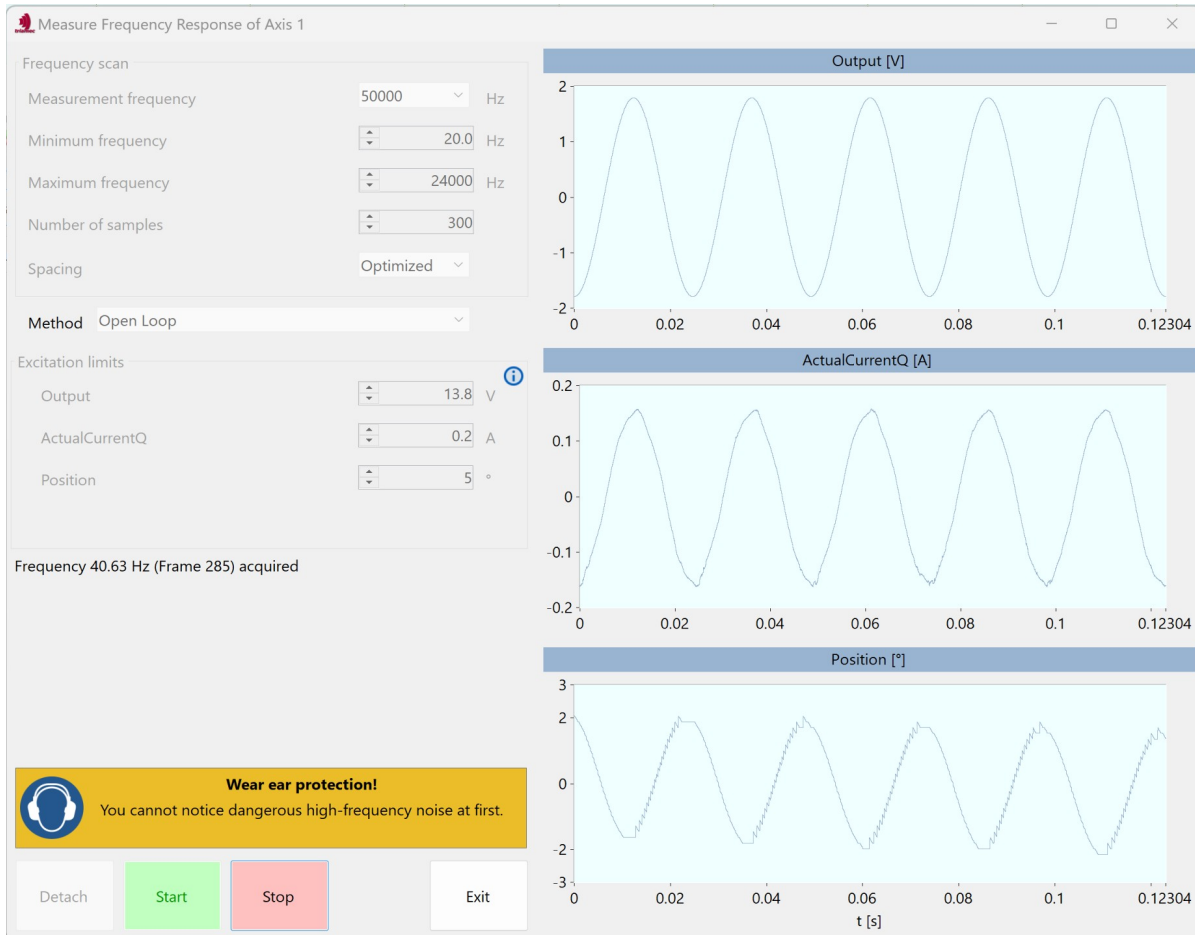


Figure 33: Frequency Response measurement application.

The following parameters can be set:

- **Frequency scan:** Set up the frequency range and the resolution of the measurement in the frequency domain.
 - ♦ **Measurement frequency:** This parameter defines the rate of data logging (default 50 kHz).
 - ♦ **Minimum Frequency:** This parameter defines the lower limit of the frequency range (default 20Hz).
 - ♦ **Maximum Frequency:** This parameter defines the upper limit of the frequency range (default 24kHz at a measurement frequency of 50 kHz). This parameter is reset to the maximum possible value when changing the measurement frequency.
 - ♦ **Number of samples:** This parameter defines the resolution of the measurement in the frequency range. Depending on the 'Spacing' parameter the range defined by 'Minimum Frequency' and 'Maximum Frequency' is divided into the number of 'Frequency Steps'. With a higher number of frequency steps, the measurement takes longer but will provide a higher reso-

lution which is especially crucial to capture sharp resonances (default 300 samples).

- ♦ **Spacing:** Defines how the range between **Minimum Frequency** and **Maximum Frequency** is sampled. **Optimized** shortens the measuring time while maintaining the quality of the measurement. Alternatives are **Logarithmic** or **Linear** spacing (default **Optimized**).
- **Method:** This parameter defines the state of the controllers during the measurement as well as the measuring points where the frequency response is measured. **Closed Loop** methods require a stable setting for the position controller and the current controller. Choose the method as called out in the setup procedure (default **Open Loop**).
- **Excitation limits:** These parameters specify the maximum allowed amplitudes for each measuring point. The names of the parameters change according to the respective measuring point. The following explanations about maximum and default values apply to each possible measuring point according to its unit. Changing the method reverts these parameters to their default value.
 - ♦ **Output:** This parameter defines the maximum allowed amplitude of the voltage signal. The default value is half of the output limit according register `Axes[].Parameters.CurrentController.OutputLimit`.
 - ♦ **ActualCurrentQ:** This parameter defines the maximum allowed amplitude of the current. The default value is half of the nominal current of the motor according register `Axes[].Parameters.Motor.NominalCurrent`. The value is limited by `Axes[].Parameters.PositionController.OutputLimit`
 - ♦ **Position:** This parameter defines the maximum allowed amplitude of the position. The default value is chosen according to the position unit.

Warning The excitation limits are used to deduce the amplitude of the harmonic excitation of the axis. This parameters must be set with caution, as too big an excitation can damage the axis. It is recommended to start with small values for the amplitudes (especially for the current) and to increase the values carefully until a sufficient signal to noise ratio is reached.

A Frequency Response measurement can generate loud noise also in frequencies that are not perceived by the human ear. Wear hearing protection and keep animal companions away.

4.2.1.3 Measurement

The following controls are available for the execution of the measurement:

- **Attach/Detach:** Take or release control of the axis.
- **Start:** When the 'Start' button is pressed, the measurement starts.
- **Stop:** The measurement can be aborted with the 'Stop' button.
- **Exit:** Close the Frequency Response measurement window.

After attaching, start the measurement by pressing **Start**. During the measurement the amplitude of the output signal is adapted in such a way that amplitudes of the states (voltage, current and position) are as big as possible but do not exceed the specified maximum.

Once the measurement is done, the **Frequency Response Measurement Result** window is opened (Figure 34) which allows to examine the quality of the signals.

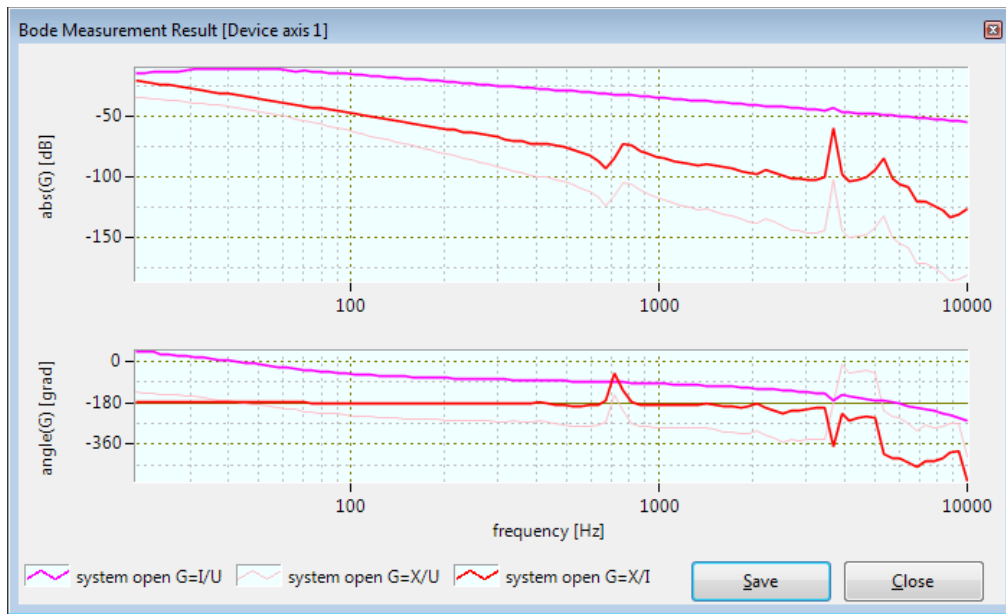


Figure 34: Frequency Response measurement result.

For later use of the measurement with the Frequency Response Tuning tool, the measurement has to be saved by clicking the **Save** button. The measurements are saved as a *.csv file. It could make sense to adjust the default file name or path for better identification of the measurement.

4.2.1.4 Remarks

For a first recording it is recommended to use the following frequencies settings:

- **Minimum Frequency** = 20 Hz; **Maximum Frequency** = 24000 Hz; Number of samples ≥ 300

In some cases it is required to have a higher resolution in the frequency domain, for example for axes with critical resonances. In this case, you can repeat the measurement in a reduced window of the frequency domain, but with a high number of samples. Multiple measurements can be shown overlaid later in Controller Tuning.

The dynamic behavior of a machine may change depending on the location of the axes. Therefore it is recommended to do the measurement at several locations in the machine workspace and save them as *.csv files. For the later tuning of the axis, multiple measurements can be loaded into the Frequency Response Tuning tool (see section 4.2.2).

Generally the quality of the measurement increases with bigger amplitudes (especially with low resolution encoders or high friction), but the restrictions of the machine must be considered when increasing the maximum values.

Normally, the phase plot of response X/I should show a phase shift of about 180° at low frequencies. If the phase shift is closer to 0° , the parameter `Axes[].Parameters.Motor.InvertDirection` might have to be altered (see also section 5.5.1).

If the excitement of the axis varies depending on the position of the motor probably one of the following parameters is not set correctly:

- `Axes[].Parameters.Motor.InvertDirection`
- `Axes[].Parameters.Motor.PolePairs`

- `Axes[].Parameters.Motor.EncoderCountsPerMotorRevolution`

In some cases it is required to do a closed loop measurement e.g. to avoid drifting of the axis. In this case the **Method** property can be set to **Closed Loop**. A closed loop Frequency Response measurement requires stable parameterization of the position controller.

When two encoders are configured for the axis (e.g. dual-loop), the signals from both encoders will be acquired.

It is possible to further customize the *Frequency Response* measurement. For example to measure cross coupling between two axis or to measure a signal from an external sensor read by an option module. This is explained in the dedicated [FrequencyResponseConfig repository at GitHub](#).

4.2.2 Controller Tuning

The saved Frequency Response measurements (see section 4.2.1) can be used to design and optimize the position and current controllers with the **Frequency Response Tuning** editor. The following steps are required to open the saved *Frequency Response* measurements in the editor:

- To open measurements in the editor click **Tune Controllers...** (Figure 32).
- Select one or several Frequency Response measurement files and click 'Open'. Use **Ctrl** or **Shift** to select more than one file. Make sure the measurement corresponds to the selected axis in the *Topology Tree*.
- The **Frequency Response Tuning** editor is opened and the frequency response of the current loop is displayed. If the plots are empty, the controller gain may be zero. Set the gain to a value > 0 in this case.

4.2.2.1 Editor Window

The editor consists of the following items (see also Figure 35):

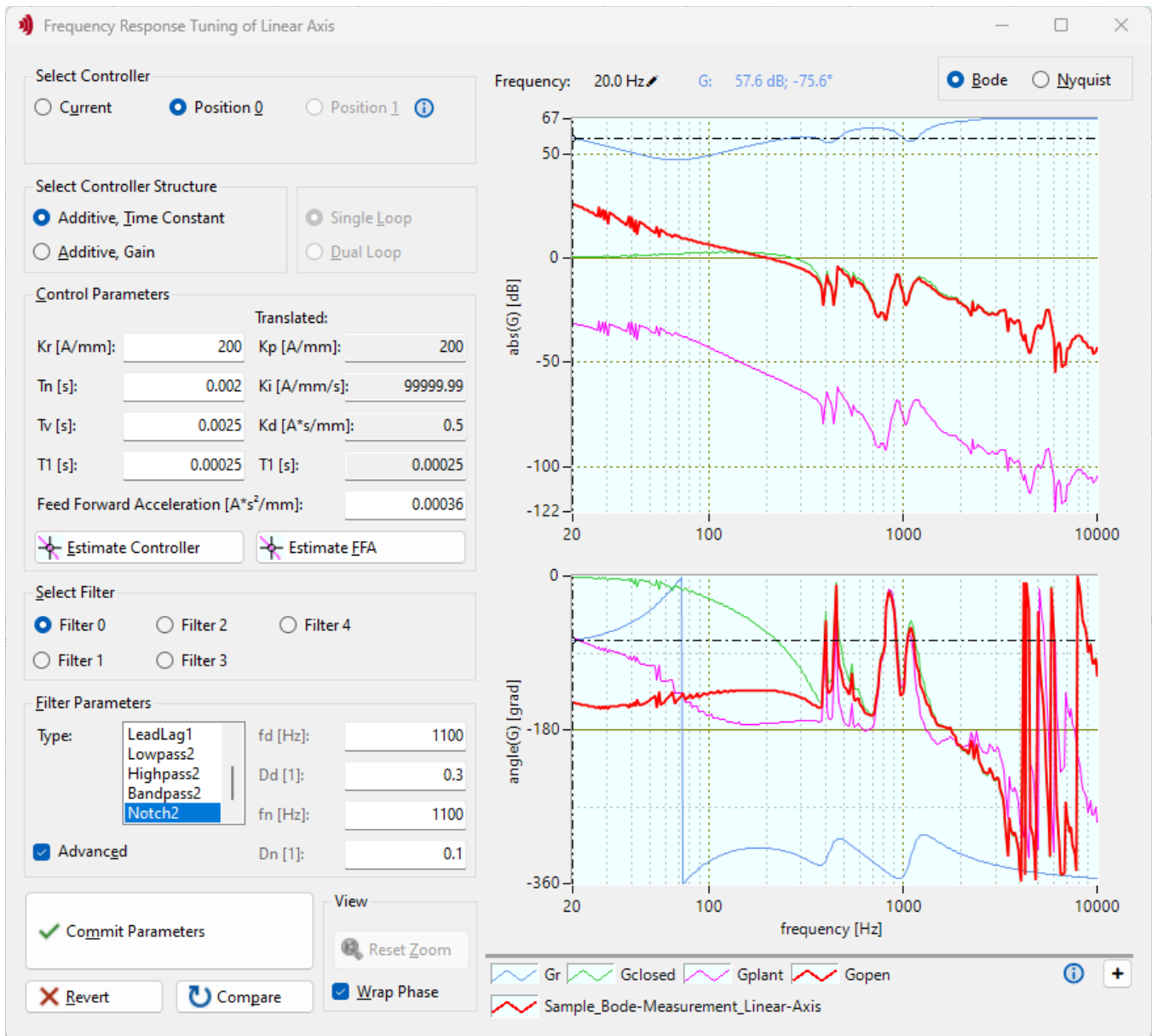


Figure 35: Frequency Response Tuning editor with current controller and Bode plot selected.

1. **Title Bar:** In the title bar, the name of the selected axis is shown. Make sure the axis is consistent with the loaded measurements.
2. **Select Controller:** This radio button can be used to switch between different controllers e.g. current loop or position loop.
3. **Control Parameters:** The parameters of the selected controller loop are displayed and can be modified. If a parameter is modified, the values are stored like prepare values in registers: The background changes its color to orange, indicating a modified but not yet committed parameter. After each change, the *Bode* and *Nyquist* Figures are recalculated. When some parameters are illegal, no data is drawn. With the button **Estimate Controller** an initial parameter set can be calculated (see section 5.4.4 and 5.4.8). The button **Estimate FFA** can be used to estimate the acceleration feed forward (see section 5.5.4).
4. **Commit Parameters, Revert, Compare:**

- ♦ **Commit Parameters:** Modification of parameters does not affect the configuration of the drive. Only when the commit button is pressed, the drive configuration will be updated with the modified values.
 - ♦ **Revert:** This button is used to reset the parameters to the actual drive-configuration.
 - ♦ **Compare:** Press and hold this button to show the transfer functions of the actual drive-configuration. This can be used to compare modified parameters with the actual drive setup.
5. **View: Reset Zoom** can be used to reset zoom and pan operations.
Checking **Wrap Phase** reduces the phase range into $[-360,0]^\circ$ in the Bode plot. It has no effect in the Nyquist plot.
 6. **Bode / Nyquist:** In the top right corner, the display mode of the frequency response can be selected:
 - ♦ **Bode:** The amplitude response and the phase response of the axis are plotted. The signal colors match to transfer functions as following (see also Figure 36).
 - Red: calculated open loop frequency response based on the measurement and the controller parameters
 - Blue: controller transfer function
 - Green: calculated closed loop frequency response based on the measurement and the controller parameters

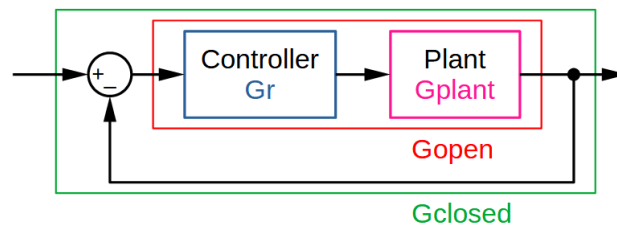


Figure 36: Signal colors according to transfer functions in the control loop.

- ♦ **Nyquist:** The red curve shows the open loop frequency response in the complex plane calculated based on the measurement and the controller parameters.
7. **Cursor:** A cursor is shown together with a label at the top left corner of the graph. The label displays the frequency, gain and phase at the cross-hair location. To select a different signal, the cross-hair has to be dragged to the new signal. At the beginning, the cursor might be hidden at the boundaries of the graph.
 8. **Legend:** The first three legend items show the colors for controller (G_r), closed (G_{closed}) and open (G_{open}) loop curves. For each loaded measurement, a legend item is generated, showing the file name and the curve colors. Hovering over the legend items will highlight G_{open} and G_{closed} loop curves from the respective measurement. If many files are loaded, the legend will wrap. In this case the size of the legend frame may be adjusted by dragging the upper border or the vertical scroll-bar at the right can be used.
You can add and remove measurements via the legend.
Remove a measurement by clicking on the **X** button which fades in when hovering over the legend item.
Add measurements using the **+** button on the right side of the legend. In the **Open Frequency Response(s)** dialog set the radio button either to **Append** in order to add the measurement to the loaded measurements, or **Replace** to only load the new measurement (Figure 37). The same dialog is shown when repeatedly using the **Tune...** button in the **Module View** tab.

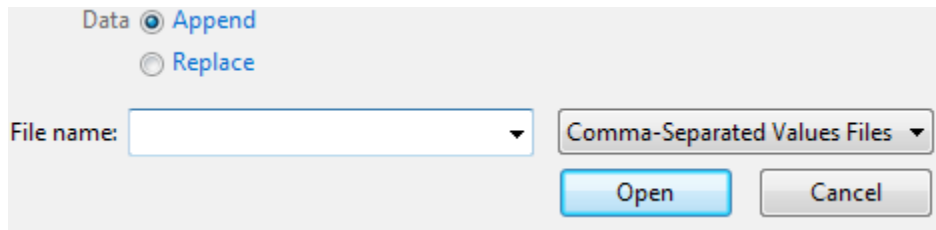


Figure 37: When opening Frequency Response measurements, you can choose whether to append new data, or to replace all measurements currently shown in the Frequency Response tuning window.

4.2.2.2 Zooming and Panning

This section describes how to zoom and pan the plot area:

- Press Shift and left mouse-key while defining the zoom area with the mouse.
- Press Ctrl and left mouse-key to pan the plot area.
- To undo the zoom area: press Shift or Ctrl and right mouse-key to reset the zoom area.
- Zooming is also possible by pressing Shift and turning the mouse-wheel.
- If zoom is applied, the cross-hair can be out of scope. Zoom out is required (e.g. Shift + scroll wheel).
- Reset zoom using the respective button.

5 Drive Configuration

This chapter explains how to configure the registers in the Topology Tree to commission a servo axis. The commissioning steps are also visualized in chapter 7.

5.1 Preparation

If the servo drive was already used for a different setup it is recommended to reset it to default settings right-clicking the device node in the register tree, then clicking – **Manage persistence...** – **Disable persistence** and rebooting the drive.

5.1.1 Required Data

Before commissioning a servo axis, axis specifications are needed. This section gives an overview of the required data. If desired, additional values can be written to the Axes[].Information registers (see also section 3.1.4).

Motor

Specifications found in the motor data sheet:

- Type of motor: dc/ac, synchronous, linear/rotational
- Peak current²
- Nominal current²
- Thermal time constant or current square time constant (or peak current duration)
- if rotational motor
 - ♦ Torque constant²
 - ♦ Number of pole pairs (check if number of pole pairs or if number of poles is specified)
 - ♦ Screw pitch if axis with lead screw
- if linear motor
 - ♦ Force constant²
 - ♦ Pole-pair pitch (distance between two north-poles or two south-poles)
- if motor temperature is observed by the drive:
 - ♦ Type of sensor
 - ♦ Temperature limit

² Important: Consider if the current is given as peak value or as root mean square value (RMS). Contact the manufacturer of the motor if this is not declared in the data sheet.

Encoder

Specifications found in the encoder data sheet:

- Encoder pitch or encoder counts per motor revolution
- Max allowed encoder speed

Additional information about the axis

The following aspects of the axis should be considered before commissioning an axis:

- How should the axis be named (e.g. Axis_X, Axis_Y ...)
- Total inertia of the axis
- Topology of the axis (linear/rotational, single-loop/dual-loop, horizontal/vertical, ...)
- How are the encoders connected to the drive (which connector of the drive is used to connect the encoders)
- How is the motor connected to the drive (which connector of the drive is used to connect the motor)
- How will the axis be referenced
- Move range of the axis
- Dynamic specifications of the axis (max velocity, max acceleration, max jerk)
- How should the axis be controlled (*TwinCAT*, stand-alone with *Tama* program, *PC-application* with *TAM API*)
- Safety considerations (*STO*, ...)
- Additional requirements ...

5.1.2 Naming of Stations and Axes

To easily identify the related physical axis, it is recommended to change the station name and the axis name to a meaningful expression. For example if Axis0 is used to drive the X axis and Axis1 is used to drive the Y axis, name the station *Station_XY* and the axes *Axis_X* and *Axis_Y*. See sections 3.1 and 3.2 for how to rename stations and axes.

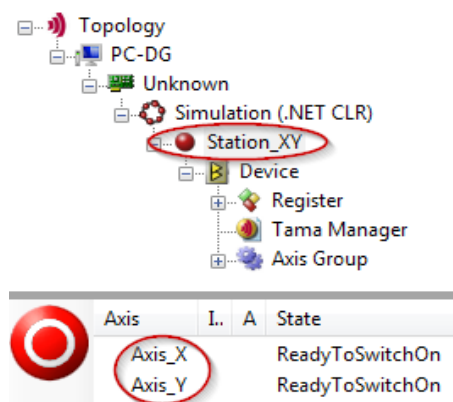


Figure 38: Example for renaming of stations and axes.

5.1.3 Units

The unit for the position can be set with register `Axes[].Parameters.PositionController.PositionUnit`. For linear axes *m* or *mm* and for rotational axes *turns*, *degree* or *rad* can be set.

Note It is recommended to set the unit for the position prior to commissioning the drive. All other dimensions are defined according the *International System of Units (SI)*.

5.2 Initial Setup of the Register Tree

This section describes the initial setup of the parameter registers. It is good practice to fill the registers from top of the tree to bottom while leaving not yet known parameters (e.g. controller gains) out for later configuration. This section follows this practice and explains how to set the relevant parameters for a common application. Parameters not covered in this section keep their default value. See section 3.1.1 on how to write and commit parameters. For people with experience in Triamec drive configuration, Figure 71 can lead you through the setup.

Remark If the drive was used for a different application before, it is recommended to reset the registers to default values (see section 3.5.1).

The register Tree contains four nodes:

- **General:** The General node contains registers which are relevant for the drive but not dedicated to an axis.
- **Axis:** The Axis node contains registers which are related to the axis.
- **Application:** These registers are only relevant in conjunction with a *Tama program* and are not covered in this section.
- **Gantry:** The Gantry node contains registers which are related to the Gantry Axes.

The following parameters must be configured for an initial setup:

5.2.1 General.Parameters

- `DcBusVoltageLowerLimit`, `DcBusVoltageUpperLimit`: These values are used to monitor the DC bus voltage (`General.Signals.DcBusVoltage`). If the measured DC bus voltage is out of the specified range, a `DcBusVoltageOutOfRange` error is issued.
It is recommended to set the range plus minus 10 to 20 % of the DC bus voltage. For applications with a TP or TPDC device the upper limit has to be set slightly above the setpoint voltage of the brake resistor. The setpoint of the brake resistor can be found in the hardware manual of the TP or TPDC device. For the TSP700 drive the brake point is set internally to 10V below the value of `DcBusVoltageUpperLimit`.
- `EncoderTopology`: The `EncoderTopology` is only different from Standard if the encoders are connected to the drive in a nonstandard way e.g. if option modules are used (see also [7]).

- **PowerSupply.SinglePhaseOperation:** With the TSP700, the monitoring of the power supply phases can be changed. If all three phases must be present, the parameter must be set to **False**. If only one or two phases are permitted, the value must be set to **True**.
- **PowerSupply.ExternalBrakeResistor:** With the TSP700, you can define whether the internal or an external braking resistor should be used. If the internal braking resistor is used, the parameter must be set to **False**. If an external resistor is used, the value must be set to **True**.

5.2.2 Axes[].Parameters.Motor

This node contains registers related to the motor.

- **Type:** Servo axes mostly use three-phase synchronous motors for both linear and rotational axes. Set Type to **SynchronousAC** in this case. In case of a DC-motor (e.g. brushed dc-motors, voice-coil motor...) set Type to **DC**. Stepper motors might have only two phases, select **SynchronousTwoPhaseAC**.
- **BrakeReleaseAction:** This parameter allows to select a digital output which is set to release the brake when the axis is enabled.
 - ♦ With 'SetOut1' the digital output O1 is set. Also the Brake output of the motor connector X40 is set.
 - ♦ With 'SetOut2' the digital output O2 is set.
 - ♦ For more information on how to connect the brake see the corresponding hardware manual.
- **BrakeHoldTime:** If there is a significant delay between the instance when the brake output is reset and the instance when brake is applied mechanically, the **BrakeHoldTime** can be used to compensate this delay. The axis will remain enabled for the duration of **BrakeHoldTime** to avoid a movement of the axis during this delay.

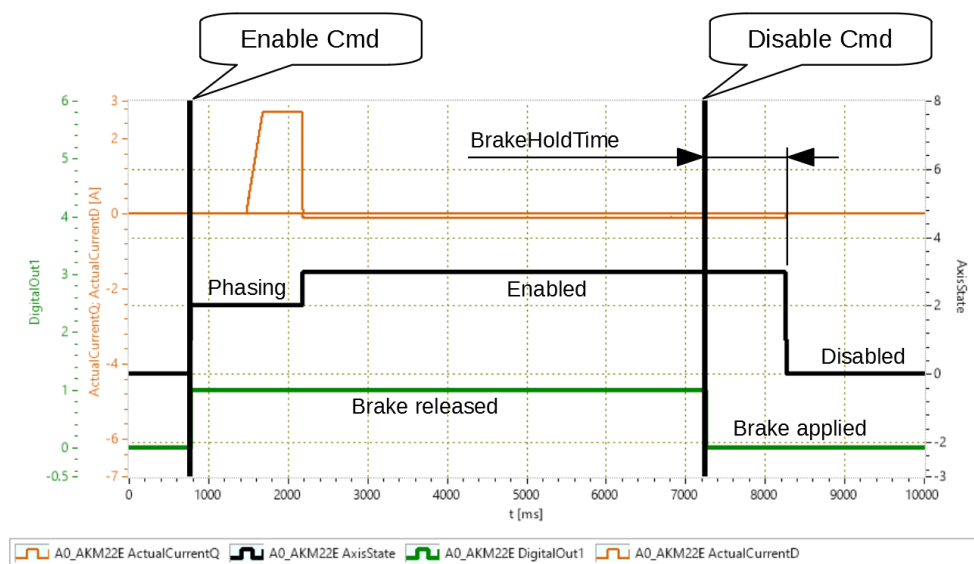


Figure 39: **BrakeHoldTime:** Timing diagram of the brake output with **BrakeReleaseAction = SetOut1** and **BrakeHoldTime = 1s**.

- **InvertDirection:** With this parameter it is possible to alter the electrical direction of the motor. The position controller requires, that the electrical direction of the motor has the same orientation as the direction of the encoder. If the orientation is opposite, the parameter **InvertDirection** has to be altered to align the directions. If the encoder direction is changed, flip the motor **InvertDirection**

too.

An incorrect setting of this parameter is often the cause for problems when the motor is enabled the first time. The following symptoms could indicate an incorrect setting which requires to alter the parameter `InvertDirection`:³

- ♦ The *Frequency Response* measurement shows a phase shift of about 0° or a multiple of 360°. A phase shift of -180° is expected if the alignment is correct.
- ♦ For a 3-phase motor, if the excitement of the axis varies depending on the position of the motor during the *Frequency Response* measurement.
- ♦ The enabling fails immediately after the phasing of the axis.
- ♦ If a current causes the axis to move in opposite direction.

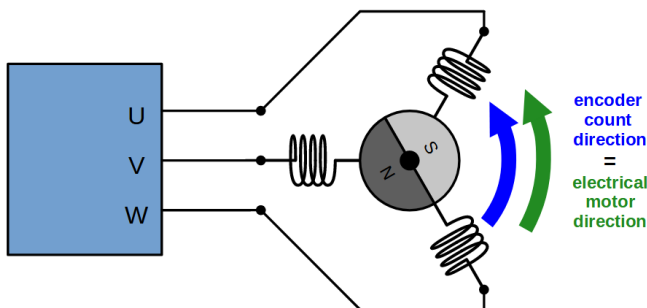


Figure 40: If the electrical direction of the motor is aligned with the encoder count direction the parameter `InvertDirection` has to be set to `False`.

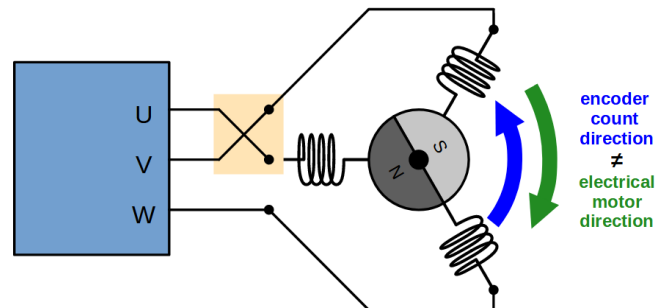


Figure 41: If the electrical direction of the motor is opposite to the encoder count direction the parameter `InvertDirection` has to be set to `True`.

- **PolePairs:** This value has to be set to the number of pole pairs according to the data-sheet of the motor. In case of a *linear motor* or a *DC-motor*, this value is set to 1. In case of stepper motors, select the number of steps and divide it by 4.
- **EncoderCountsPerMotorRevolution:** This parameter describes the number of encoder lines per motor revolution. The definition of this parameter depends on the type of motor (linear/rotational) and the type of encoder (absolute/relative). See also [7] for more information.
 - ♦ In case of a rotational motor and a relative or analog encoder⁴, `EncoderCountsPerMotorRevolution` corresponds to the number of encoder lines per motor revolution.
 - ♦ In case of a rotational motor and an absolute encoder without analog signals⁵, set `EncoderCountsPerMotorRevolution` to 1.
 - ♦ In case of a linear motor, set this parameter to the number of encoder lines per pole pair pitch: $\text{EncoderCountsPerMotorRevolution} = \text{round}(\text{PolePairPitch} / \text{EncoderPitch})$. The resulting value has to be rounded to the nearest integer value. The value for the *PolePairPitch* can be taken from the motor data sheet. The value for *EncoderPitch* is taken from the encoder data sheet (see also Figure 43).
- **PeakCurrent:** This parameter has to be set to the peak current specified in the data sheet of the motor. If the absolute current applied to the motor exceeds the value of `PeakCurrent` a `MotorPeakCurrentLimit` error is issued.

For three-phase motors, consider if the current is given as peak value or as root mean square value (RMS). If the value is specified as root mean square, multiply it by $\sqrt{2}=1.414$.

³ Similar symptoms are caused by incorrect configuration of `InvertDirection`, `EncoderCountsPerMotorRevolution` and `PolePairs` (see also section 5.5.1).

⁴ `IncrementalIRS422`; `IncrementalTTL`; `Analog`; `AnalogEndat`; `AnalogBissB`; `AnalogBissC`:

⁵ `DigitalEndat`; `DigitalBissB`; `DigitalBissC`; `DigitalTamagawa`; `DigitalNikon`

- **PeakCurrentWarning:** This parameter can be used to rise a warning if a certain current level is exceeded. The warning level is defined by a factor 0...1 relative to the value of the register **PeakCurrent**. If the factor is 1, the warning is disabled.
- **NominalCurrent:** This parameter has to be set to the nominal current specified in the data sheet of the motor. If the averaged current of one of the motor phases exceeds this value, a **MotorI2t** error is issued.
For three-phase motors, the value has to be entered as root mean square value (RMS) which is also the scale used in most data sheets.
- **NominalCurrentWarning:** This parameter can be used to rise a warning if a certain current level is exceeded. The warning level is defined by a factor 0...1 relative to the value of register **NominalCurrent**. If the factor is 1, the warning is disabled.
- **CurrentSquareTime:** This parameter is used for the calculation of the i2t current and corresponds to the *Thermal Time Constant* of the motor. The *Thermal Time Constant* should be specified in the motor data sheet. If the *Thermal Time Constant* is not available but the *Peak Current Duration* is specified, the **CurrentSquareTime** can be calculated with the following equation:

$$\text{CurrentSquareTime} = \frac{-\text{PeakCurrentDuration}}{\ln\left(1 - \frac{2 \cdot \text{NominalCurrent}^2}{\text{PeakCurrent}^2}\right)}$$

If both values aren't specified, set this value to 10-30s for an iron-less motor and 60-600s for an iron-core motor.

Warning The limitation of the current (**NominalCurrent**, **PeakCurrent**, **CurrentSquareTime**) must be configured correctly before the drive is enabled the first time. Incorrect settings could damage the motor.

5.2.3 Axes[].Parameters.PathPlanner

This node contains the parameters relevant for the internal path planner. These parameters do not influence the trajectory generation of an external control system like *TwinCAT*. Nevertheless, the parameters need to be set to reasonable values as the internal path planner is used for testing of the controller loop and also for some moves commanded by the external control system (e.g. homing moves). Moreover the setup of the modulo values need to be in accordance with the setup of the external control system (e.g. *TwinCAT*).

- **ModuloPositionMaximum** and **ModuloPositionMinimum:** If the axis does not require a modulo motion, which is the normal case if an axis is not a spindle, both values are set to zero. Modulo motion is typically required for spindles with long running velocity moves and without a restriction of the move range. For example a typical setting for the modulo range with position unit in radian is:
 - ♦ **ModuloPositionMaximum** = 6.283185307179586476925286766559 (= 2π)
 - ♦ **ModuloPositionMinimum** = 0.0

The dynamic settings of the path planner depends on the mechanical and electrical properties of the axis and on the requirements of the process. The following inequations may help to estimate reasonable upper limits for the settings:

$$u_{max} > \frac{JL}{K_t} r_{max} + \frac{JR}{K_t} a_{max} + K_v v_{max}$$

$$i_{max} > \frac{J}{K_t} a_{max} + \frac{T_{ext}}{K_t}$$

Maximal velocity	v_{max}	[rad/s]	Inductance of the motor	L	[H]
Maximal acceleration	a_{max}	[rad/s ²]	Inertia of the rotor	J	[kg m ²]
Maximal jerk	r_{max}	[rad/s ³]	Induced voltage constant	K_v	[Vs/rad]
Maximal current	i_{max}	[A]	Moment-current constant	K_t	[Nm/A]
Maximal voltage	u_{max}	[V]	External torque	T_{ext}	[Nm]
			Resistance of the motor	R	[Ω]

For DC motors the maximal voltage is given by the DC bus voltage and for AC motors by the DC bus voltage divided by square root of three. The inequations above are simplified.

- PositionMaximum and PositionMinimum: Set soft limits for the stroke of the axis. If the axis does not require soft limits, both values are set to zero. Values other than zero enable the following behavior.
 - ♦ If a move command such as *MoveAbsolute* targets a position outside the limits, the move is rejected with an error message.
 - ♦ If the axis is in coupled motion and the control system runs out of limit, the axis will throw an error at the soft limit position and trigger an emergency stop.
 - ♦ If the axis already is outside the limits, only move commands towards the legal side are allowed. The same applies to coupled motion, with the following restrictions.
 - If the position is higher than PositionMaximum, the velocity must be negative⁶.
 - If the position is smaller than PositionMinimum, the velocity must be positive⁶.
 - ♦ The software limits are only active, if homing is done!
- VelocityMaximum: This parameter specifies the max allowed velocity. The following aspects need to be considered to set this parameter:
 - ♦ design specifications e.g. max required processing speed
 - ♦ electrical and mechanical properties of the axis (see also equation above)
 - ♦ max allowed encoder speed (see encoder data sheet)
 - ♦ cut-off frequency of the encoder input (see drive hardware manual)
 - ♦ available DC bus voltage
- AccelerationMaximum, DecelerationMaximum: This parameters restrict the acceleration and deceleration during the move. In most cases the same values are used for both parameters. Acceleration and deceleration are basically restricted by the max available current (see equation above). But also the following factors need to be considered:
 - ♦ static load (e.g. vertical axis)
 - ♦ process forces and friction
 - ♦ force restrictions of mechanical elements like spindle drives.
- DecelerationEmergency: This parameter is relevant if an emergency stop is commanded. If its value is zero, the value of DecelerationMaximum is used for the calculation of the emergency stop.
- JerkMaximum: This parameter defines the maximum slope of the acceleration profile. The max pos-

⁶ The DirectCoupledMotion state expects valid triplets of Position-Velocity-Acceleration data. In the illegal case where triplets contain velocity=0, the software limit feature will not work properly.

sible jerk is basically restricted by the available voltage (see equations above). Furthermore the jerk limit is also relevant regarding how much the mechanical system is excited. A smaller jerk reduces the bandwidth of the excitement and will therefore also reduce the position error during the move. Typically the value of the jerk limit is set by factor 10 to 100 bigger than the acceleration maximum.

- **JerkEmergency:** This parameter is relevant if an emergency stop is commanded. If its value is zero, the value of **JerkMaximum** is used for the calculation of the emergency stop.
- **DynamicReductionFactor:** The dynamic reduction factor f_{drf} is a value in the range [0..1] and can be used to reduce the path dynamics. The factor scales the limits accordingly the following rules:

$$v_{red} = v_{max} f_{drf}; \quad a_{red} = a_{max} f_{drf}^2; \quad r_{red} = r_{max} f_{drf}^3$$

5.2.4 Axes[].Parameters.PositionController

The position controller node contains registers to configure the encoder, the position controller and the feed forward. For a normal axis only **Encoder[0]** and **Controller[0]** needs to be configured. **Encoder[1]** and **Controller[1]** are used if an axis provides a second encoder, e.g. for dual loop-control.

- **PositionUnit:** This parameter is used to define the unit of the positions and it ensures that position units in *TAM System Explorer* are displayed correctly.
With *EtherCAT* this parameter is also used to set the scaling of the set-points received from the control system. See [2] for more information.
- **MasterPositionSource:** With this parameters the source for **Signal.PositionController.MasterPosition** can be defined.

The following parameters can be used to configure the feed forward. Figure 54 shows the diagram of the feed forward block.

- **FeedForwardPosition:** This parameter compensates a force which is proportional to the position, e.g. a spring. For an initial setup this parameter is set to zero in most cases.
- **FeedForwardVelocity:** This parameter compensates a viscous friction. As the viscous friction is not known in advance, this parameter is set to zero for the initial setup.
- **FeedForwardAcceleration:** This parameter compensates the inertia of the axis. This is an important parameter regarding the dynamic performance of the axis. Basically, if the inertia of the axis and the torque constant of the motor are known, the parameter can be set to:

$$FeedForwardAcceleration = \frac{Inertia}{TorqueConstant}$$

But in general it is recommended to set this parameter to zero for the initial setup. The parameter can then be set and optimized later based on measurements as described in section 5.4.10.

- **FeedForwardCoulombCurrent:** This parameter compensates Coulomb friction. Initially this parameter should be set to zero. If required, the parameter can then be set later based on measurements.

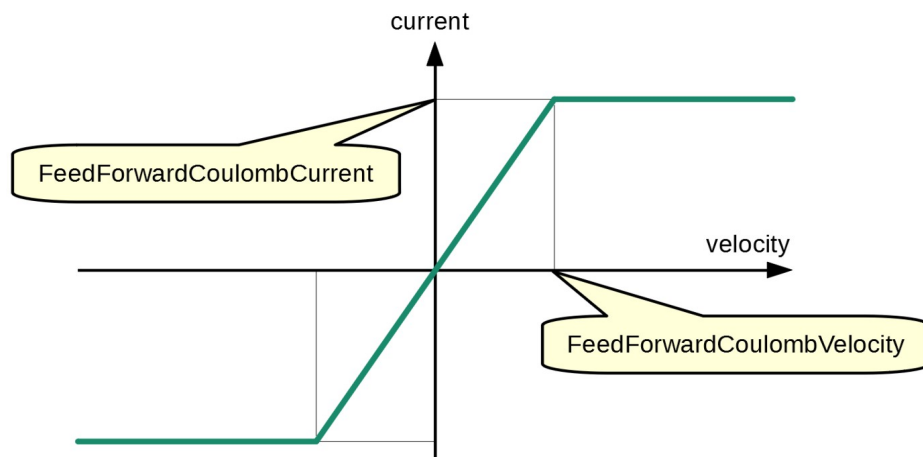


Figure 42: Coulomb feed forward.

- **FeedForwardCoulombVelocity:** With this parameter the transition range of the coulomb feed forward can be set. This allows to smooth the transition of the current when the direction of the movement is changed.
- **FeedForwardFilter:** By default the Type of the filter should be set to bypassed.
- **OutputLimit:** This parameter restricts the output of the position controller. Typically this parameter is about 5-10% below the peak current of the motor or the drive, depending on which is smaller.

5.2.5 Axes[].Parameters.PositionController.Encoders

The encoder node contains registers to configure Encoder[0] and Encoder[1]. For an axis with only a single encoder only Encoder[0] has to be configured. Encoder[1] is used, if an axis provides a second encoder signal, e.g. for dual loop-control.

- **Type:** This parameter is set to the encoder type connected to the related connector. By default this is connector X20 for Axes[0] and X21 for Axes[1]. Set Type to Analog if the position information is provided by a pair of analog sine and cosine signals with an amplitude of 1Vpp. See [7] for information about other supported encoder types. If no encoder is attached to the connector, the parameter has to be set to None to avoid an EncoderError.
- **InvertDirection:** This parameter is used to define the positive direction of the position. Set this parameter to True, if the positive direction of the encoder position is opposite to the positive direction of the physical axis. In case this parameter is changed on an already commissioned axis, change the parameter Motor.InvertDirection too.
- **Pitch:** This parameter defines the scale of the position. If Type is Analog or Incremental, the value of Pitch is the distance between two encoder lines. Examples:
 - ♦ For a rotational encoder with 2048 lines and with the position unit in degree, the Pitch is $360^\circ / 2048 = 0.17578125^\circ$.
 - ♦ For a linear encoder the value for Pitch is taken from the data sheet of the encoder, e.g. 0.02mm for an encoder with a pitch of 20um and position units of "mm".
 - ♦ If an encoder with digital protocol is used refer to [7] on how to set the Pitch parameter.

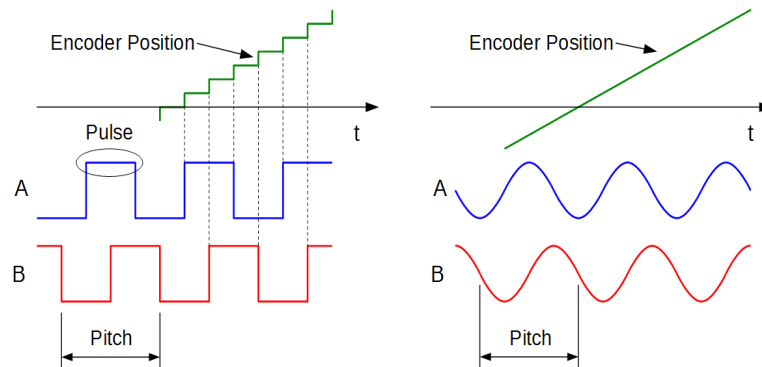


Figure 43: The pitch is displayed for incremental encoders (left) and analog encoders (right). The encoder pitch corresponds to the distance between the encoder lines or to one signal period.

- **VelocityFilterT1:** With this parameter the bandwidth of `PositionController.Encoders[].Velocity` can be defined. The value corresponds to the time constant of the filter.

Backlash compensation can be applied if an axis shows a directional contouring error during slow motion. This can for example be caused by the backlash of a spindle drive.

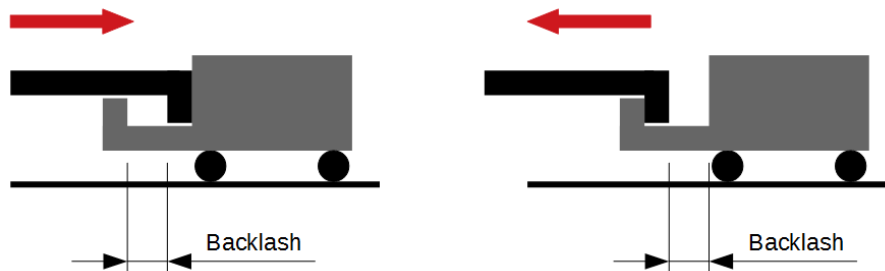


Figure 44: Backlash compensation.

- **BacklashDistance:** The value of `BacklashDistance` has to be set equal to the value of the contouring error. For dual-loop axes, the backlash has to be configured for the encoder of the inner controller loop (`Encoder[0]` in most cases). To align the backlash compensation with the axis reference, the direction of the reference move can be considered for the compensation by setting the sign of the parameter:
 - ♦ Use a positive sign, if the reference move is done in positive direction. In this case the correction is applied when the axis is moving in negative direction.
 - ♦ Use a negative sign, if the reference move is done in negative direction. In this case the correction is applied when the axis is moving in positive direction.
- **BacklashDuration:** The value of `Parameters.PositionController.Encoders[].BacklashDuration` can be used to configure the smoothing of the backlash compensation. If backlash compensation is applied, a change of direction causes a position shift of the actual position. To smoothing this shift, a second order low pass filter is applied. The dynamics of this filter can be configured with the parameter `BacklashDuration`. Typical values are in range of 0.1 to 0.5s.
 - ♦ A small value causes a fast shift of the position.
 - ♦ A big value causes a slow shift of the position.

Note Backlash Compensation is suitable to compensate contouring errors during slow motion. For high dynamic moves the application of backlash compensation can be counterproductive as the backlash error not only depends on the move direction but also on the acceleration and deceleration.

5.2.6 Axes[].Parameters.PositionController.Controllers

The controller node contains registers to configure Controller[0] and Controller[1]. The structure of the controller is described in section 5.4. For an axis with a single encoder only Controller[0] has to be configured. Controller[1] is only used if an axis provides a second encoder signal e.g. for dual loop-control.

- Kp, Ki, Kd, T1: These registers are used to configure the PIDT1 position controller. For the initial configuration, these values can be set to zero. The setup and optimization of the position controller is described in section 5.4.5.
- PositionErrorLimit: This parameter defines the maximum allowed position error (position lag). If the position error exceeds this limit, the axis is disabled and a PositionErrorLimit error is issued. This is an important parameter which helps to protect the machine in case of a malfunction. Set this parameter as small as possible without causing faulty activation of the error. Typical values are 0.1mm to 0.5 mm for linear axes and 0.1 degree to 1.0 degree for rotational axes.
- PositionErrorWarning: This parameter can be used to rise a warning if a certain position error is exceeded. The warning level is defined by a factor 0...1 relative to the value of register PositionError. If the factor is 1, the warning is disabled.
- IntegratorOutputLimit: This parameter restricts the current of the integral term. It is recommended to set this value to about 40% to 80% of the nominal current.
- Filters: Filters can be used to optimize the stability of the controller loop. See section 5.4.9 for more information. For the initial setup the Type of the filters should be set to bypassed.

Warning PositionErrorLimit must be configured correctly before the drive is enabled the first time to protect the axis.

5.2.7 Axes[].Parameters.Commutation

Commutation is the process of applying current to the phases in the right order to generate motion. For brushed DC-Motors, commutation is ensured mechanically by contacting the correct winding with the brush. For synchronous AC motors, commutation has to be performed electronically, in most cases based on the encoder position.

For electronic commutation it is crucial to initially determine the orientation between the electrical phases (stator) and the magnetic poles (rotor). The process to find this orientation is called *phasing*. With the following parameters, the phasing process can be configured:

- PhasingMethod: This parameter selects the phasing method with the following choices (see also Figure 45):
 - ♦ NoPhasing: This method is used if Axes[].Motor.Type = DC or if a motor with an absolute encoder is already initialized with the correct phasing angle (see also section 6.1 in case the commutation information of an absolute encoder should be used).
 - ♦ RotorAlignment: The rotor is aligned by applying a current to a specific phase angle. This causes

the rotor to align with the magnetic field of the phases. If the rotor is not already aligned by chance, this causes a movement of the rotor. This method must be used for the initial phasing during commissioning of synchronous AC-motors as it does not require a tuned position controller.

- ◆ AngleSearch: With this method, the electrical direction of the current is controlled in such a way, that the motor does not move during angle search. This method requires a stable setup of the position controller. It is recommended to use this method for axes which must not move during phasing. As AngleSearch requires a tuned position controller it is not suitable for the initial setup.

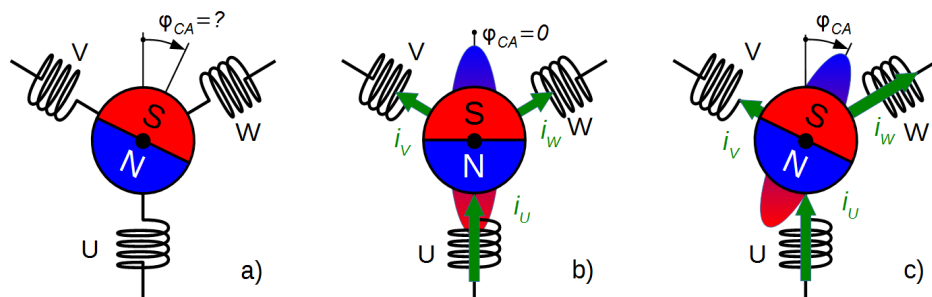


Figure 45: Phasing Method: a) Before phasing the commutation angle is unknown. b) RotorAlignment: The rotor is aligned with a static electromagnetic field generated by the phase currents. c) AngleSearch: The angle of the electromagnetic field is adjusted to match with the magnetic field of the rotor without moving it.

- ◆ AngleSearch2: This method is an improved version of the AngleSearch method that allows the motor to be in motion when the phasing algorithm starts. It is recommended to use this method for axes with low damping (e.g. axes with air bearings). From the start of the procedure until RampRiseTime is reached, the velocity of the axis is kept constant. Then, the axis is decelerated to zero velocity by the time half of the RampConstTime is reached. All this happens simultaneously while the algorithm determines the phasing angle. As this method uses velocity estimation, the VelocityFilterT1 of the master encoder must be configured to an appropriate value, especially when using serial encoders. Also, AngleSearch2 requires a tuned position controller and is not suitable for an initial setup.

Note The method AngleSearch2 is available with firmware version $\geq 4.24.0$ and TAM System Explorer version $\geq 7.29.0$.

- EnablingMethod: This parameter specifies if phasing is executed when the motor is enabled. In case of an absolute encoder the parameter is used to specify the source for the commutation offset. The following options can be selected:
 - ◆ ForcePhasing: Phasing is executed at every enabling of the motor.
 - ◆ Automatic: Phasing is executed only if Axes[.].Signals.Commutation.State is Invalid. This could for example be the case after a power-cycle of the drive. The commutation state will change to Valid when phasing is executed successfully. This method should not be used before the commissioning of the current controller and the position controller is successfully done.
 - ◆ NoPhasing: This option is used if Axes[.].Motor.Type = DC.
 - ◆ AbsoluteEncoder; AbsoluteEncoderOffsetEncoder; AbsoluteEncoderOffsetDrive: See section 6.1 for how to setup the commutation based on absolute encoders.

For an initial setup it is recommended to use ForcePhasing. Depending on the application the setting can be changed for example to Automatic after the commissioning is successfully done.

During phasing, a current is applied. The shape of this current can be configured with the following parameters (see also Figure 46).

- Angle: This is the electrical angle of the phasing vector used in the RotorAlignment method. Default value is zero.
- StartTime: This parameter delays the start of the phasing. During this period, the voltage controller of the drive is already active and causes damping of the motor because of the back electromotive force. A StartTime greater than zero might be useful in the following cases:
 - ♦ In case a brake with significant delay has to be released before the phasing starts.
 - ♦ In case the phasing fails because of ongoing movements of the axis. Increasing the StartTime reduces the motion of the axis. This is a typical use case for axes with low friction e.g. axes with air-bearingins.
- RampRiseTime: This parameter is used to configure the duration of the current ramp. The recommended setting for this parameter depends on the PhasingMethod:
 - ♦ RotorAlignment: RampRiseTime typically 1.0-2.0s
 - ♦ AngleSearch and AngleSearch2: RampRiseTime typically 0.1-0.2s
- RampConstTime: This parameter should be set to a value which allows the axis to reach steady state. Depending on the type of axis this is typically 0.5-2.0s. The value has to be increased, if the axis is still in motion at the end of the phasing.
- Current Amplitude: This parameter defines the amplitude of the applied current. It is typically set to 50-100% of the nominal current.

It is possible that the initial position of the axis is exactly opposite to the applied current-angle. In this case nearly no torque is applied to the motor and phasing may fail. To prevent this case, the angle of the applied current is oscillated three times at the beginning of the phasing. This should get the axis out of its equilibrium. The following parameters can be used to configure the oscillation.

- SineAmplitude: Amplitude of the angular oscillation of the commutation angle. Typical values are 0.1-0.5rad.
- SineFrequency: Frequency of the angular oscillation of the commutation angle. Typical value is 10Hz. Ensure that: $3 / \text{SineFrequency} < 0.8 \cdot (\text{RampRiseTime} + \text{RampConstTime})$.

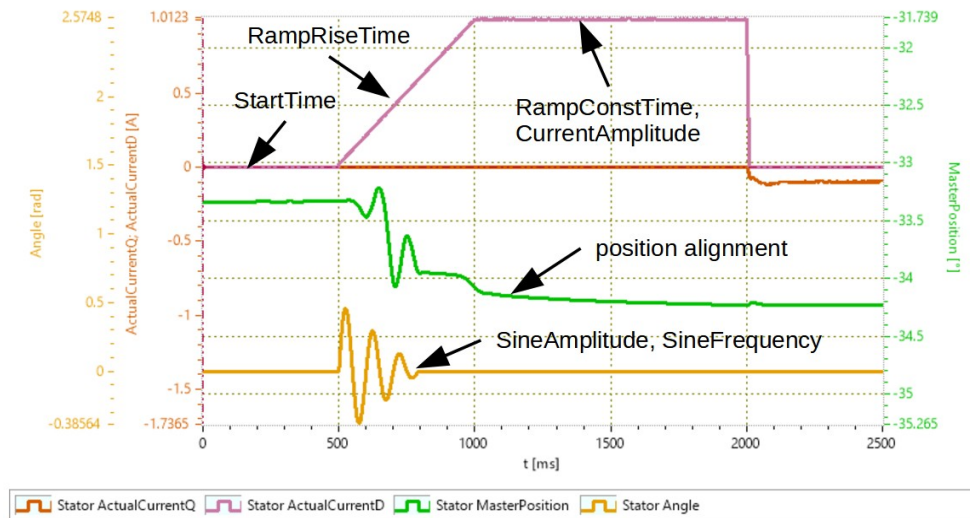


Figure 46: Phasing sequence with Type = RotorAlignment.

5.2.8 Axes[].Parameters.CurrentController

This section describes the initial setup of the current controller. The structure of the controller is described in section 5.4.

- **PwmFrequency:** This parameter is used to define the frequency of the pulse width modulation. In most cases this parameter should be set to `pwm50kHz`, as this allows higher drive currents. Only for motors with very low impedance (e.g. spindles) `pwm100kHz` can be considered to reduce the heating of the motor.
- **Kr, Tn:** These registers are used to configure the PI current controller. For initial configuration these values can be set to zero. The setup and optimization of the current controller is described in section 5.4.1.
- **IntegratorOutputLimit:** This parameter can be used to restrict the output of the integral term to avoid a windup behavior of the controller. Typically this value is set to 0.5..1.0 of the `CurrentController.OutputLimit` (see below)
- **R, L, Kt:** These are the voltage feed forward parameters with R representing the stator resistance (half of phase to phase resistance), L representing the stator inductance (half of the phase to phase inductance) and Kt representing the voltage constant of the motor.
Remark: These parameters are also required for sensorless mode [8], in most other cases it is recommended to set these parameters to zero as voltage feed forward is not required due to the high bandwidth of the current controller.
- **FeedForwardLimit:** Set this parameter to limit the feed forward voltage calculated by the R, L and Kt parameters. If `FeedForwardLimit` is set to zero, the feed forward path is disabled.
- **OutputLimit:** This parameter restricts the maximum output voltage over one phase. Typically this parameter is set to the maximum possible voltage which depends on the available DC-bus voltage and the `Motor.Type`:
 - ♦ **SynchronousAC:** The maximum setting is 0.577 times the dc-bus voltage.
 - ♦ **DC:** In case of a DC-motor the `OutputLimit` can be set equal to the DC-bus voltage.

- Filter: The filter can be used to shape the output of the controller loop. See section 5.4.9 for more information. For the initial setup the Type of the filters should be set to bypassed.

5.2.9 Axes[].Parameters.CurrentController.FieldWeakening

Field weakening is a method for increasing the speed of electrical motors if the DC bus voltage is the restricting factor. This is accompanied by a reduction of the maximum torque of the motor. Field weakening with synchronous AC motors and Field Oriented Control is achieved by inducing a negative i_d current depending on the speed of the motor. It is mainly applied to high speed spindles, if the desired speed is not reached.

- VelocityStart: This parameter determines the velocity above which the field weakening is applied.
- Gain: With the Gain the induced current $i_{d_{fw}}$ is calculated depending on the velocity according to the following equation. To induce a negative $i_{d_{fw}}$ the Gain has to be smaller than zero. Set the Gain to zero to disable the field weakening.

$$i_{d_{fw}} = \begin{cases} \text{Gain} \cdot (|\text{velocity}| - \text{VelocityStart}) & \text{if } |\text{velocity}| > \text{VelocityStart} \\ 0 & \text{else} \end{cases}$$

5.2.10 Axes[].Parameters.CurrentController.FeedForwardFilter

By default and for a first setup the Type of the filter should be set to bypassed to deactivate it. Refer to Figure 49 for more information.

5.2.11 Axes[].Parameters.Homing

This node can be used to configure a homing sequence executed by the drive. More information about how to setup the homing of an axis is documented in [10].

- Method: For the initial setup of the axis no homing is required and the homing method is set to None.

5.3 Verification of the Encoder

With this test the configured encoder pitch and the encoder direction are verified. For the test, move the axis by hand for a certain distance e.g. for one turn and observe the position in the *TAM System Explorer*. To observe the position the following options are available.

1. Select the associated axis module in the *Topology Tree* and observe the position displayed at the upper right corner of the axis module (see Pos 5 in Figure 31).
2. Observe the value of the register Axes[].Signals.PositionController.Encoders[].Position in the **Registers** tab of the *Tab Panel*.
3. Drag the register Axes[].Signals.PositionController.Encoders[].Position into the *Scope Area* and start the scope.

Check the counting direction and if the relative distance of the displayed position corresponds to the

executed move. If there is a mismatch, adjust the following parameters and execute the test again.

- `Axes[].Parameters.PositionController.Encoders[].InvertDirection`
- `Axes[].Parameters.PositionController.Encoders[].Pitch`

In case the position signal shows no movement at all, the connection of the encoder has to be verified. Especially if an option module is used also check the setting of `General.Parameters.EncoderTopology` (see [7] for more information).

5.4 Controller Tuning

After initial configuration of the parameters, the controllers are ready to be tuned. For the tuning of the controllers the *Frequency Response* module provided by *TAM System Explorer* is used (see chapter 4). The following sections show the structure of the controller and explain how to execute the *Frequency Response Tuning*. For people with experience in Triamec drive configuration, Figure 72 can lead you through the tuning procedure.

Figure 47 shows a simplified block-diagram of the implemented controller structure. The controller structure supports cascaded control with a *PI* controller for the current and two *PIDT1* controllers for the position.

For an axis with one encoder, only `Controller[0]` is used. If a second encoder is installed, use `Controller[1]` to setup a dual loop controller. Both, the position controller and the current controller support feed forward to set the current and the voltage based on the setpoint trajectory.

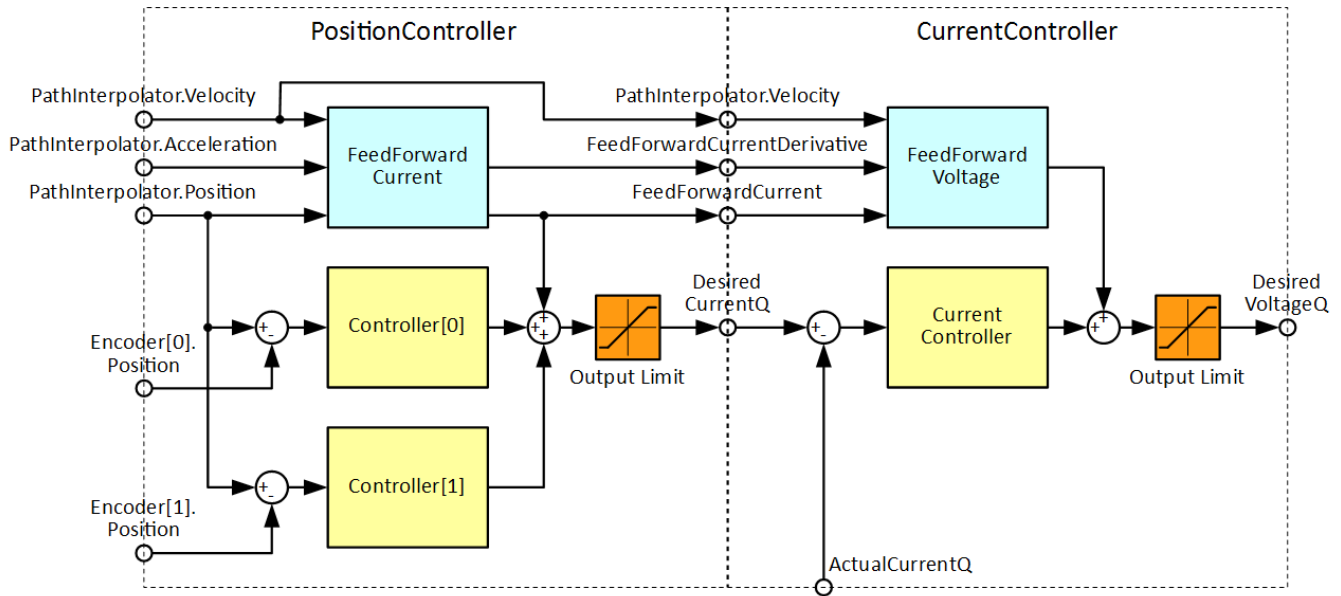


Figure 47: Block-diagram of the controller structure with current controller and position controller (simplified).

5.4.1 Current Controller Structure

The controller structure consists of a PI-controller with proportional gain K_r and integral time constant T_n . Additionally, the integral term can be restricted to prevent integral wind-up. At the output a filter of second order can be configured.

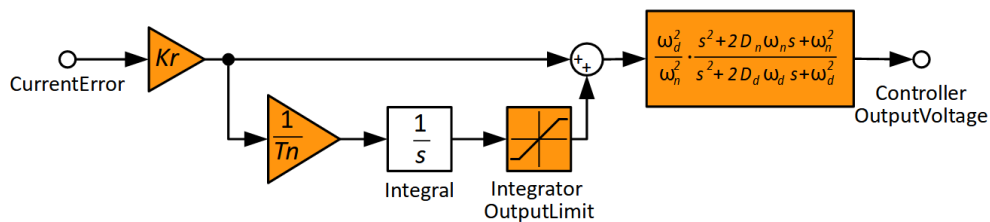


Figure 48: Block-diagram of the PI current controller (simplified).

In some cases the dynamic of the current loop can be improved by applying voltage feed forward. Figure 49 shows the block-diagram for voltage feed forward.

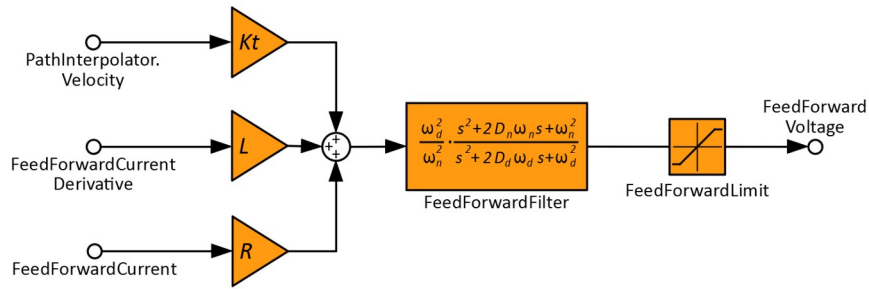


Figure 49: Block-diagram of the voltage feed forward.

5.4.2 Tuning of the Current Controller

This section describes how the proportional gain K_r and the integral time constant T_n for the current controller can be determined with *Frequency Response Tuning*. See section 5.2 on how to initially set up the other parameters in the current controller structure. The following steps are required to start the tuning:

1. Make sure, that the controller gain K_r is set to 0 (this step is not required if the current controller is already configured with a stable setting).
2. Execute a Frequency Response measurement and save the measured data. See section 4.2.1 on how to do the measurement.
3. Open the measured data with the **Frequency Response Tuning** application. See section 4.2.2 on how to open the measurement and how to use the application.
4. To setup the current controller, select the **Current** radio button in the **Select Controller** panel.
5. Initially, set the controller gain K_r to 1 (if K_r is zero, no transfer functions are plotted).
6. Start the tuning based on the criteria below. Consider also the “rule of thumb” (see section 5.4.3) or use the Estimate Controller button (see section 5.4.4).

The controller parameters K_r and T_n can now be modified. After each change of a parameter the *Bode* or *Nyquist* plots are updated. The *Bode* plot shows the amplitude and the phase of the following complex transfer functions:

$$H_r(f) = K_r \left(1 + \frac{1}{s T_n} \right); \quad H_o(f) = H_r(f) \frac{i}{u}(f); \quad H_c(f) = \frac{H_o(f)}{1 + H_o(f)}; \quad s = j 2 \pi f$$

with

- f : frequency
- $H_r(f)$: current controller transfer function
- $H_o(f)$: open loop transfer function
- $H_c(f)$: closed loop transfer function
- $i/u(f)$: measured frequency response function between voltage and current.

The *Nyquist* plot shows the open loop transfer function $H_o(f)$ in the complex area.

The goal for the tuning is to get a stable controller with the following characteristics (see also Figure 50 and Figure 51):

- a) wide bandwidth of $H_c(f)$
- b) high amplification in the pass-band of $H_o(f)$
- c) sufficient attenuation in the stop-band of $H_o(f)$

Therefore, K_r should be increased while T_n is reduced as close to zero as possible by considering the following criteria. See Figure 61 on how the parameters influence the curve.

Tuning criteria in **Bode** plot view:

- d) The phase-margin of $H_o(f)$ should be in range of 50° to 70° .⁷
- e) The amplitude of the $H_c(f)$ should not exceed 1dB over the whole frequency range.

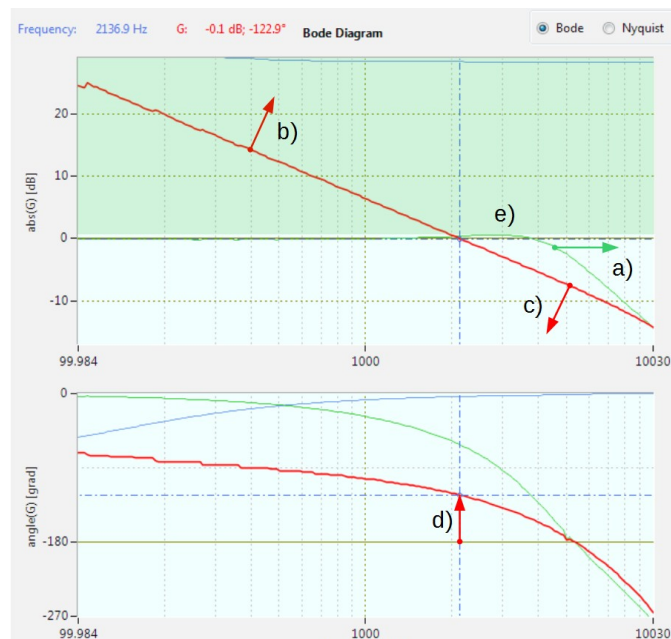


Figure 50: Tuning criteria in Bode view for the current controller.

Tuning criteria in **Nyquist** plot view (Figure 51):

- f) The displayed complex curve of $H_o(f)$ must be outside of the 1.1 circle but as close to it as possible.
- g) Following the complex curve $H_o(f)$ from low frequency to high frequency, the critical point $[-1, 0j]$ must be located in the region to the left of the curve.

⁷ The phase-margin is the difference between the phase-angle of a transfer function $G_o(f_{gc})$ at the frequency f_{gc} where the amplitude of $H_o(f_{gc})$ crosses 0dB and -180° . See d) in Figure 50.

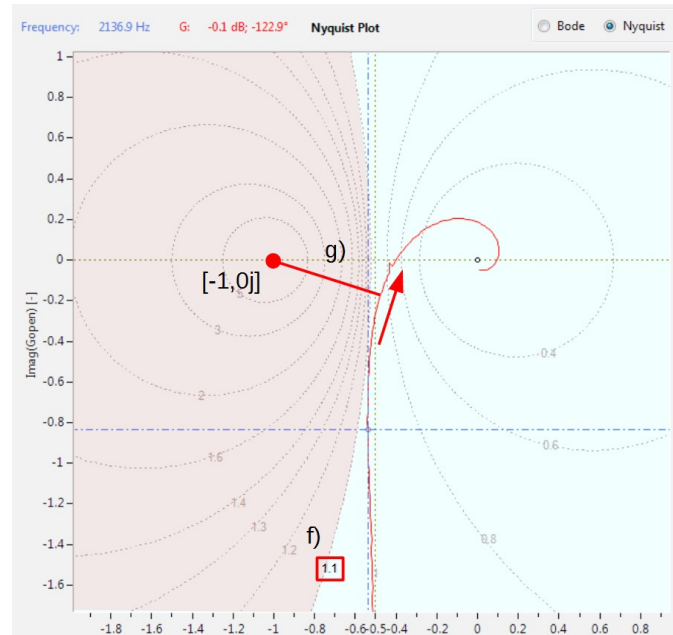


Figure 51: Tuning criteria in Nyquist view for the current controller.

5.4.3 Rule of Thumb for Current Controller Tuning

This section describes a rule of thumb for an initial setup of the current controller. See also the **Estimate Controller** section 5.4.4.

1. Set $K_r = 1$ and $T_n = 0$, and activate the **Bode** plot view.
2. With the cross-hair search the frequency f_{gc} where $H_o(f_{gc})$ has a phase margin of 60° (Figure 50, marker d)).
3. Read the amplitude A_{gc} at frequency f_{gc} in dB .
4. An initial value for the gain and the time constant can now be calculated with the following equations (see also Figure 52):

$$K_r = 10^{-\frac{A_{gc}}{20}} \quad T_n = \frac{10}{2\pi f_{gc}}$$

5. Use the criteria above to further optimize the current controller.
6. Press the **Commit** button to activate the parameter changes on the drive.

5.4.4 'Estimate Controller'-Button (Current Controller)

The **Estimate Controller** button can be used to apply an initial tuning of the current controller. The applied tuning is based on the rule of thumb described in section 5.4.3. Follow the following steps to use the **Estimate Controller** button for an initial tuning:

1. In the Bode View place the cross hair at the frequency f_{gc} where the angle of the open loop transfer function $H_o(f_{gc})$ (red line) has the desired phase margin e.g. 60° (see Figure 50, marker d)).
2. Press the **Estimate Controller** button. With this, an initial controller setting is calculated.

3. Use the criteria in section 5.4.2 to further optimize the current controller.
4. Press the **Commit** button to activate the parameter changes on the drive.

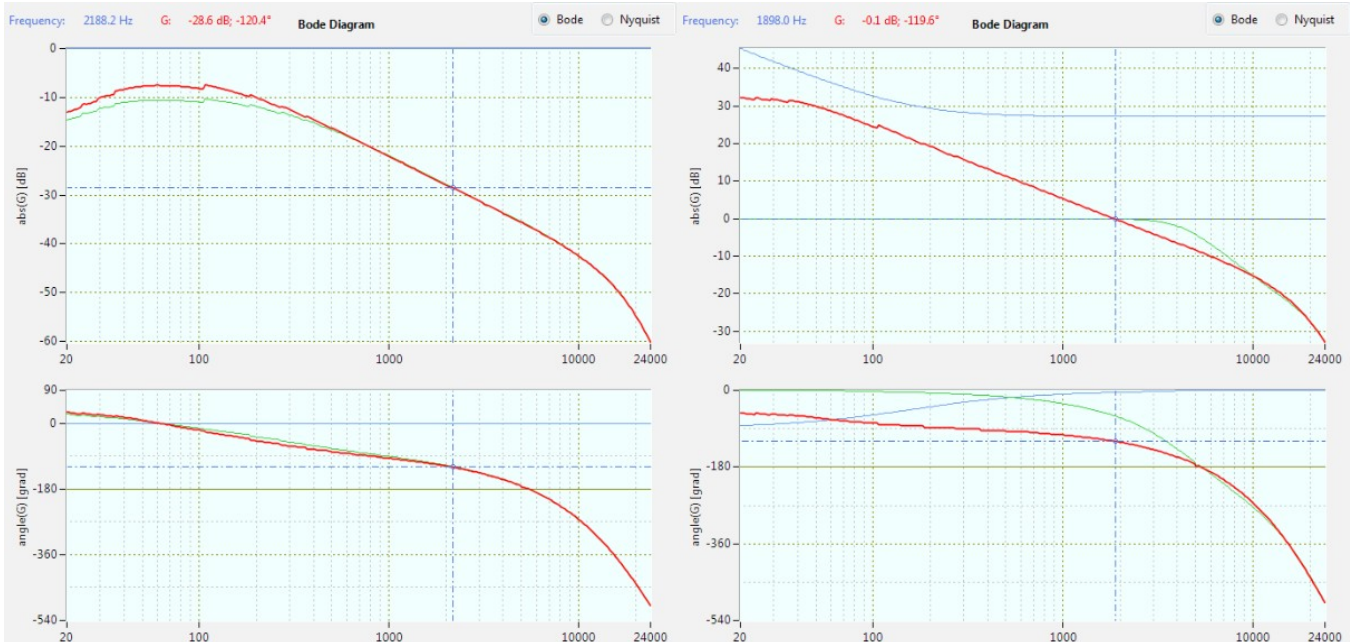


Figure 52: Tuning of the current controller. The figure to the left shows the bode plot with $K_r = 1$ and $T_n = 0$ a phase margin of 60° (angle = -120°) is reached at 2188Hz with a magnitude of -28.6dB. Based on the rule of thumb this would result in $K_r = 27$ and $T_n = 0.0007$. The figure to the right shows the further optimized transfer function with $K_r = 23$ and $T_n = 0.0001$.

5.4.5 Position Controller Structure

Figure 53 shows the controller structure of the position controller. It consists of a PIDT1-controller with proportional gain K_p , integral gain K_i and derivative gain K_d . The bandwidth of the derivative term can be restricted with the low pass time constant T_1 . With the IntegratorOutputLimit wind-up of the integral can be avoided. Additionally five second order filters can be configured.

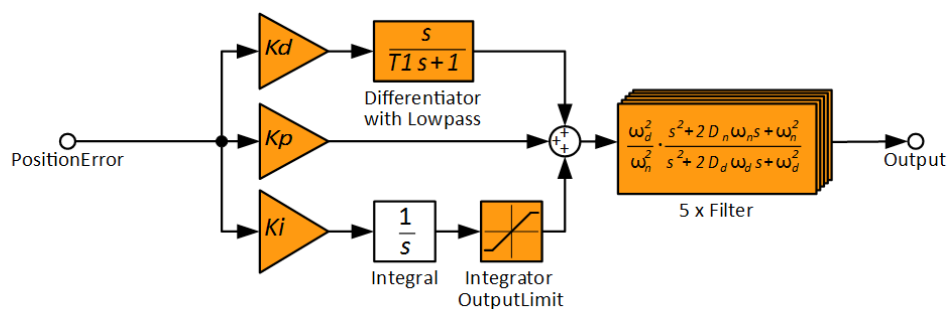


Figure 53: Block-diagram of the PIDT1 position controller.

Figure 57 shows the block diagram for current feed forward. Beside the feed forward of position, velocity and acceleration also coulomb friction can be compensated. The FeedForwardFilter can be used to

fine tune the feed-forward transfer function.

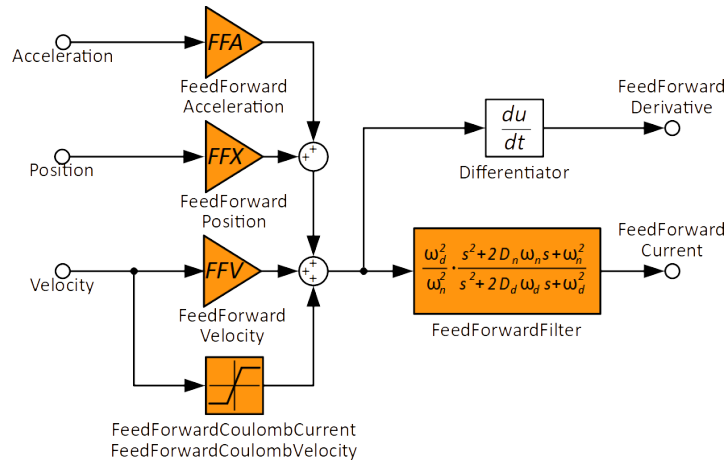


Figure 54: Block-diagram of the current feed forward.

5.4.6 Tuning of a Single-Loop Position Controller

Important The current controller (inner loop) has to be setup before the position controller (outer loop). This is because the parameterization of the current controller influences the characteristics of the position controller. The current controller in contrast is independent of the parameterization of the position controller.

This section describes the tuning of a single loop position controller⁸. With single-loop setup, only one position feedback is used for control.

- If the loaded *Frequency Response* measurement contains only one encoder signal, this signal is automatically used for the tuning. The radio button **Position 1** is grayed out and **Single Loop** is set permanently.
- If the loaded *Frequency Response* measurement contains signals from two encoders, **Single Loop** is set by default and should only be changed for dual-loop control. The signal used for the tuning can be selected as follows:
 - ♦ With **Position 0** latched, the signal from Encoder[0] is used.
 - ♦ With **Position 1** latched, the signal from Encoder[1] is used.

This section considers only the first case, as this is the default case for single-loop position controllers.

Important For synchronous AC motors the measurement of the transfer function between current and position requires a valid commutation. If the commutation angle is not provided by an absolute encoder, the initial (first time) commissioning of the position controller requires to set

Parameters.Commutation.PhasingMethod to RotorAlignment and
Parameters.Commutation.EnablingMethod must *not* be Automatic!

Important The *Frequency Response* measurement has to be executed again after the first commissioning of the current controller (see also 4.2.1.4)!

To setup the position controller, **Position 0** has to be selected in the **Select Controller** panel. The **Con-**

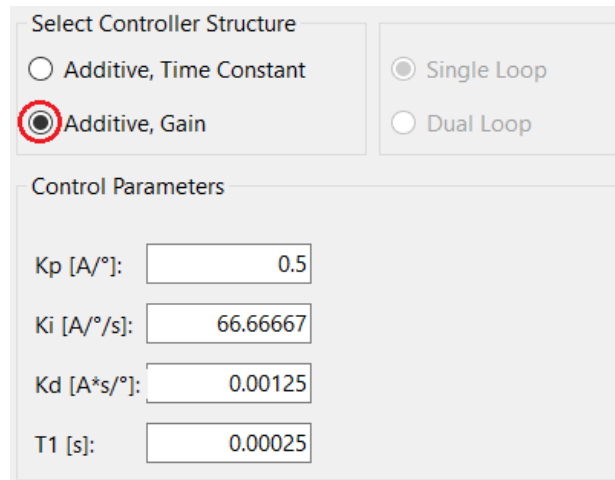
⁸ For other typologies see application note [9] for dual loop axes and [11] for gantry axes.

troller Parameter panel shows the parameters used to configure the *PIDT1* controller.

Two equivalent controller structures can be selected in the **Select Controller Structure** panel. This selection has no effect on the behavior of the controller as all structures are internally transformed to the **Additive, Gain** structure, but it allows to select the structure which is most suitable for the tuning.

- **Additive, Gain:**

Figure 55 shows the parameters used to configure the *PIDT1* controller if **Additive, Gain** is selected. With this structure, the parameters are identical with the controller registers in the Topology Tree. The controller output is the sum of the integral path weighted with **Ki**, the proportional path weighted with **Kp**, and differential path weighted with **Kd** and filtered with a low pass filter with time constant **T1** (see also Figure 56).



The screenshot shows the 'Select Controller Structure' panel with 'Additive, Gain' selected. The 'Control Parameters' section displays the following values:

Parameter	Value
Kp [A/°]:	0.5
Ki [A/°/s]:	66.66667
Kd [A*s/°]:	0.00125
T1 [s]:	0.00025

Figure 55: Controller Parameter with Additive, Gain selected.

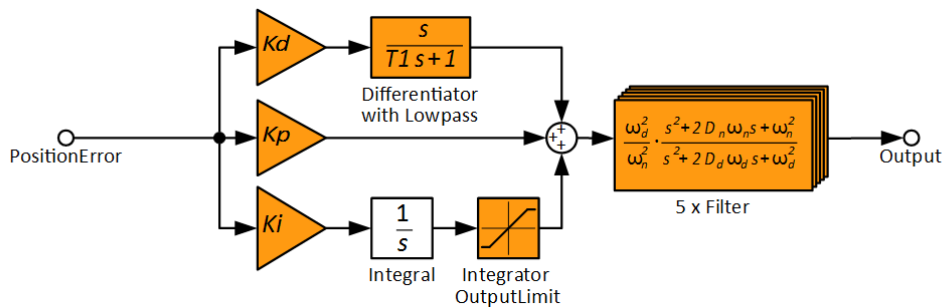


Figure 56: Additive, Gain controller - the parameterization of this structure corresponds to the controller registers in the Topology Tree.

- **Additive, Time Constant:**

Figure 57 shows the parameters used to configure the *PIDT1* controller if **Additive, Time Constant** is selected. In this case the integral and derivative terms are weighted by the time constants T_n and T_v and the position error is multiplied by K_r (see also Figure 58).

Select Controller Structure

☒ Additive, Time Constant
☐ Additive, Gain

☐ Single Loop
☐ Dual Loop

Control Parameters

Translated:

Kr [A/°]: 0.5
Kp [A/°]: 0.5

Tn [s]: 0.007499999937
Ki [A/°/s]: 66.66667

Tv [s]: 0.0025
Kd [A*s/°]: 0.00125

T1 [s]: 0.00025
T1 [s]: 0.00025

Figure 57: Controller parameters with Additive, Time Constant selected

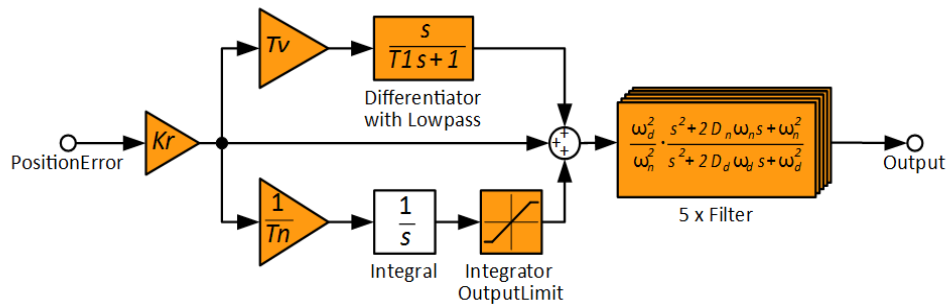


Figure 58: Additive, Time Constant structure - the parameters are directly transformed to the gain-structure as used in the Topology Tree.

The following table shows the equations of the two structures and how **Additive, Time Constant** is transformed to **Additive, Gain**:

Additive, Gain	Additive, Time Constant
$G_r(s) = Kd \frac{s}{T1s+1} + Kp + Ki \frac{1}{s}$	$G_r(s) = Kr \left(\frac{Tv s}{T1s+1} + 1 + \frac{1}{Tn s} \right)$
$Kp = Kr ; \quad Ki = \frac{Kr}{Tn} ; \quad Kd = Kr Tv ;$	

Note In most cases it is recommended and more intuitive to use **Additive, Time Constant** for the tuning of the position controller.

With the **Select Filter** panel up to 5 filters can be configured. Use the **Filter Parameters** panel to set the **Type**, frequency **f[Hz]** and damping **D[1]**. To disable the filter, the type is set to **Bypass**. If **Advanced** is checked, the frequency and damping of the numerator and denominator can be set independent (see also chapter 5.4.9).

The *Bode* plots show the following complex transfer functions:

$$G_r(f); \quad G_o(f) = G_r(f) H_c(f) \frac{x}{i}(f) F_0(f) F_1(f) \dots F_4(f); \quad G_c(f) = \frac{G_o(f)}{1 + G_o(f)}$$

with

- f : frequency
- $G_r(f)$: position controller transfer function
- $G_o(f)$: open loop transfer function
- $G_c(f)$: closed loop transfer function
- $H_c(f)$: closed loop current controller transfer function
- $x/i(f)$: measured frequency response current to position
- $F_o(f)F_1(f)...F_4(f)$: filter transfer function

The *Nyquist* plot shows the open loop transfer function $G_o(f)$ in the complex area.

The goal for the tuning of the position controller is to get a stable controller with the following characteristics (Figure 59 and Figure 60):

- a) wide bandwidth of $G_c(f)$
- b) high amplification in the pass-band of $G_o(f)$
- c) sufficient attenuation in the stop-band of $G_o(f)$ (which may require to place some filters).

with the following tuning criteria in **Bode** plot view:

- d) The phase-margin of $G_o(f)$ should be in range of 30° to 50° .
- e) The amplitude of the $G_c(f)$ should not exceed 2dB (robust) to 4 dB (advanced) for a stable controller behavior over the whole frequency range.

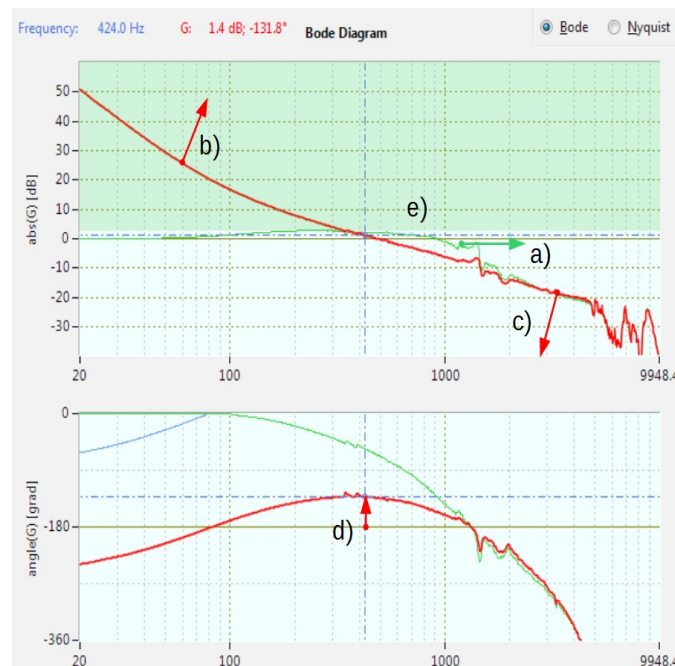


Figure 59: Tuning criteria in Bode view for the position controller.

Tuning criteria in **Nyquist** plot view:

- f) The displayed complex curve of $G_o(f)$ must be outside of the 1.3 (robust) to 1.6 (advanced) circle but as close to it as possible.
- g) Following the complex curve $G_o(f)$ from low frequency to high frequency, the critical point $[-1,$

0j] must be located in the region on the left of the curve.

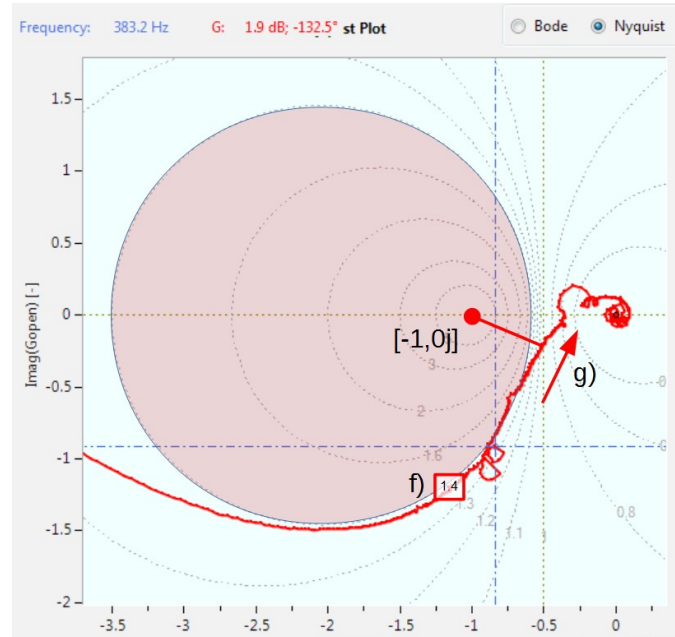


Figure 60: Tuning criteria in Nyquist view for the position controller.

5.4.7 Rule of Thumb for the Tuning of the Position Controllers

This section describes a rule of thumb for an initial setup of the position controller. See also the Estimate Controller section 5.4.8.

1. Set $K_r = 1$ and $T_n = 0$, activate the **Bode** plot view and **Additive, Time Constant** in the **Select Controller Structure** panel.
2. Choose an initial value for the gain crossover frequency f_{gc} ⁹. For a high performance axis with analog encoder, f_{gc} is typically in the range of 300Hz to 600Hz. For less stiff axes or encoders with low resolution, f_{gc} may be reduced to below 100Hz to meet the stability criteria. A good starting point is often 200Hz which can be adjusted later iterative.
3. Based on the selected f_{gc} , initial values for the parameters T_v and T_1 can be calculated:

$$T_v = \frac{1}{2f_{gc}} \quad T_1 = \frac{1}{20f_{gc}}$$

The angle plot should now show a sufficient phase margin⁷ at f_{gc} of more than 30° to 50°. If this is not the case, the gain crossover frequency f_{gc} may have to be reduced.

4. Measure the amplitude A_{gc} at frequency f_{gc} in dB in the bode plot with the crosshair. An initial value for the gain K_r can now be calculated based on the measurement:

$$K_r = 10^{-\frac{A_{gc}}{20}}$$

The magnitude of G_o should now cross the 0dB level at the frequency f_{gc} .

5. Check the stability of the controller in the Bode and the Nyquist plot based on the criteria listed

⁹ The gain crossover frequency f_{gc} is the frequency at which the gain of the tuned controller (open loop) crosses 0dB.

above. If the stability of the controller is insufficient reduce f_{gc} or try to apply filters to compensate unstable resonances.

- If the stability is sufficient, the integral time constant can be set by using the following equation as an initial value.

$$T_n = \frac{1.5}{f_{gc}}$$

- Use the criteria above to further optimize the position controller. Figure 61 shows a tuned controller and indicates qualitatively the effect of the parameter changes on $G_o(f)$ in the *Bode* and *Nyquists* plot.
- Press the **Commit** button to activate the parameter changes on the drive.

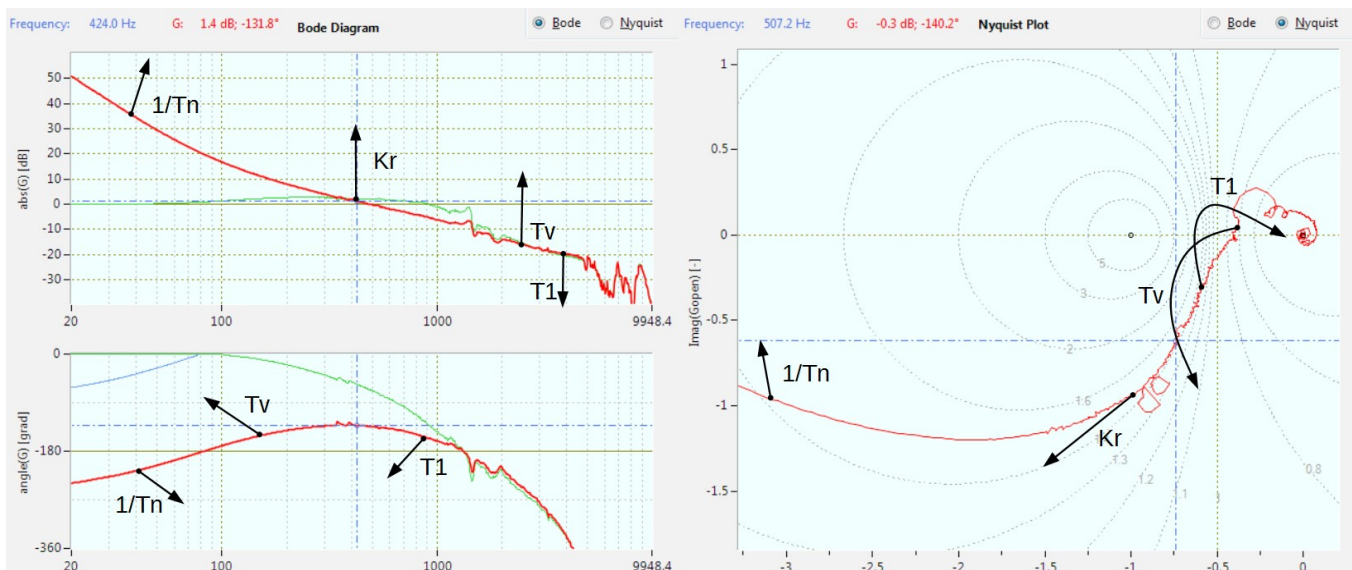


Figure 61: Tuned position controller with a gain crossover frequency f_{gc} of 420Hz and a phase margin of 48.2°. The black arrows indicate the effect of parameter changes in Bode and Nyquist plot (qualitatively).

5.4.8 'Estimate Controller'-Button (Position Controller)

The **Estimate Controller** button can be used to apply an initial tuning of the position controller. The applied tuning is based on the rule of thumb described in section 5.4.7. Follow the following steps to use the **Estimate Controller** button for an initial tuning:

- In the Bode View place the cross hair at the desired gain crossover frequency f_{gc} (see section 5.4.7. for more information about f_{gc}).
- Press the **Estimate Controller** button. With this, an initial controller setting is calculated.
- Use the criteria in section 5.4.6 to further optimize the position controller.
- Press the **Commit** button to activate the parameter changes on the drive.

Note

The **Estimate Controller** algorithm is applicable for common axes. Depending on the properties of the axis the quality of the resulting tuning can vary and in some cases the algorithm might not be applicable.

5.4.9 Filters

Filters can be used to enhance the properties of the controller loop for example by compensating mechanical resonances. Although filters can improve the controller loop, filters should be applied cautiously for the following reasons:

- Controller loops with filters are often more sensitive to changes to the axes.
- Filters increase the complexity of the controller-loop. Therefore a robust setting is more difficult to find.
- Consider that only resonances which violate the stability criterion (see section 5.4.6) need to be compensated.

The design of a filter should be done with the *Frequency Response* tool based on a *Frequency Response* measurement. Different filter types can be selected. But in most cases Notch2 is applied and will be described below.

Only for position measurement with poor resolution (e.g incremental encoders) Lowpass1 or Lowpass2 are used to smooth the signal. In this case the frequency fd is set 3 to 10 times the bandwidth of the controller and for Lowpass2 the damping Dd is set to 0.7. Consider that the application of lowpass filters will reduce the phase-margin and therefore affect the stability of the controller. It is recommended to verify the stability of the controller with the *Frequency Response* tool after a filter is modified.

Notch2

The following equation shows the transfer function of the Notch2 filter:

$$F_i(s) = \frac{\omega_d^2 s^2 + 2D_n \omega_n s + \omega_n^2}{\omega_n^2 s^2 + 2D_d \omega_d s + \omega_d^2}$$

with the following relation to the configuration parameters:

$$\begin{aligned} \omega_d &= 2\pi fd &= 2\pi \text{EdgeFrequencyDenominator} \\ D_d &= Dd &= \text{DampingDenominator} \\ \omega_n &= 2\pi fn &= 2\pi \text{EdgeFrequencyNumerator} \\ D_n &= Dn &= \text{DampingNumerator} \end{aligned}$$

The Notch2 filter can be considered as the product of a *Lowpass2* defined by fd and Dd and a *Highpass2* filter defined by fn and Dn (Figure 62). The *Highpass2* parameters fn and Dn define the location and the sharpness of the notch. The *Lowpass2* is used to compensate the strong amplification with higher frequencies caused by the *Highpass2*. As a simple approach, set the parameters as follows:

$$\begin{aligned} fn &= \text{Frequency to compensate} \\ Dn &= 0.01..0.05 \\ fd &= fn \\ Dd &= 10 Dn \end{aligned}$$

When **Advanced** mode is not checked in the **Filter Parameters** panel, fd and Dd are automatically calculated according to the equations above.

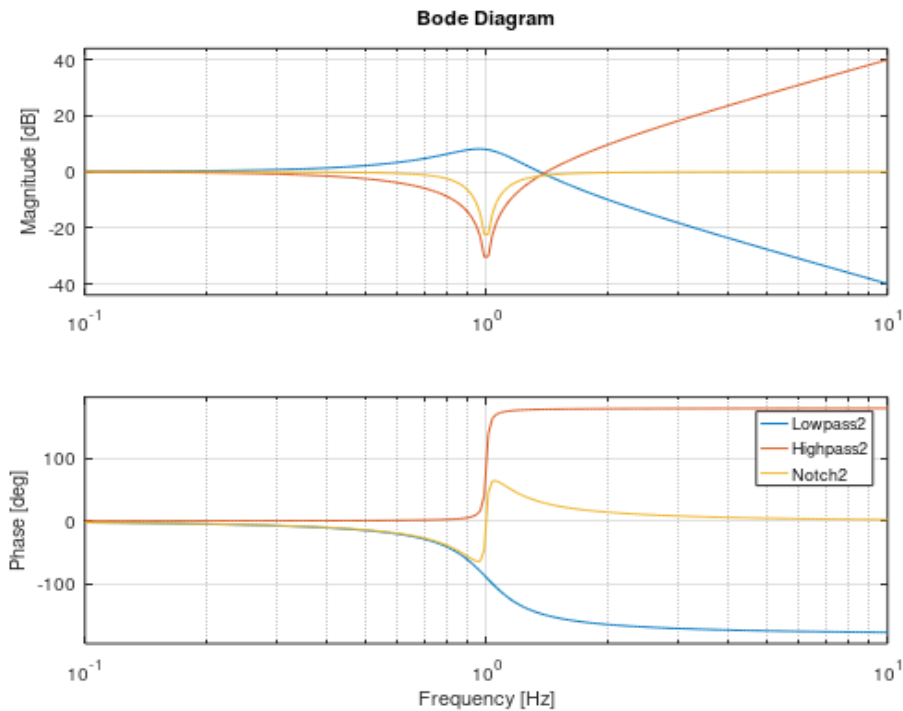


Figure 62: The Notch2 filter as product of a Lowpass2 with $f_d=1$, $D_d=0.2$ and a Highpass2 with $f_n=1$, $D_n=0.01$.

Consider the phase drop in the phase plot of Figure 62 below the notch frequency. This phase drop may reduce the phase margin and could reduce the stability of the controller. Therefore, the stability of the controller has to be checked after a filter is added or modified.

Figure 63 shows a resonance which causes an instability at 2000Hz. In the Nyquist plot, resonances appear as circles. Figure 64 shows the same transfer function now with a Notch2 filter applied at 2000Hz. The controller loop is now stable.

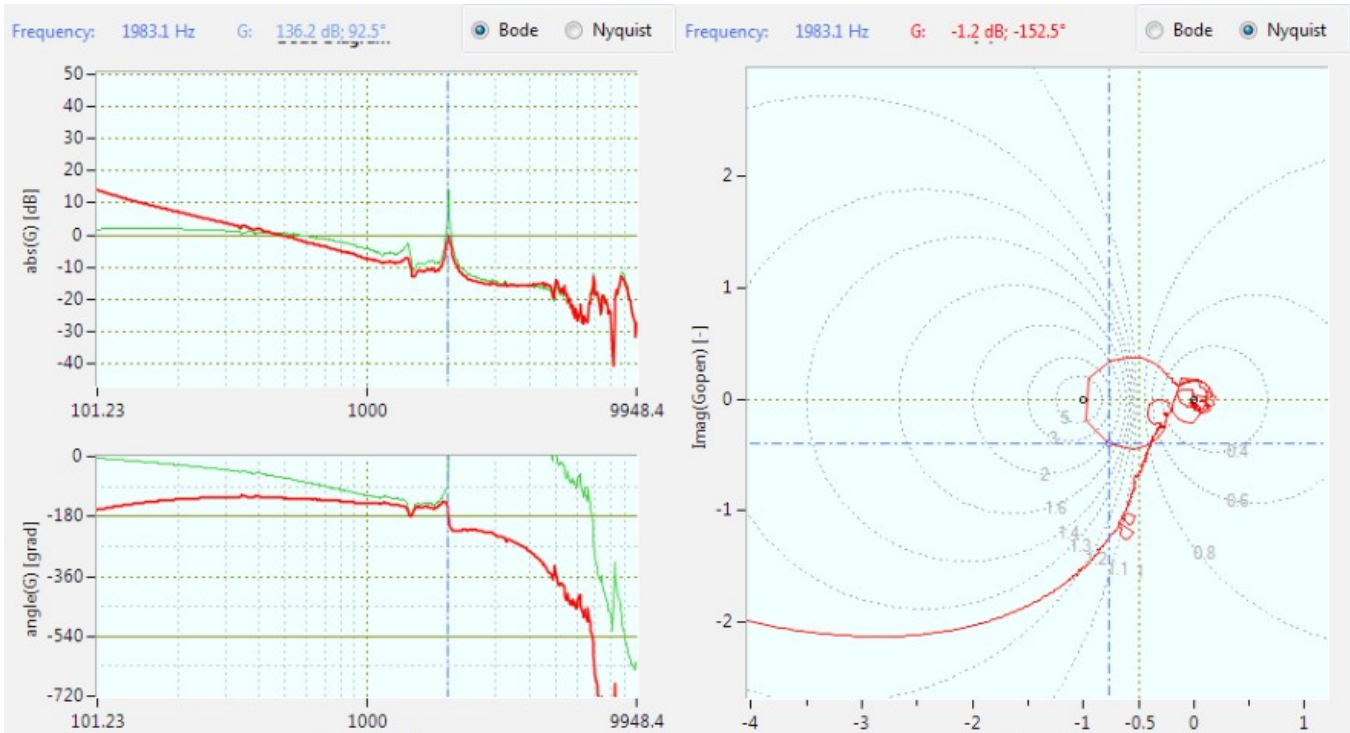


Figure 63: Resonance at 2000Hz which causes an instability of the controller-loop.

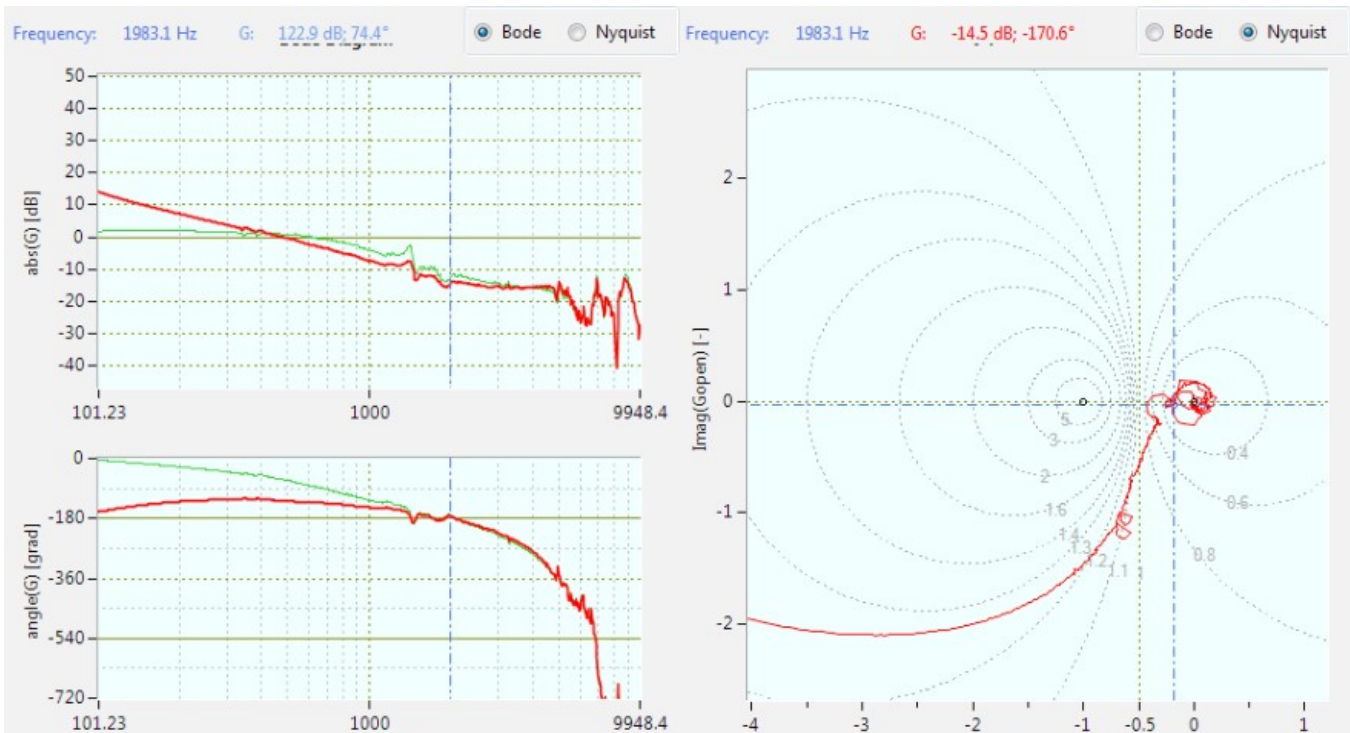


Figure 64: Notch2 filter with $f_n = f_d = 2000\text{Hz}$, $D_n = 0.01$ and $D_d = 0.1$ compensates the resonance.

5.4.10 Acceleration Feed Forward

A proper setting of the value for FeedForwardAcceleration is very important to achieve small position

errors during high dynamic movements. The value for FeedForwardAcceleration can also be determined based on the Frequency Response Measurement:

1. Open the measured data with the **Frequency Response Tuning** application. See section 4.2.2 on how to open the measurement and how to use the application.
2. Select **Position 0** in the **Select Controller** panel if single loop controller is used.
3. At lower frequencies the amplitude in the Bode view of *Gplant* should show a linear section with a drop of 40dB per frequency decade and a phase angle of -180 degree. Place the cursor at a frequency f within the linear section (see also Figure 65).
4. Press the **Estimate FFA** button to evaluate the gain for the acceleration feed forward.
5. Press the **Commit** button to activate the parameter changes on the drive.

The **Estimate FFA** algorithm uses the following equation for the calculation with A the magnitude of *Gplant* at the selected frequency f .

$$\text{FeedForwardAcceleration} = \frac{1}{10^{\frac{A}{20}} (2\pi f)^2}$$

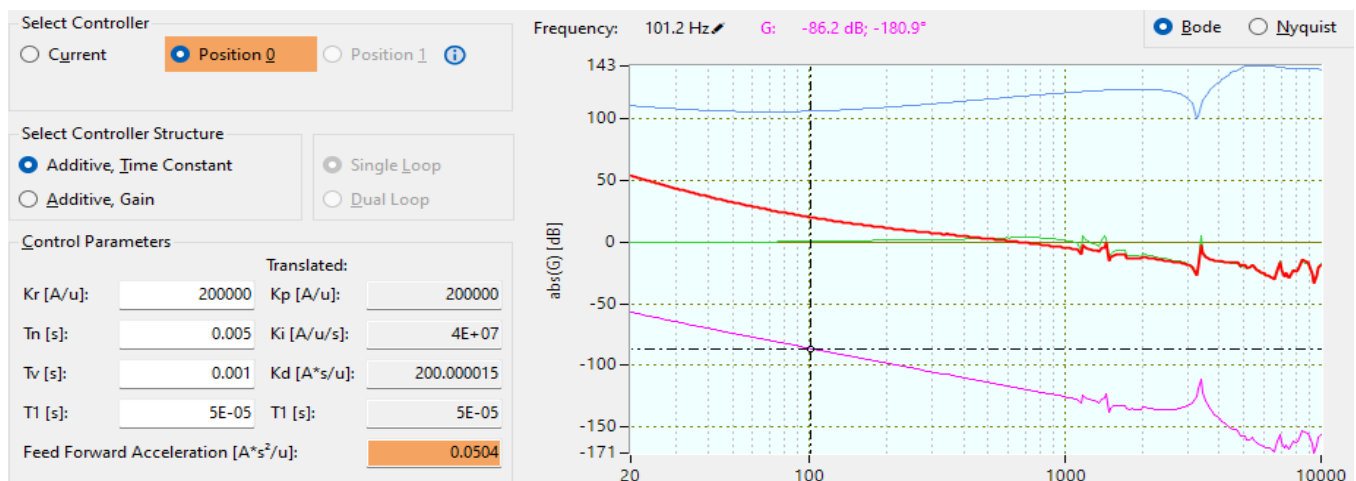


Figure 65: Calculation of the acceleration feed forward. With a measured amplitude of -65.3dB at 29.8Hz, the acceleration feed forward is set to 0.052.

5.5 Verification of the Setup

This section describes how to use the **Axis Module** and the **Scope** to verify the setup (see also section 3.3 and 4.1).

5.5.1 Motor Direction and Magnetic Pitch

To verify the motor direction and the magnetic pitch, the TestGenerator is used to apply a rotating current vector, which will move the axis in positive electrical direction. It is recommended to execute this check if:

- The Frequency Response measurement shows an unexpected characteristic.
- The axis goes in an axis error state immediately after the phasing.
- There is uncertainty if the magnetic pitch is configured correctly.

Preconditions:

- To rotate a current vector, the current controller needs to be configured properly as described in section 5.4.2.

The following steps are required to run the verification.

- Temporary increase `PositionController.Controllers[0].PositionErrorLimit` to a value bigger than one pole-pair pitch (e.g. to 10 times the magnetic pitch).
- Temporary set the gains `Kr`, `Ki`, `Kd` of the position controller to zero to disable the position controller.
- Prepare the scope by loading the *Scope Configuration* named **Direction Analysis**.
- Prepare the TestGenerator with the following values.
 - ♦ Frequency: This parameter defines how fast the axis will move during the test. For example, if the parameter is set to 0.5Hz, it will take 2s to move a distance of one pole pair pitch.
 - ♦ Amplitude: 0
 - ♦ Offset: 50% to 100% of the nominal current
 - ♦ RampTime: 2s
 - ♦ CommutationAngle: 0rad
- Enable the axis, start the scope and set `Axes[0].Commands.TestGenerator.Command` to `StartRotatingVectorConstantCurrent` and press Enter.

Warning As soon as `StartRotatingVectorConstantCurrent` is committed, the axis will start to move. The value of `TestGenerator.Frequency` has to be set slow enough to stop the axis in time (see next item).

- If the setup is correct, the axis will now move with the defined frequency. To stop the axis, set `TestGenerator.Command` to `Stop` and press Enter. Stop the scope and disable the axis.
- To verify the motor direction, check if the signals of `Commutation.Angle` and `PositionController.Encoders[0].Position` have the same direction (see also Figure 66). If the direction is opposite, switch the state of the parameter `Motor.InvertDirection`.
- To verify the magnetic pitch, check if one electrical turn ($2 \cdot \pi$) on the `Commutation.Angle` corresponds to the magnetic pitch measured on `PositionController.Encoders[0].Position` (see Figure 66).
- Do not forget to reset `PositionController.Controllers[0].PositionErrorLimit` and the position gains `Kr`,

Ki and Kd.

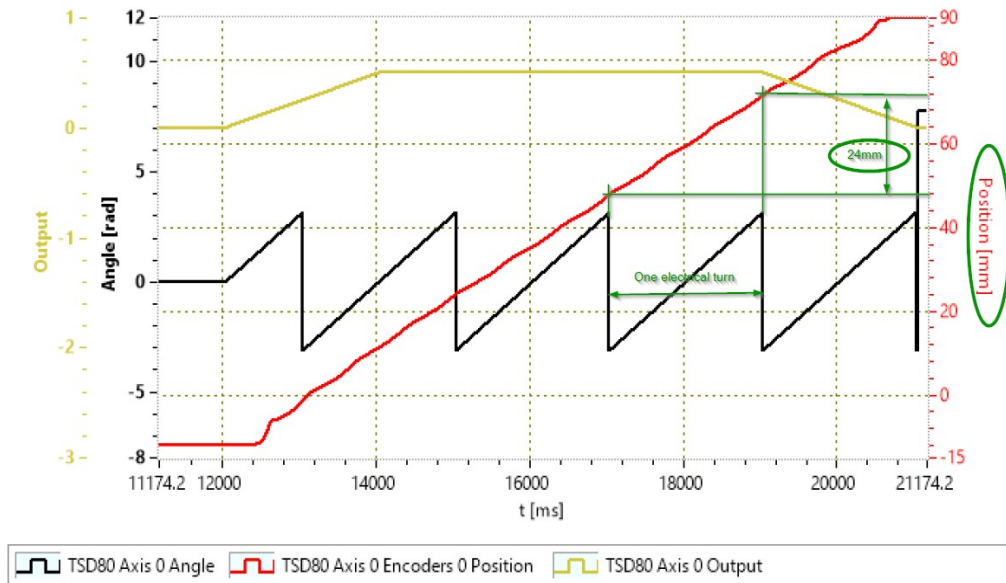


Figure 66: Verification of motor direction and magnetic pitch: The motor direction is correct as the signal of Angle and Position show the same direction. The magnetic pitch is 24 mm according to the measurement.

5.5.2 Commutation

To verify the commutation of the axis, the axis should be enabled at different positions and it should be verified if a move can be executed:

1. Check if Parameters.Commutation.PhasingMethod is set as desired and if Parameters.Commutation.EnablingMethod is *not* set to Automatic!
2. Prepare the scope by loading the *Scope Configuration* named **Phasing Analysis**.
3. Start the scope and enable the axis.
4. Check the scope if the characteristic of the signals is as expected.
 - ◆ If Commutation.Commutation.PhasingMethod is RotorAlignment:
 - Check if the current ramp of ActualCurrentD is applied as expected.
 - Check if the position signal shows an alignment move.
 - Axes with low friction (e.g. with air-bearings) could show a slow oscillation during the phasing which could lead to an invalid commutation. In this case the PhasingMethod should be changed to AngleSearch2 and the commutation parameters should be adjusted accordingly (see section 5.2.7).
 - If Commutation.PhasingMethod is AngleSearch or AngleSearch2:
 - Check if the current ramp of ActualCurrentD is applied as expected.
 - Check if the position signal shows only a minor motion.
 - Check if the position shows no instability and the position error is zero at the end of the phasing. If the error is not zero, try to reduce Parameters.Commutation.RampRiseTime and verify the stability of the position controller. Consider switching the PhasingMethod back to RotorAlignment to verify the position controller with a new *Frequency Response* measurement.
 - ◆ No loud noises should be heard during phasing.

5. Check if the axis is stable by manually applying torque to the axis if possible.
6. Reduce the **Acc.** and **Vel.** values in the **Axis Module**, start the scope and execute a move of at least one electrical turn.
7. Repeat this sequence at different positions.
8. If the phasing works as expected, `Parameters.Commutation.EnablingMethod` can be set to Automatic.

Note If the axis goes into error state immediately after the phasing sequence or at the start of a move, the parameter `MotorInvertDirection` might be incorrect. Also verify `Parameters.Motor.PolePairs` and `Parameters.Motor.EncoderCountsPerMotorRevolution` (see also section 5.5.1).

5.5.3 Position Controller

If the commutation works as expected, the controller setup can be verified in time domain. In the following, two different approaches are described with which a setpoint can be generated:

Test Generator

If registers in node `Commands.TestGenerator` are configured accordingly, a square wave can be added to the position setpoint. By measuring the position or the position-error the step answer can be analyzed. The step size (amplitude) of the test signal has to be small enough, to not exceed the output limits of current and voltage. Therefore step answer is especially useful to analyze the small signal behavior and the stability of the controller.

1. Prepare the scope by loading the *Scope Configuration* named **Position Controller Disturbance Tuning**.
2. Increase the value of `Parameters.PositionController.Controllers[].PositionErrorLimit` temporarily to not cause an error during the test.
3. Start the scope.
4. Enable the axis for example by using the **Axis Module**.
5. Prepare the test generator: Frequency: 5Hz, Offset: 0, RampTime: 0. Amplitude: The amplitude should be set to a small value to not exceed the limits of current or voltage.
6. Start the test signal by setting `Commands.TestGenerator` to `StartPositionSquare`.

Axis Module

With the axis module, a move can be executed which is generated by the internal path planner with restriction of velocity, acceleration and jerk. Therefore this test is useful to analyze the process related stability and behavior of the axis and the settings of the feed forward.

1. Prepare the scope by loading the *Scope Configuration* named **Position Controller Follower Tuning**.
2. Prepare the **Axis Module**: Setup a reasonable move between A and B. Activate the **Loop After** check-box and set a wait time of about 500ms. Reduce the dynamic by adjusting the **Acc.** and **Vel.** values in the **Axis Module**.
3. Start the scope and execute moves by pressing **Move AB**. Check the transient response of the signals. Slowly increase the dynamic settings.

If the position error shows oscillations of a certain frequency, check the Bode plot for a reason of the

oscillation (see also Figure 67 and Figure 68). Try to adjust the parameters to increase the damping (increase phase margin) or reduce the amplitude. Consider applying filters to damp the oscillation.

If the position error increases when a certain velocity is reached, the limit for the voltage output might have been reached (see also OutputLimit of the current controller in 5.2.8).

If the position error increases when a certain acceleration is reached, the limit for the current output might have been reached (see also OutputLimit of the position controller in 5.2.4).

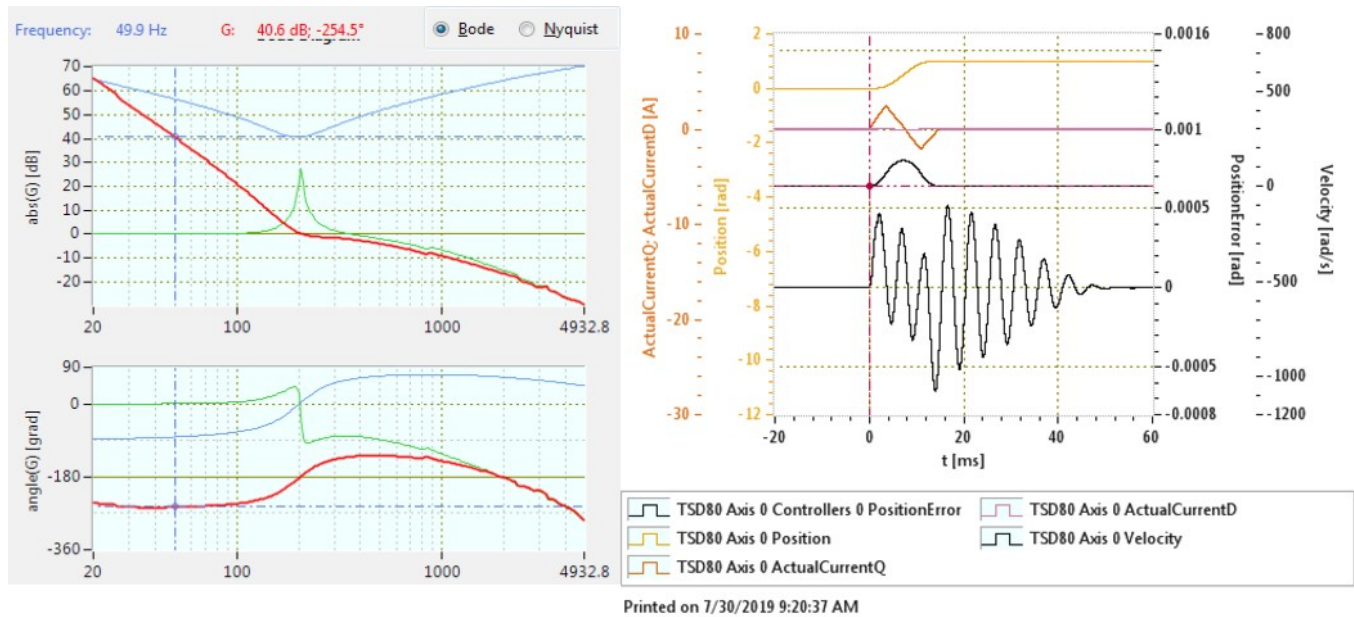


Figure 67: The Bode plot shows an insufficient phase margin at about 200Hz which is caused by a too aggressive setup of the integral. In time domain this causes oscillations of the position at 200Hz.

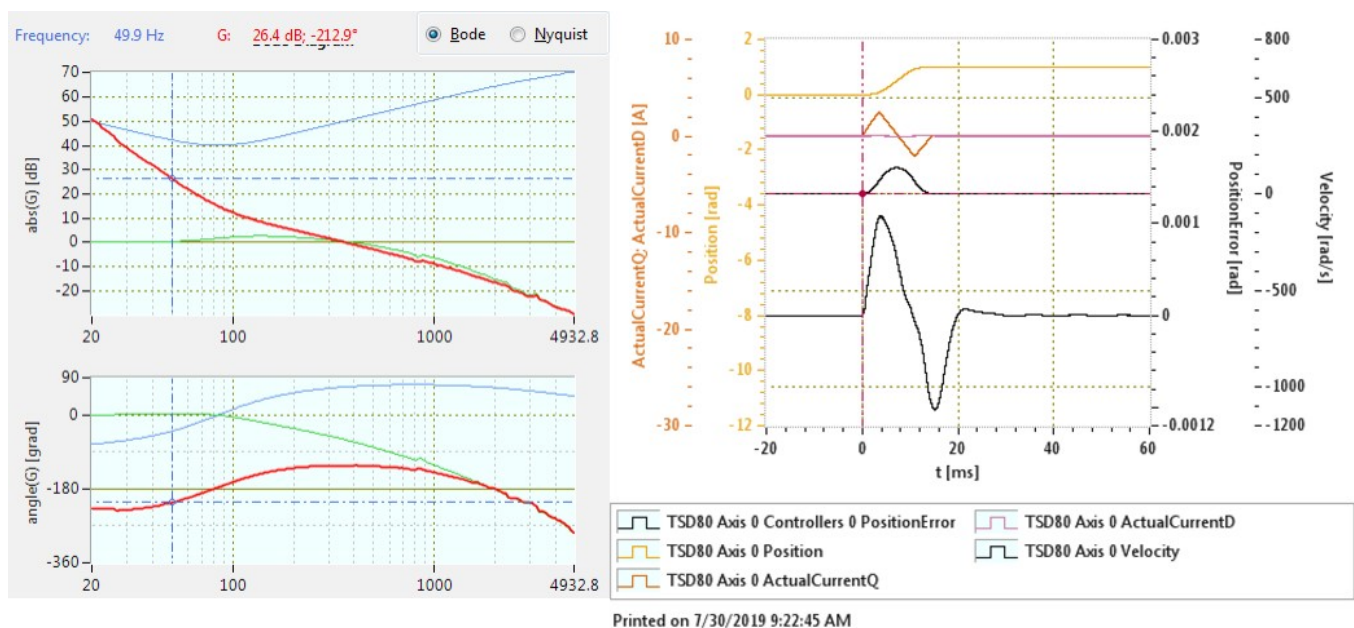


Figure 68: The bode plot shows a sufficient phase margin with sufficient damping. This results in a nearly aperiodic transient response in time domain.

5.5.4 Acceleration and Velocity Feed forward.

The following setup is used to verify the acceleration feed forward (see also 5.4.10):

1. Prepare the scope by loading the *Scope Configuration* named **Feed Forward Analysis**.
2. Prepare the **Axis Module**: Setup a reasonable move between A and B. Activate the **Loop After** check-box and set a wait time of about 500ms.
3. Start the scope and execute moves by pressing **Move AB**.

If the acceleration feed forward is correctly configured, the FeedForwardCurrent should match the acceleration related part of ActualCurrentQ and the position error becomes much smaller compared to without acceleration feed forward. In some cases also velocity feed forward has to be added to get a good match.

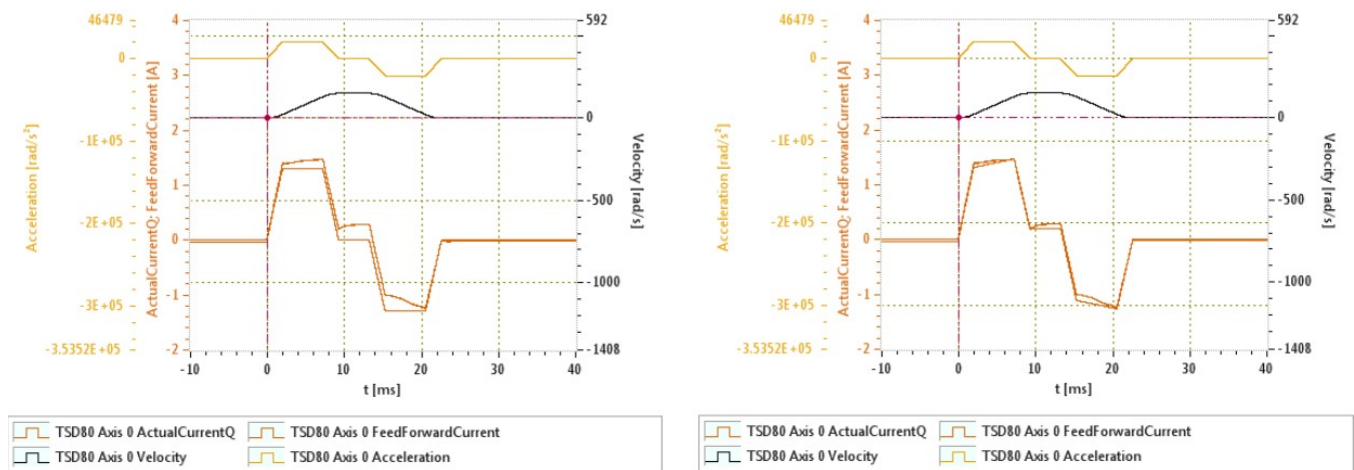


Figure 69: Feed forward signal with only acceleration feed forward (left) and with additional velocity feed forward (right).

6 Advanced Topics

This chapter handles topics which are not part of the basic setup of an axis described in chapter 5.

6.1 Commutation with Absolute Encoders

For the commutation of a synchronous AC motor the angle between the magnetic field of the rotor and the stator windings needs to be known. With relative encoders, this angle has to be determined at least after each restart of the drive which is called *Phasing* (see also section 5.2.7). With absolute encoders, the commutation angle does not change relative to the encoder position and therefore the phasing has to be executed only once. The resulting commutation angle can then be stored and reused after each restart. The following absolute encoder protocols are supported:

- Endat
- BissB, BissC
- Tamagawa
- Nikon

This section describes how to setup the commutation with absolute encoders and how the commutation angle can be saved to the drive or to the encoder.

Note Consider that mechanical adjustments of the encoder or the motor or replacement of the encoder, motor or the drive affect the commutation angle and may require an execution of a phasing-sequence to restore the commutation angle.

6.1.1 EnablingMethod

The parameter `Commutation.EnablingMethod` allows to define how the commutation angle is derived from the absolute encoder. The following options are available:

- ♦ `AbsoluteEncoder`: In this case the commutation angle is directly derived from the absolute position of the encoder. This case requires a preset of the encoder by the manufacturer of the motor or by the user (see section 6.1.2).
- ♦ `AbsoluteEncoderOffsetEncoder`: In this case the commutation offset is loaded from the encoder memory. This requires a successful initial phasing of the axis followed by storing the offset to the encoder (see section 6.1.3).
- ♦ `AbsoluteEncoderOffsetDrive`: In this case the commutation offset is loaded from the drive memory. This requires a successful initial phasing of the axis followed by storing the configuration persistent on the drive (see section 6.1.4).

6.1.2 Set Absolute Encoder Zero equal to Commutation Zero

The following steps are required to set the encoder zero equal to the commutation zero:

1. The commutation sequence has to be set to RotorAlignment in `Axes[].Parameters.Commutation` (see also section 5.2.7).
2. Execute the phasing-sequence by executing the command `Axes[].Commands.Commutation.Command = StartPhasingAndZeroEncoder`.
3. Check if `Axes[].Signals.Commutation.State` is Valid. If State is *not* Valid, check the configuration, especially the commutation settings, and restart the phasing-sequence.
4. To activate commutation based on the absolute encoder set `Axes[].Parameters.Commutation.EnablingMethod` to `AbsoluteEncoder`.

6.1.3 Save Commutation Offset to the Encoder

The following steps are required to save the commutation offset to the encoder:

1. The commutation sequence has to be configured correctly in `Axes[].Parameters.Commutation` (see also section 5.2.7).
2. Execute the phasing-sequence by executing the command `Axes[].Commands.Commutation.Command = StartPhasingAndSaveEncoder`.
3. Check if `Axes[].Signals.Commutation.State` is Valid. If State is *not* Valid, check the configuration, especially the commutation settings, and restart the phasing-sequence.
4. To use the commutation offset from the encoder set `Axes[].Parameters.Commutation.EnablingMethod` to `AbsoluteEncoderOffsetEncoder`.

6.1.4 Save Commutation Offset to the Drive

The following steps are required to save the commutation offset to the drive:

1. The commutation sequence has to be configured correctly in `Axes[].Parameters.Commutation` (see also section 5.2.7).
2. Execute the phasing-sequence either
 - ♦ by enabling the axis with the axis module (see 4.1) or
 - ♦ by executing the command `Axes[].Commands.Commutation.Command = StartPhasing`.
3. Check if `Axes[].Signals.Commutation.State` is Valid. If *not* Valid, check the commutation settings and restart the phasing-sequence.
4. To use the commutation offset from the drive set register `Axes[].Parameters.Commutation.EnablingMethod` to `AbsoluteEncoderOffsetDrive`.
5. If `Axes[].Signals.Commutation.State` is Valid save the configuration persistent on the drive (see section 3.5.1) to store the commutation offset.

6.1.5 How the Commutation Offset is Derived

During startup of an encoder, the axis checks if the parameter `axes[].Parameters.Commutation.EnablingMethod` requires taking the commutation offset from any absolute encoder:

- `AbsoluteEncoderOffsetDrive`: try getting the offset from the persisted drive data.
- `AbsoluteEncoderOffsetEncoder`: try getting the offset from the encoder name plate.

If this is successful, the commutation state will be Valid and enabling will be performed without phasing.

If this fails, the axis will issue the error NoDigitalEncoderPersistency, which can be acknowledged. The next enabling will then depend on the parameter `axes[].Parameters.Commutation.PhasingMethod`. If this is NoPhasing, enabling will throw an error. Otherwise, the specified phasing method is executed and enabling is possible.

With this concept, the user can specify, how the drive should behave in case of a missing absolute commutation information.

6.1.6 Signals and States

Beside the parameters used to configure phasing different commands and signals are provided to analyze and debug the phasing sequence.

`Axes[].Commands.Commutation.Command`: This register can be used to control the phasing state and to save the commutation angle to the encoder (if supported by the encoder¹⁰).

- Invalidate: Sets the commutation state to Invalid which forces a phasing sequence (if specified) when the axis is enabled next time.
- StartPhasing: Starts a phasing sequence as specified by the commutation parameters. StartPhasing can be executed whether the axis is enabled or disabled. After a successful phasing, the commutation offset is written to the register `Axes[].Signals.Commutation.OffsetAngle`, the commutation state is set Valid and the axis state will return to the state before the phasing.
- StartPhasingAndSaveEncoder: Same as StartPhasing but additionally the commutation offset is saved persistent to the encoder. If `EnablingMethod` is set to `AbsoluteEncoderOffsetEncoder` this offset will be used for the commutation after a power cycle of the drive.
- StartPhasingAndZeroEncoder: Same as StartPhasing but additionally the zero of the absolute encoder is set equal to the zero of the commutation angle. This will cause a shift of the position. In case the zero of the encoder is equal to the zero of the commutation angle it is possible to set the parameter `Commutation.EnablingMethod` to `AbsolutEncoder`.
- InvalidateEncoder: This command marks the commutation offset saved on the encoder as invalid. After a power cycle of the drive, the commutation state will be Invalid.

`Axes[].Signals.Commutation.State`: This register displays the current phasing state.

- Disabled: The motor type and angle search method chosen do not require angle search (e.g. DC-motor).
- Invalid: A valid commutation angle is not yet set. This will force a phasing when the axis is enabled next time.
- Phasing: The phasing sequence is ongoing.
- Hold: The phasing is frozen to allow saving of the commutation offset to the encoder.
- Valid: Any phasing was successful or the commutation offset was successfully derived from an absolute encoder depending on the `EnablingMethod` parameter. In both cases the commutation offset is written to the register `Axes[].Signals.Commutation.OffsetAngle`.

`Axes[].Signals.Commutation.Angle`: The commutation angle describes the magnetic field of the rotor relative to the stator-coils (Figure 70). The commutation angle runs from $-\pi p$ to πp , where p is the number

¹⁰ Endat; BissB; BissC; Tamagawa; Nikon

of pole pairs.

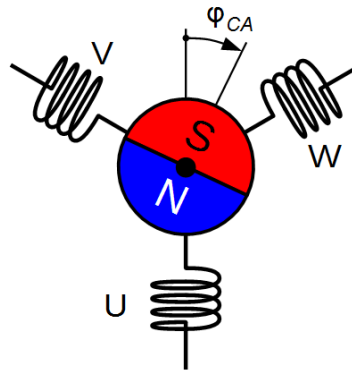


Figure 70: Commutation angle φ_{CA}
for a rotor with one pole pair.

`Axes[.Signals.Commutation.OffsetAngle`: The commutation offset is the angle between the encoder position and the zero of the commutation angle. This is the value evaluated during the phasing sequence.

7 Flow Charts

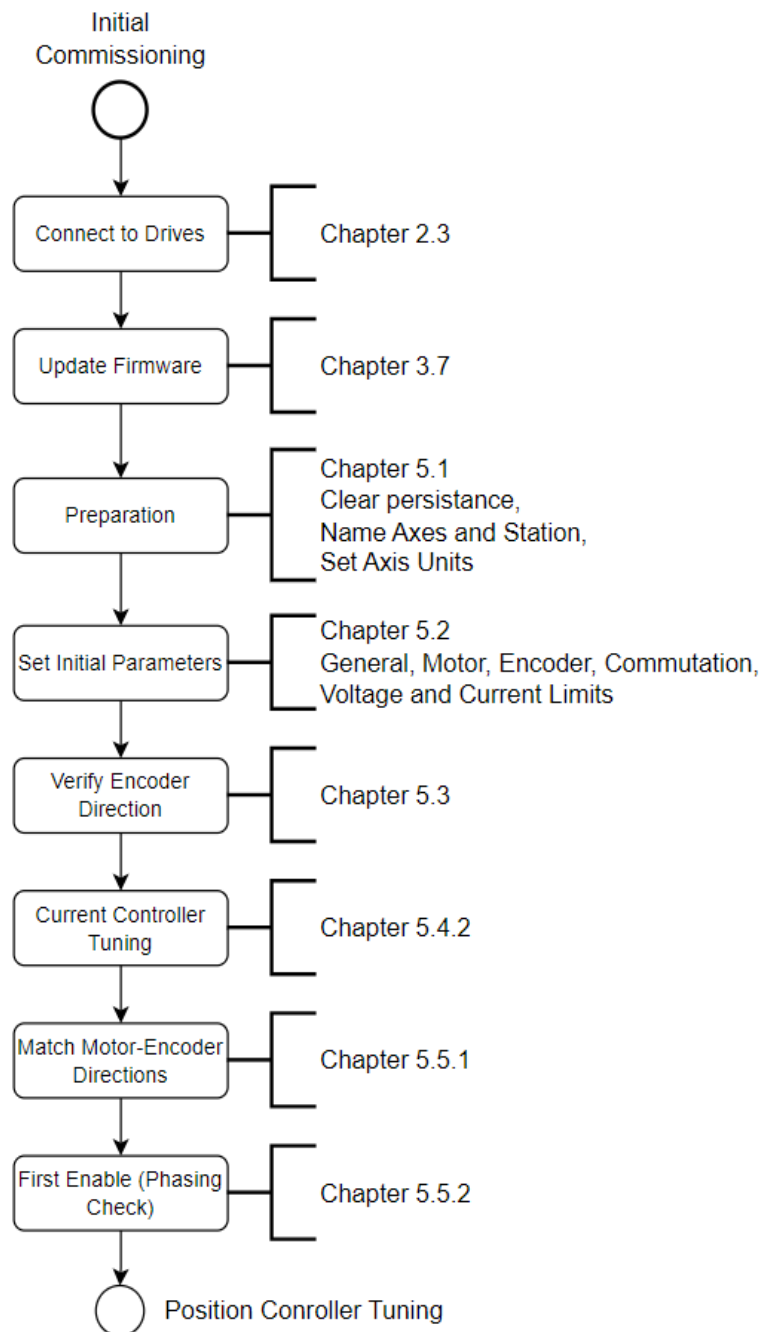


Figure 71: Initial Setup Guide Workflow

Position Controller Tuning

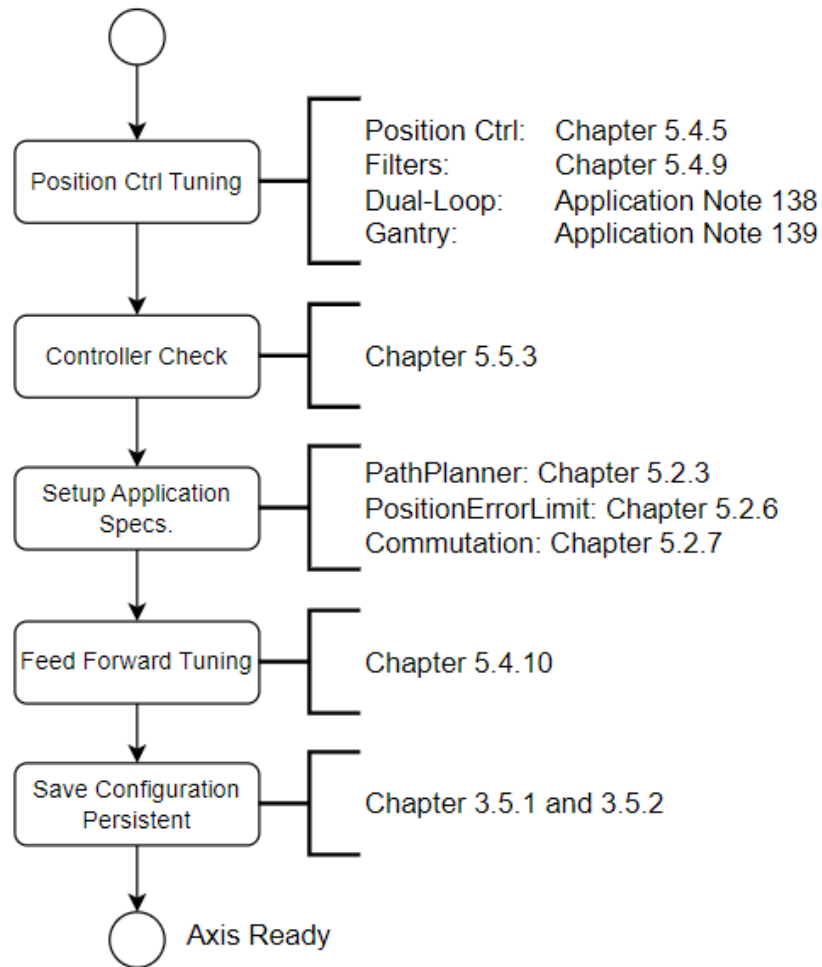


Figure 72: Position Controller Tuning Workflow

References

- [1] "Data Export and Import in TAM System Explorer Application Note", AN132_TamSystemExplorerDataExchange_EP006.pdf, Triamec Motion AG, 2022
- [2] "Triamec TwinCat Quick Startup Guide", SWTC_TwinCAT-UserGuide_EP024.pdf, Triamec Motion AG, 2016
- [3] "Triamec TwinCat Ethercat Quick Startup Guide", SWTC_TwinCAT-UserGuideEcat_EP005.pdf, Triamec Motion AG, 2019
- [4] "Ethernet Interface", AN123_Ethernet_EP004.pdf, Triamec Motion AG, 2022
- [5] "TAM API Developer Manual", SWNET_TamApiDeveloperManual_EP033.pdf, Triamec Motion AG, 2019
- [6] "Tama Real-Time Drive Programming", SWTAMA_UserGuide_EP001.pdf, Triamec Motion AG, 2024
- [7] "Encoder configuration for the TSD drive series", AN107_Encoder_EP017.pdf, Triamec Motion AG, 2022
- [8] "Sensorless Motor Commissioning User Guide", AN133_SensorlessCommissioning_UserGuide_EP005.pdf, Triamec Motion AG, 2022
- [9] "Dual-Loop Controller with Screw Drive", AN138_DualLoopControllerWithScrewDriveAxis_EP006.pdf, Triamec Motion AG, 2022
- [10] "Homing Procedures and Setup", AN141_HomingProceduresAndSetup_EP002.pdf, Triamec Motion AG, 2022
- [11] "MIMO Gantry Commissioning", AN139_MIMOGantryCommissioning_UserGuide_EP004.pdf, Triamec Motion AG, 2022
- [12] "FieldBus", AN155_FieldBus_EP001.pdf, Triamec Motion AG, 2025

Revision History

Version	Date	Editor	Comment
014	2022-01-14	chm	TAM System Explorer now has PCI access excluded in the default configuration
015	2022-04-04	sm	Update references, insert Flow Charts, minor fixes.
016	2022-05-13	sm	Insert the usage of scope templates, describe soft limit registers, minor updates
017	2022-08-17	dg	New function of Commutation.StartTime explained and new CurrentController.Field-Weakening parameters documented.
018	2022-08-24	chm	New chapter 3.5.4: Working Offline, update flow charts, remove 7.2
019	2022-12-12	chm	New chapter 3.9: Setup Protection
020	2023-02-20	dg	Review rb processed.
021	2023-05-25	chm	Review USB access in chapter 3.1. Document log directory in chapter 3.4 and F6 key in chapter 3.10. Document homing controls in chapter 4.1.
022	2023-07-03	dg	New second order filter for current controller documented.
023	2023-08-09	chm	Nest chapters in 4.2: Frequency Response Module
	2023-11-20	ab	Review of whole document, some minor corrections, added Tsp700 parameters
024	2023-11-27	ns, dg	Replace Bode with Frequency Response where applicable
025	2024-02-02	ab	Corrections in 5.2.1 and 'below' in 3.5.2, correct some bad Figure links, moved 5.5.4 to correct position
026	2024-03-19 2024-04-02	ab ns	Added BissC Added TAM Snapshot
027	2024-07-03	fm dg	Modification of chapter 2.3. New FW registers documented
028	2024-09-05	dg dg	Description for Estimate Controller and Estimate FFA added. Chapter Tama Program revised and linked to the Tama User Guide
029	2025-01-13	fm	Removed chapter 2.3, added reference to field-bus doc AN155
030	2025-04-25	yz	Added section for AngleSearch2

Copyright © 2025
Triamec Motion AG
All rights reserved.

Triamec Motion AG
Lindenstrasse 16
6340 Baar / Switzerland

Phone +41 41 747 4040
Email info@triamec.com
Web www.triamec.com

Disclaimer

This document is delivered subject to the following conditions and restrictions:

- This document contains proprietary information belonging to Triamec Motion AG. Such information is supplied solely for the purpose of assisting users of Triamec products.
- The text and graphics included in this manual are for the purpose of illustration and reference only. The specifications on which they are based are subject to change without notice.
- Information in this document is subject to change without notice.