XPS-D

Universal High-Performance Motion Controller/Driver

ESP COMPATIBLE

Newport®

Configuration Manual

Table of Contents

Universal High-Performance Motion Controller/Driver XPS-D Controller

1.0 Introduction

1.1 Scope of the Manual

The XPS is an extremely high-performance, easy to use, integrated motion controller/driver offering high-speed communication through 10/100/1000 Base-T Ethernet, outstanding trajectory accuracy, and powerful programming functionality. It combines user-friendly web interfaces with advanced trajectory and synchronization features to precisely control from the most basic to the most complex motion sequences. Multiple digital and analog I/O's, triggers and supplemental encoder inputs provide users with additional data acquisition, synchronization and control features that can improve the most demanding motion applications.

To maximize the value of the XPS Controller/Driver system, it is important that users become thoroughly familiar with available documentation.

The present **XPS-D Configuration Manual** describes how to configure the controller to drive Newport or non-Newport motorized stages or custom axes. Moreover, it provides all necessary information to properly configure and get the benefit of all the controller functions explained in Features Manual. It applies to several controller versions of the XPS-D family. Therefore, some details of the screenshots presented in this manual may slightly differ from reality (background picture or footer product name for instance).

1.2 Prerequisite

It is mandatory that both **XPS-D Start-Up Manual** and **User Interface Manual** be thoroughly read and understood before trying to configure the controller.

Particularly, appropriate driver cards must be installed, all stages must be connected and an Ethernet connection must be established between the computer and the controller, either directly or through a network.

Unless only Newport standard ESP compatible stages are to be used, it is also highly recommended to have read **XPS-D Features Manual** to understand the role of each parameter.

1.3 Special case of HXP-ELEC-D controller

When preconfigured to drive a hexapod, the XPS-D controller (then referenced HXP-ELEC-D) only offers two axes that can be configured for single axis stages. All the configuration options presented further are applicable to these two axes except those specific to XYZ group. The configuration parameters of the 6 axes dedicated to the hexapod are fully set in factory and should not be modified by the user otherwise the hexapod may not work properly. To prevent undesired modifications, the standard XPS-D website configuration pages are not accessible. Hence, the last two axes must also be set in factory or with the instructions given by Newport Support service. However, all the configuration files remain editable and changes should only be applied by experimented users.

1.4 Configuration Overview

There are three possibilities to configure the controller: Default configuration, Quick configuration and Manual configuration.

1.4.1 Default Configuration

Default configuration is the simplest method to configure the controller, but has some limitations:

- Default configuration works only with Newport ESP compatible positioners.
- Default configuration configures all detected positioners as single axis groups. However, single axis groups provide limited functionality (no synchronized motion, no trajectories, no XY or XYZ compensation). To take full benefit of the capabilities of the XPS controller, a manual configuration is needed.

1.4.2 Quick Configuration

Quick configuration is required for:

- Non ESP compatible Newport stages

1.4.3 Manual Configuration

Manual configuration is required for:

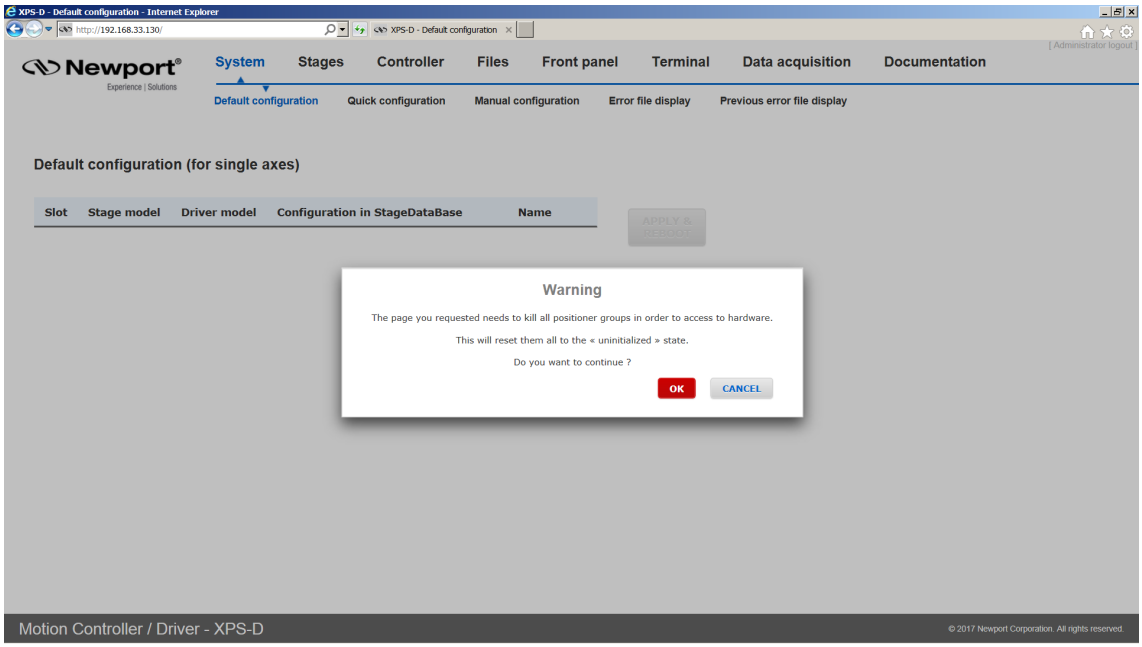
- Non-Newport stages or older version of Newport stages.
- Some vacuum compatible stages (no ESP chip).
- Stages with adjustable home position (-1, 0, +1), if the home position is changed from the standard position 0 to -1 or +1. The positions +1 and -1 require different settings in the stage database, as the home switch position is not recognized by the ESP chip.

2.0 Default Configuration

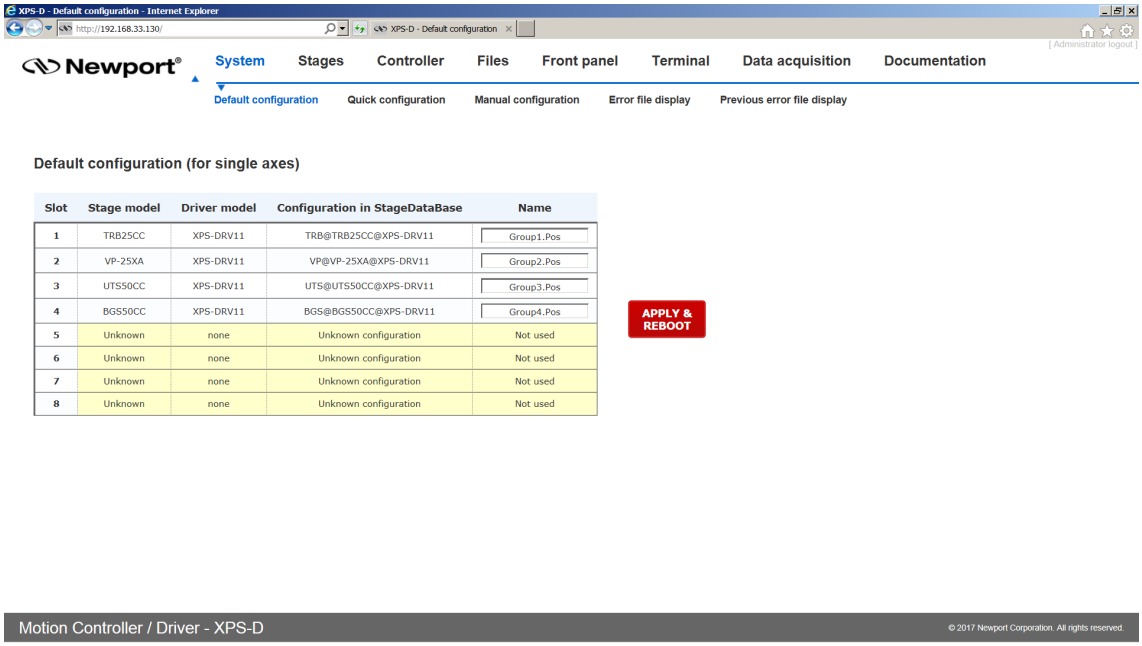
The Default configuration takes the stage configuration from the StageDataBase file.

2.1 Procedure

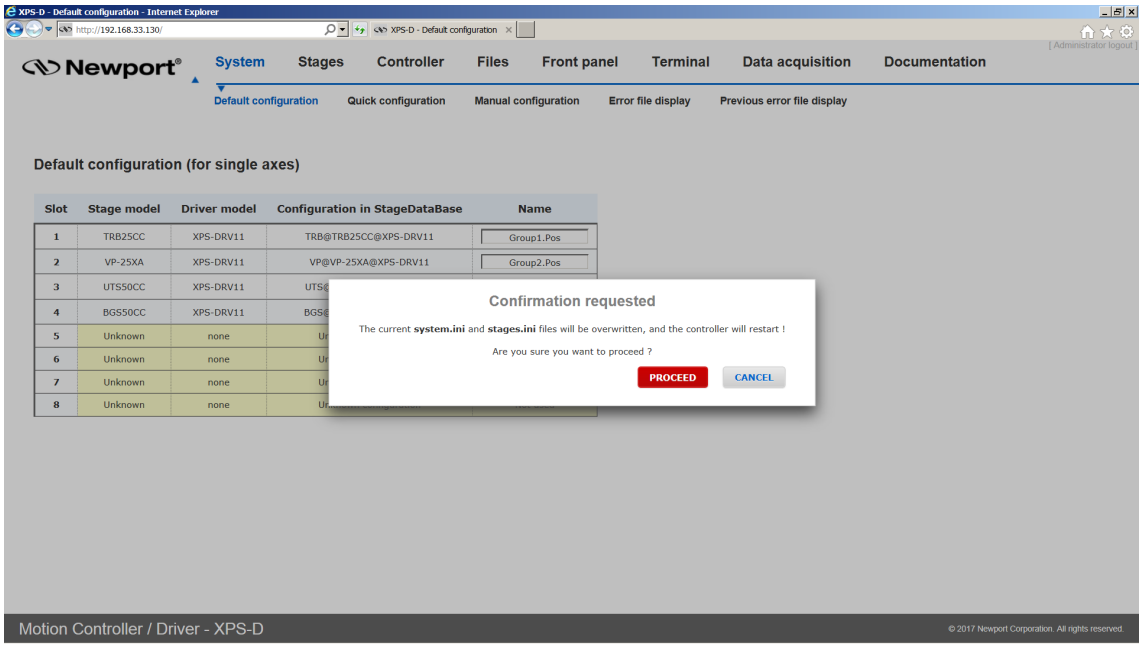
- When logged in as Administrator, select **SYSTEM**, then **"Default configuration"**. The following screen appears:



- If you want to continue, click the "OK" button and the following page appears:



- Check, if all connected stages are recognized by the system. Change the Group name type in the desired name under the Name column.
- If all connected stages are recognized, click "APPLY & REBOOT" and the following page appears:



- To configure the XPS click “PROCEED.” The controller reboots and the Login screen appears (this may take up to 1 min).

NOTE

“APPLY & BOOT” deletes your current system.ini configuration file. For troubleshooting a system, make sure to backup the original system.ini file for recovery.

- When the controller has finished booting login, select **FRONT PANEL**, and then select **“Move”**.

Your system is now ready to use. For more advanced functions, please read the rest of this manual.

NOTE

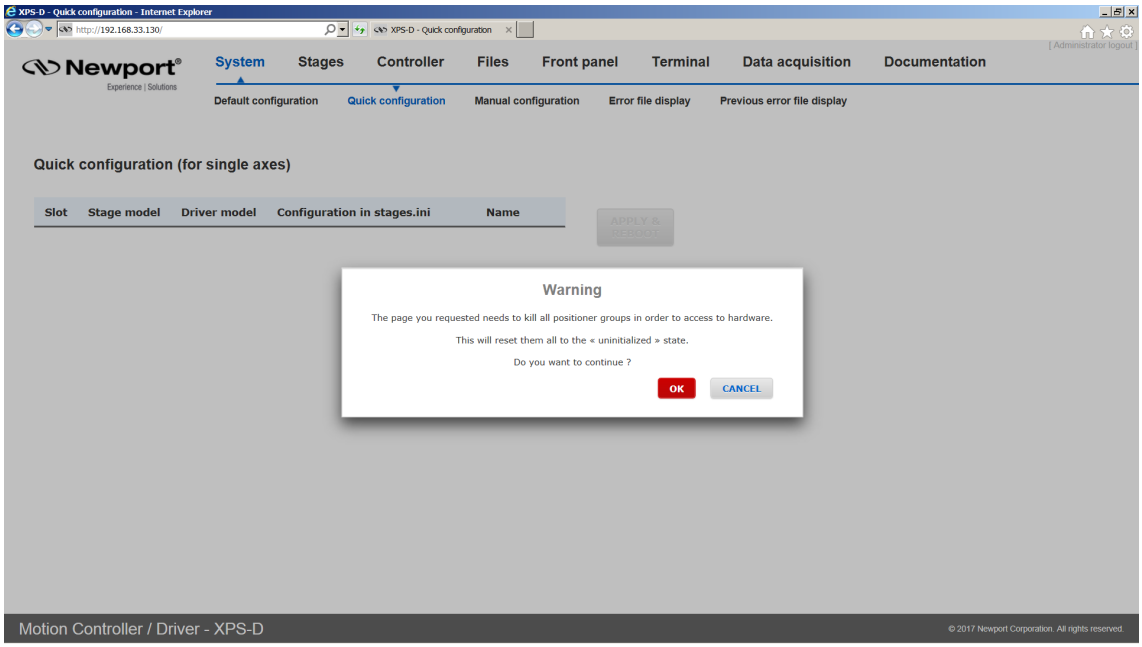
In “DEFAULT-CONFIGURATION” the default group type is set as SingleAxis. To set the positioners to a different group type, use manual configuration.

3.0 Quick Configuration

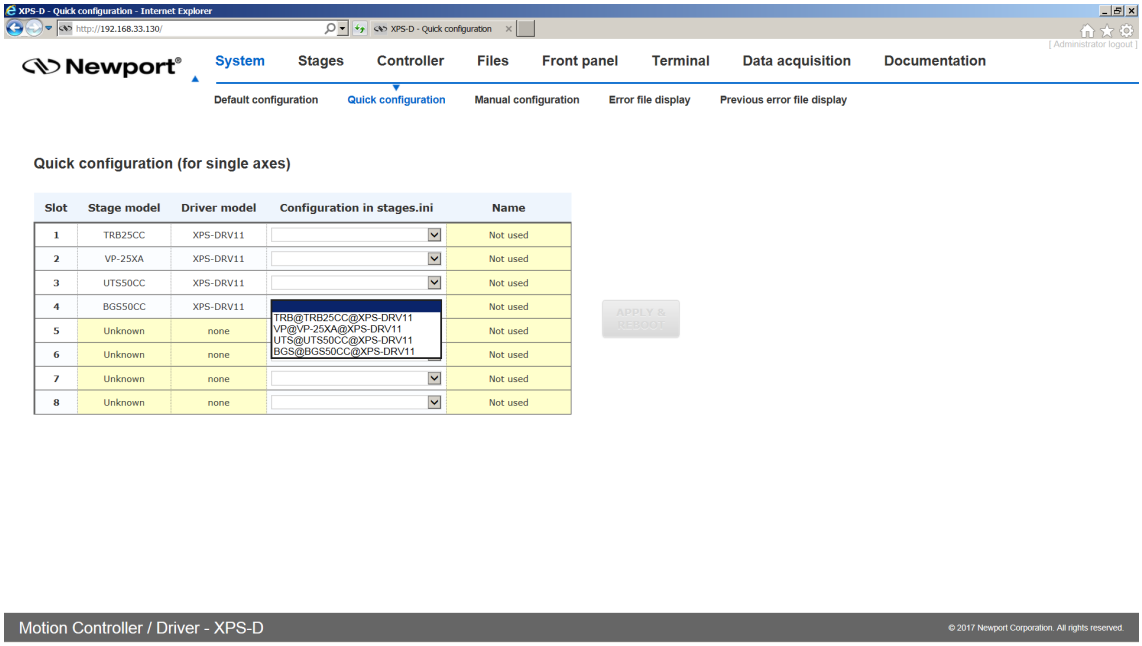
The Quick configuration is very similar to the Default configuration except the stage configuration is taken from the stages.ini file.

3.1 Procedure

- Before using Quick Configuration, it is needed to populate the stages.ini file with the needed configurations, using **Add, remove or edit** stages under the main tab **Stages** (see section **4.1 Adding Newport Stages**).
- Then, when logged in as Administrator, select **SYSTEM**, then **Quick configuration**. The following screen appears:

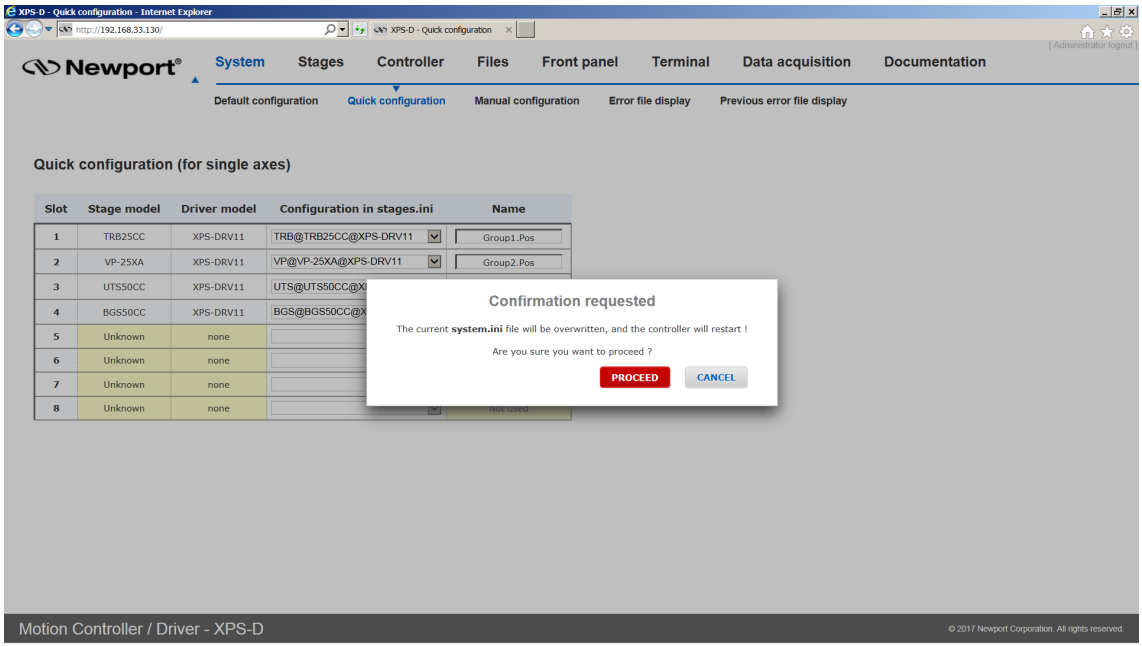


- If you want to continue, click the "OK" button and the following page appears:



The Quick configuration also lists all detected hardware including Newport ESP compatible stages and motor drivers under the respectively columns Stage Model and Driver model.

- Check, if all connected stages are recognized by the system under Stage model. Use the drop-down menu to select the stage configuration for the selected axis. The drop-down menu lists stage configuration(s) available from the stages.ini file stored on the XPS controller.
- If all stages are recognized and after selecting the stage parameters, click **"APPLY & REBOOT"** and the following page appears:



- To configure the XPS click "PROCEED." The controller reboots and the Login screen appears (this may take up to 1 min).

NOTE

"APPLY and BOOT" deletes your current system.ini configuration files. For diagnosing or troubleshooting a system, make sure to first backup the original system.ini of the system.

- When the controller has finished booting, login, select **FRONT PANEL** and then select **"Move"**.

Your system is now ready to use. For more advanced functions, please read the rest of this manual.

NOTE

In "Quick-configuration" the default group type is SingleAxis. To set the positioners to a different group type, use "Manual configuration".

4.0 Manual Configuration for Newport Positioners

Manual configuration provides users access to all capabilities of the XPS controller.

For manual configuration, users first need to build the stage database using the web tool "**Add, remove or edit stages**" under the main tab **Stages**. When adding a new stage from this web tool, the controller copies the parameters from its internal database (which contains parameters for all Newport stages) and stores these parameters in a file called stages.ini. Hence, the stages.ini file contains the parameters for only a subset of stages as defined by the user. Users can assign any name for their stages. The default name is the Newport part number, but in some cases, it makes sense to use a different name. This way, for instance, it is possible to add the same set of parameters several times in the stage database under different stage names. Later, you can modify certain parameters, like travel ranges or PID settings, to optimize the stage for different applications.

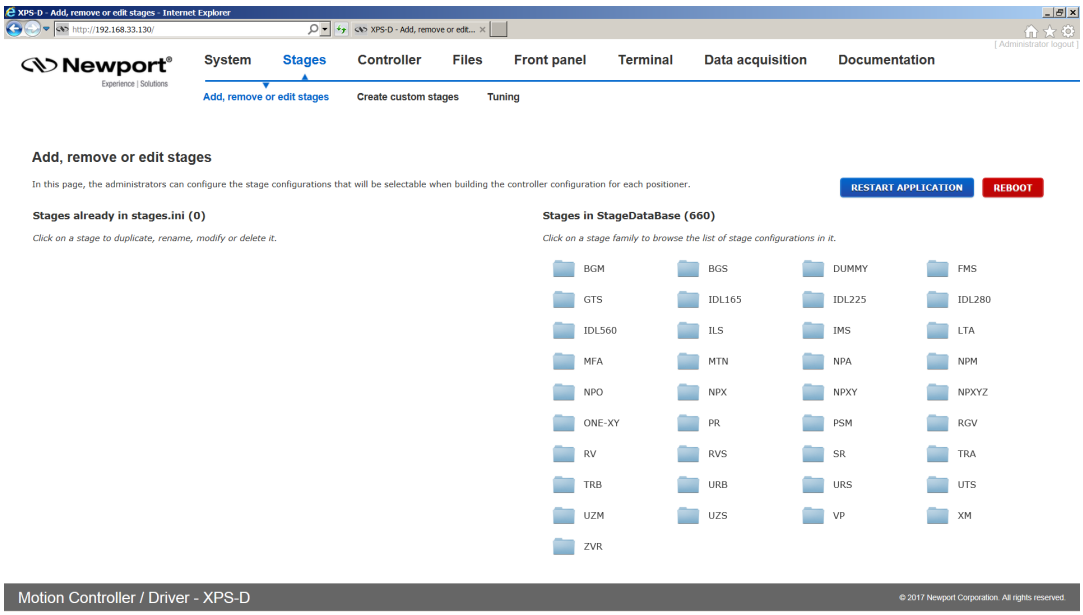
All stage parameters can be modified using the Web Tool **"Add, remove or edit stages"** under the main tab **Stages**. Click on a stage to duplicate, rename, modify or delete it. Another Web Tool for modifying stage parameter can be found under Files → Configuration files using the text editor (see User Interface Manual for details). Alternatively, the stage parameters can be modified directly in the stages.ini file using a text editor. The stages.ini file is located in the Config folder of the XPS controller. This folder is accessible via ftp, see User Interface Manual for details.

When all stages are added to the stages.ini file, build the system using the web tool "**Manual Configuration**" under the main tab **SYSTEM**. In this tool, the stages get assigned to positioners and the positioners get assigned to motion groups. Please refer to Features Manual for details on the different motion groups and their specific features. The group name and positioner name can be any user given name. Once the system has been built, all system information is stored in a file called system.ini. Also, the system.ini file is located in the Config folder of the XPS controller and can be viewed or edited from the Web Tool text editor under **Files → Configuration files** (see User Interface Manual for details).

The following describes the different steps needed to add a stage, to modify the stage parameters and to build a manual configuration.

4.1 Adding Newport Stages

- Once you are logged in as Administrator, click on **Stages** and then click on **"Add, remove or edit stages"**. The following screen appears:

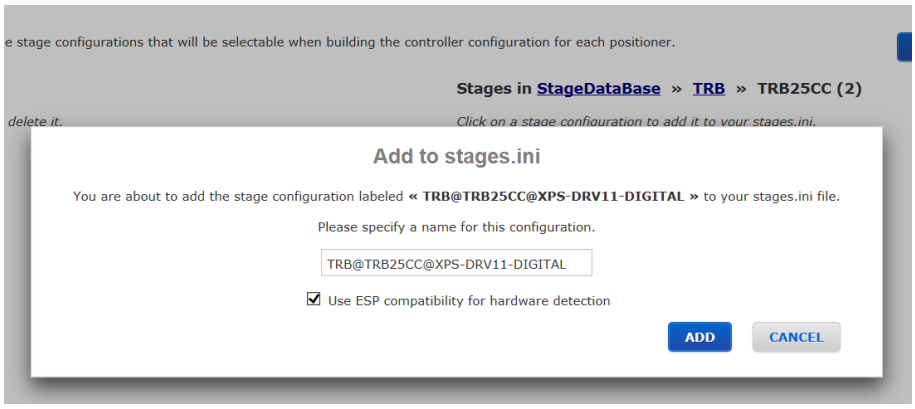


- Click on the family name from the “**Stages in StageDataBase”** list and its content appears.
- Click the part number corresponding to your hardware.
- Select the driver (corresponding to your hardware) and configuration.

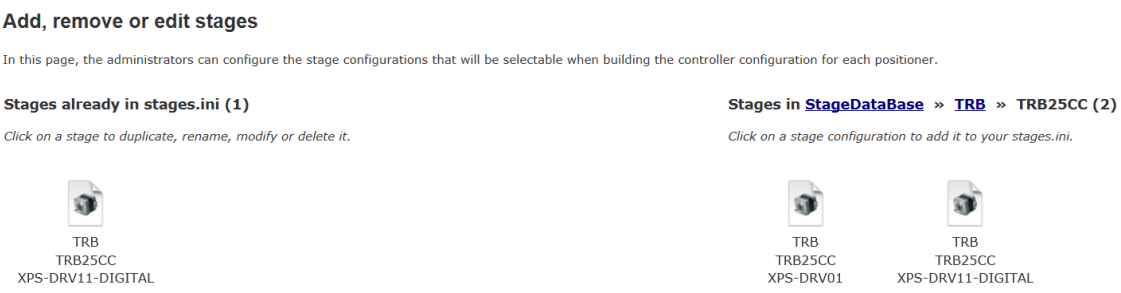
For all continuous rotation stages, you can choose between a “regular” stage configuration and a “Spindle” configuration. A Spindle is a specific rotary device (no indexing) with a periodic position reset at 360° (by default), meaning 360° equals 0°. When defining the stage as Spindle in the stages.ini, you must assign this stage also to a Spindle group in the system configuration and vice versa. For details about Spindles, please refer to Features Manual.

For some stages, you can choose between "regular" initialization or LMI (Large Move Initialization). The LMI method produces a larger movement of the stage for commutation and could be used if "regular" initialization fails.

- The following window appears:



- The box “Use ESP Compatibility for Hardware detection” is checked by default. If your stage has an ESP chip inside (see the ESP-compatible sticker on the stage) this box should remain checked. Otherwise, with vacuum compatible stages or with old Newport stages, or with non-Newport stages, uncheck this box.
- Click on **“ADD”** to add the stage to the stages.ini file. The new configuration appears in the “Stages already in stages.ini” area:

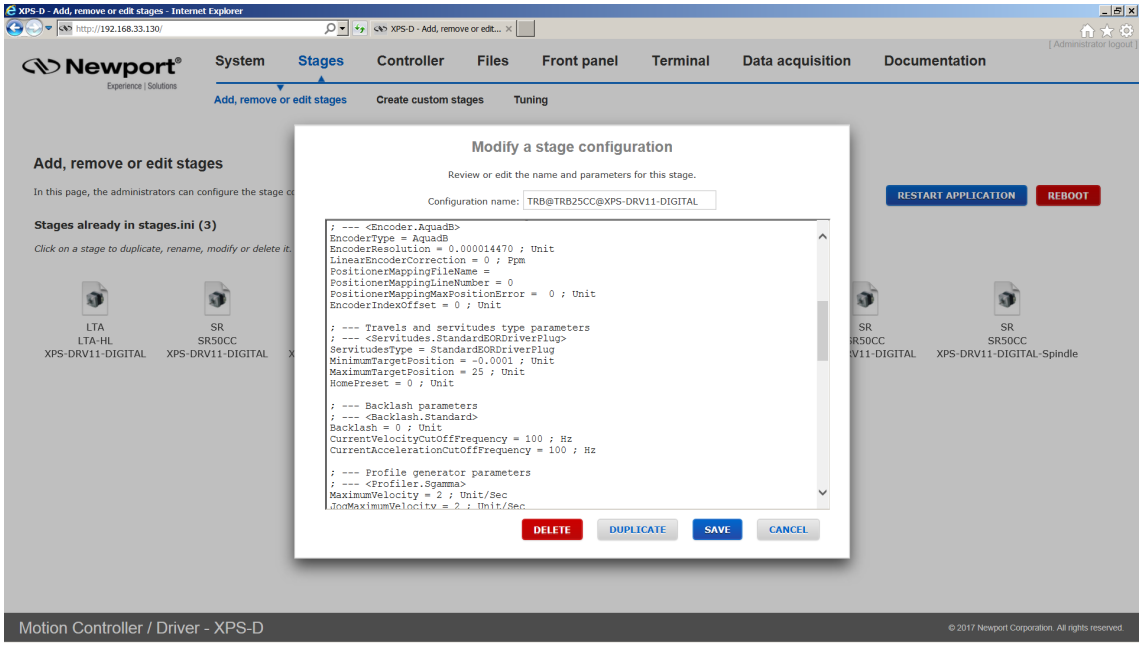


- Click on “StageDataBase” (or on the family name) to select the following stage configuration

4.2 Modifiying Stage Parameters

Once all stages have been added to the stages.ini file, you can review or modify these parameters:

- Click on a stage from the list under **“Stages already in stages.ini”.** A window appears which allows the user to modify the stage in the stages.ini file.
- Scroll down to the section that contains the parameters that will be modified. Parameters commonly changed, are the minimum and the maximum target positions of a rotation stage. For example, to enable larger rotations of a rotation stage that is not configured as a Spindle, set the maximum target position to a very high value and the minimum target position to a very low value. In this case it is also required to disable the limit switches of the rotation stage, see stage manual for details.



1. When done, click "Save" to apply the new values, or click “Cancel” if a mistake was made.
2. To take the new values into account, reboot the controller or use “Restart Application” button.

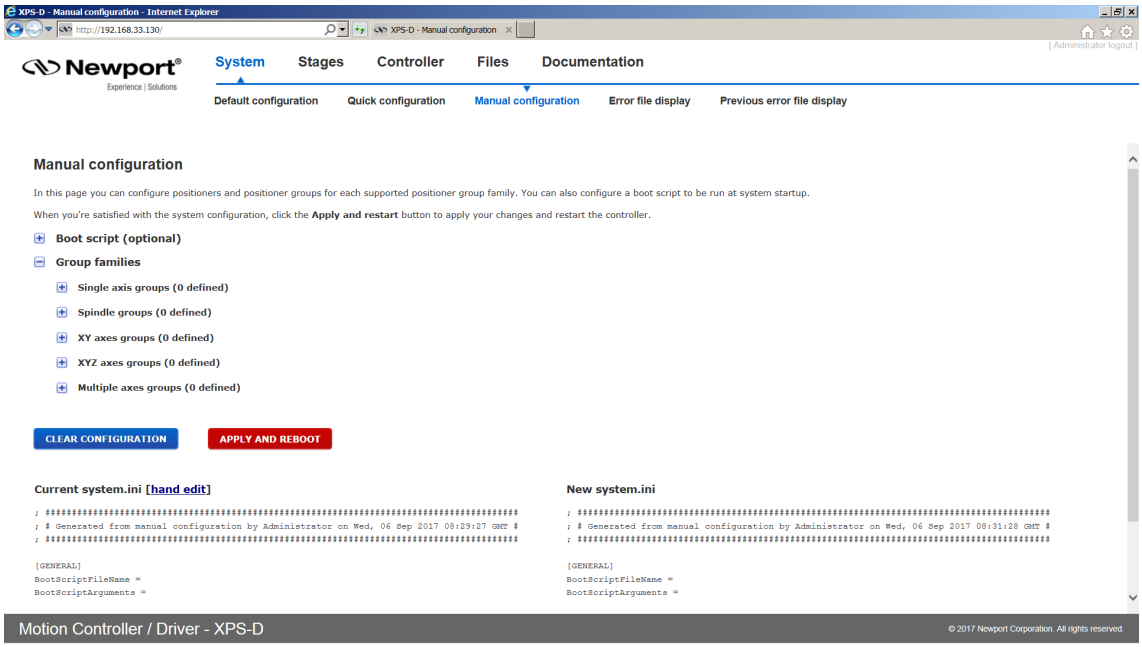
The same screen allows duplicating stages in the stages.ini (in most case some parameters are modified as a second step) or to delete stages from the stages.ini.

NOTE

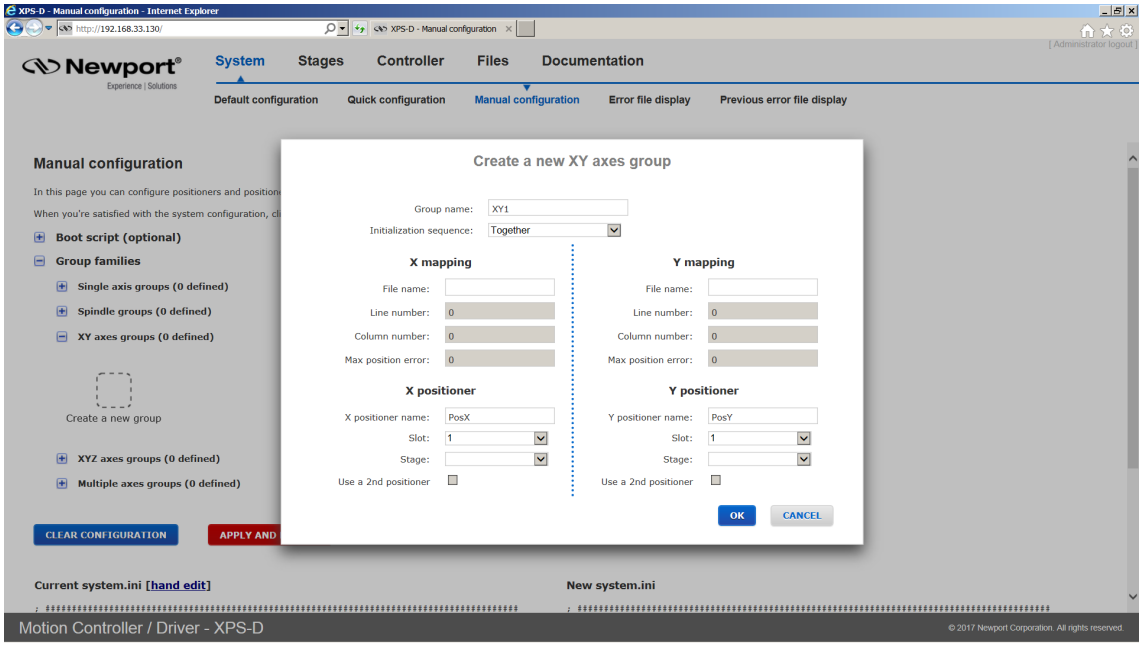
From this screen, you have access to all stage parameters. Only experienced users should modify these parameters. For the exact meaning of the different parameters, please refer to chapter 5.0: “Manual Configuration for Non Newport Stages”.

4.3 Manual Configuration

- When all stages have been added to the stages.ini file, click on **"Manual Configuration"** under **SYSTEM**. The following screen appears:

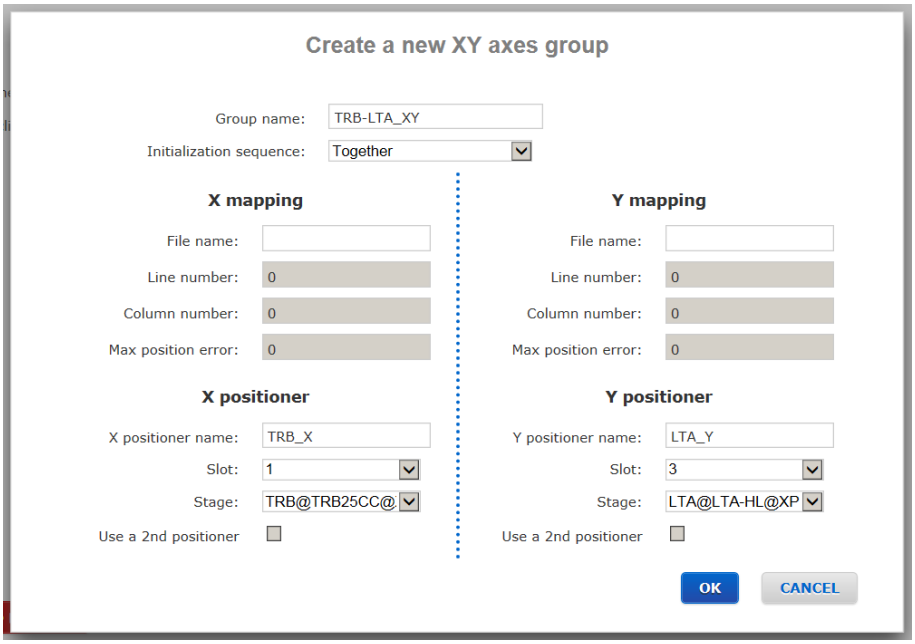


- Click on a Group type. For example, if you are setting up two ILS stages, you can set them up as two Single axis groups, one XY axes group or one or two Muliple axes groups.
- For example, click on **"Create a new group"** of "**XY axes groups**"; the following pop up window appears:

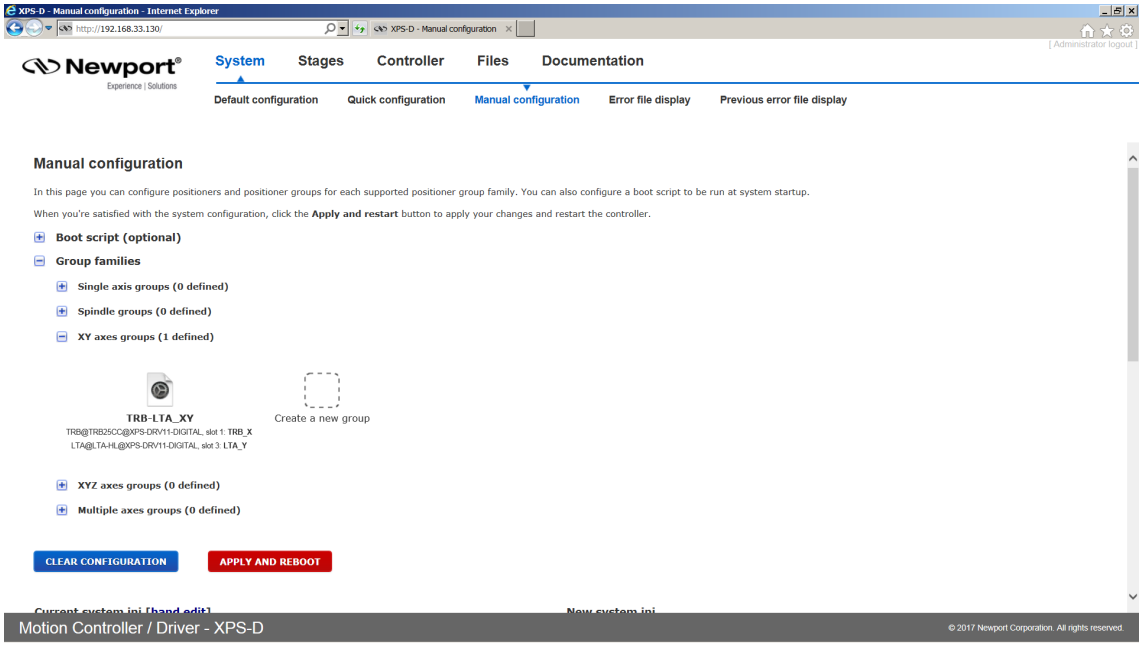


- Enter the group name. In the example, the group name is TRB-LTA_XY.
- Define the home sequence ("Together" or "XThenY" or "YThenX").
- For error compensation, define the name and structure of the correction data, otherwise leave these fields blank. For details about error compensation, see Features Manual.
- Enter the positioner names in this example the positioner names are TRB_X and LTA_Y.

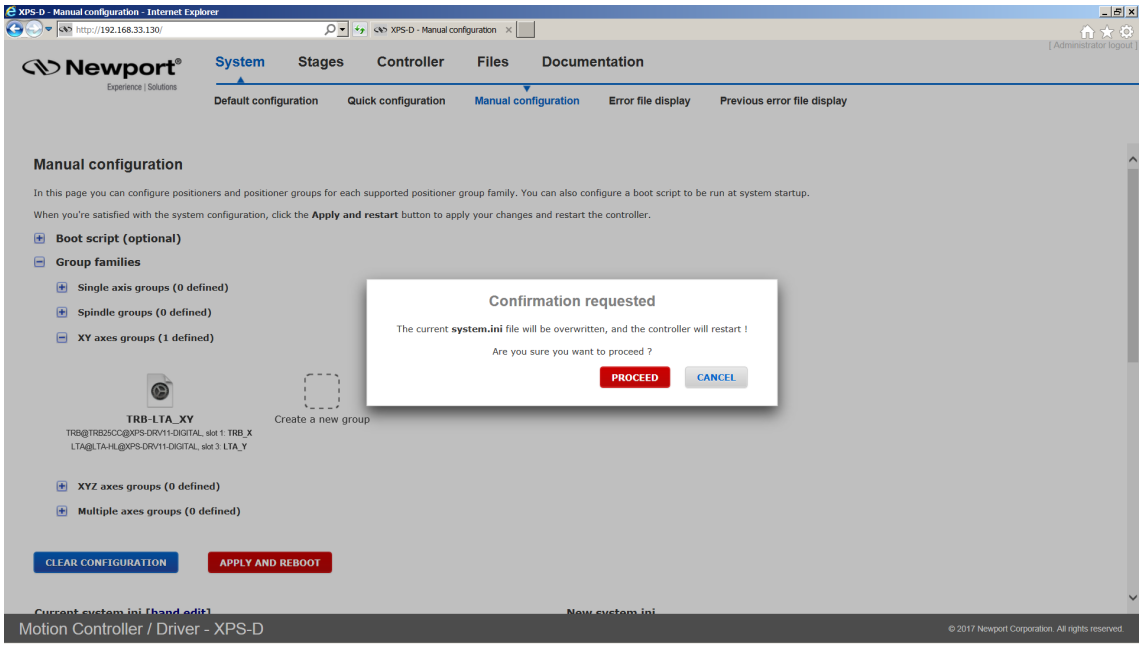
- For each positioner select the Slot number. The Slot number is the axis number where the stage is physically connected to the XPS controller. Looking at the rear of the controller, plug number 1 is the right plug and the number increases to the left.
- Select the StageName from the list of stages. These stage names refer to the stages previously defined with the **"Add, remove or edit stages"** page.



- Click on **"OK"** to return to the initial screen.



- Continue the same way with the other motion groups.
- When done, click on **"APPLY AND REBOOT"** to complete the System configuration. The following message appears:



- To configure the XPS click “PROCEED.” The controller reboots and the Login screen appears (this may take up to 1 min).

NOTE

“APPLY and BOOT” deletes your current system.ini configuration files. For diagnosing or troubleshooting a system, make sure to first backup the original system.ini of the system.

- When the controller has finished booting, login, select the **SYSTEM** tab, then **“Error File Display”**. When there is no entry in the error file, your system is configured correctly and ready to use. If not, this file provides some valuable information for troubleshooting; see also “Troubleshooting” chapter in “Maintenance and Service” section of Start-Up Manual.

Below is an example of a system.ini file with one XY group and one Spindle group:

```
[GENERAL]
BootScriptFileName =
BootScriptArguments =

[GROUPS]
SingleAxisInUse =
SpindleInUse = Spin
XYInUse = TRB-LTA_XY
XYZInUse =
MultipleAxesInUse =

; ##############################################
; # Spindle group 'Spin' and its positioner: Rot

[Spin]
PositionerInUse = Rot
```

```
[Spin.Rot]
PlugNumber = 4
StageName = SR@SR50CC@XPS-DRV11-DIGITAL-Spindle

;
##########################################################
######
; # XY axes group 'TRB-LTA_XY' and its positioners: TRB_X, LTA_Y

[TRB-LTA_XY]
PositionerInUse = TRB_X, LTA_Y
InitializationAndHomeSearchSequence = Together
XMappingFileName =
XMappingLineNumber = 0
XMappingColumnNumber = 0
XMappingMaxPositionError = 0
YMappingFileName =
YMappingLineNumber = 0
YMappingColumnNumber = 0
YMappingMaxPositionError = 0

[TRB-LTA_XY.TRB_X]
PlugNumber = 1
StageName = TRB@TRB25CC@XPS-DRV11-DIGITAL
SecondaryPositionerGantry = Disabled

[TRB-LTA_XY.LTA_Y]
PlugNumber = 3
StageName = LTA@LTA-HL@XPS-DRV11-DIGITAL
SecondaryPositionerGantry = Disabled
```

5.0 Manual Configuration for Non Newport Stages

The Manual Configuration for non Newport stage is also achieved through the **"Manual Configuration"** under **SYSTEM** tab (see previous section) but since these custom stages are not included in the XPS general stage database, they must be entirely created and configured in the "stages.ini" file.

The chapter will present all possible configurations of the XPS controller with regards to drive and control capabilities.

The XPS controller uses two configuration files, named "system.ini" and "stages.ini". The files are stored on the XPS controller and are accessible through the XPS webpage **Files → Configuration files**. These configuration files are read during the boot sequence of the controller. The "system.ini" file specifies the system configuration and the configured motion groups. The "stages.ini" file defines the parameters for all positioners.

The aim of this chapter is to provide a better understanding of the drive and control capabilities of the XPS controller and possible settings in the "stages.ini" file. It is important that users have a working understanding of the structure of this file, because it may be necessary to make modifications to the default parameters to access all features of the XPS controller. In order to configure the XPS controller to drive non-Newport stages, it is important that users have an in-depth understanding of this file and the meaning of the included entries.

In each subsection we provide a detailed definition of each parameter, the physical meaning and one example to set it. Since there are many options available for configuration, not all strategies and methods are detailed. Especially for the definition of the tuning parameters (PID, filters, etc.). Please refer to supporting literature for an in-depth treatment of tuning.

IMPORTANT NOTE ABOUT THE UNITS

The XPS controller accepts any dimension for the position unit such as: mm, inch, µm, deg, rad, etc. In this documentation, the generic term "unit" is used for the position unit. This generic unit is carried forward into units that reference the position unit, for example speed and acceleration would carry units such as: units/s or units/s². The physical dimension assignment of the position unit for closed-loop systems is done by *stage displacement per encoder count* as part of the parameters of the *position encoder interface.* For open-loop systems the physical dimension assignment is done as part of the parameter settings for the driver command interface; examples include: *stage displacement per motor full step* or *command voltage at minimum target position.* It is important to note that the position unit in the configuration files will determine the values of all derived parameters. Therefore, the choice of the position unit will impact most parameters.

5.1 Creating Custom Stages

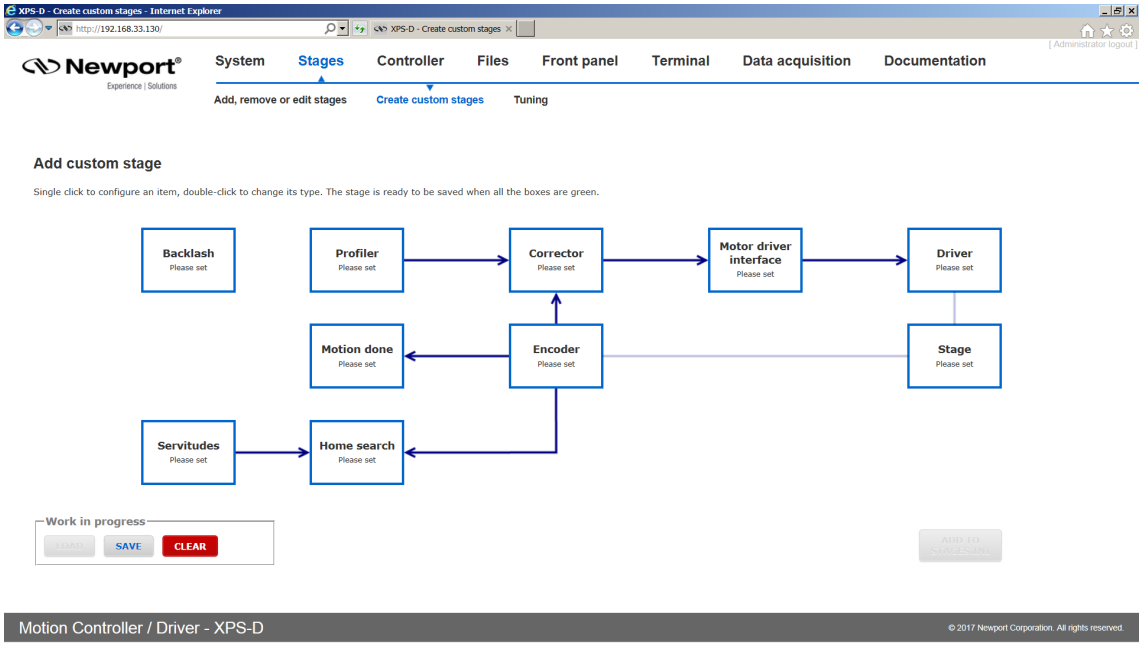
The integrated web tool, **Stages → Create custom stages,** is accessible when logged in as administrator. This web tool is designed to help users configure the XPS controller for motors and stages that are not included in the XPS general stage database such as stages not manufactured by Newport. The tool generates a new entry in the customer's stage database, *stages.ini*, which is stored on the controller and is accessible through the webpage **Files → Configuration files**.

To generate a new stage configuration using the web tool, users must complete an entry for each configuration category, following the arrow sequence where applicable. The software dynamically updates the next category list based on compatibility to prior selections so that only correct settings for the determined configuration are available. This avoids configurations that are not supported by the XPS controller.

NOTE

This web tool cannot compensate for configurations that are not compatible with the connected hardware such as stages with external amplifiers (example using XPS-DRV00P drive).

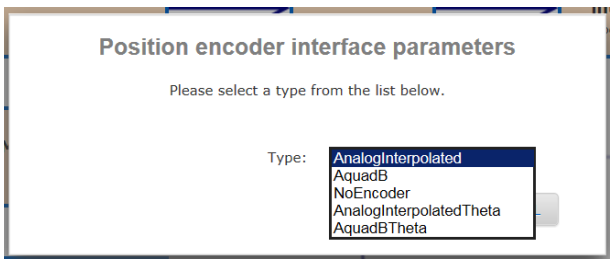
- Once you are logged in as Administrator, click on **Stages** and then click on **"Create custom stages"**. The following screen appears:



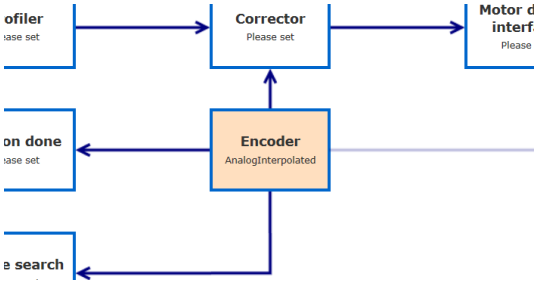
5.1.1 First Level Parameters

The first level settings must be set following the arrows such as defining **Encoder** before **Corrector**, **Motion done** or **Home search**.

- To enter the first level settings for a configuration category, click on the category to get a list of the available types. These settings are presented in the following sections.
  Example with Encoder first level setting:



- Once the first level of setting is entered, the configuration category changes to a light pink color.

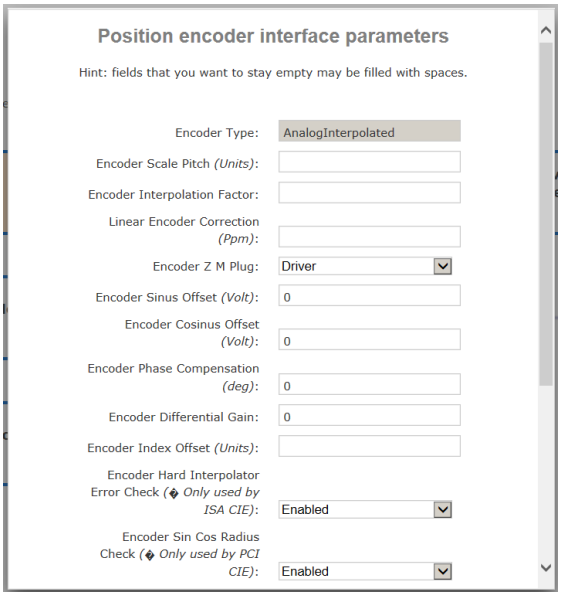


5.1.2 Second Level Parameters

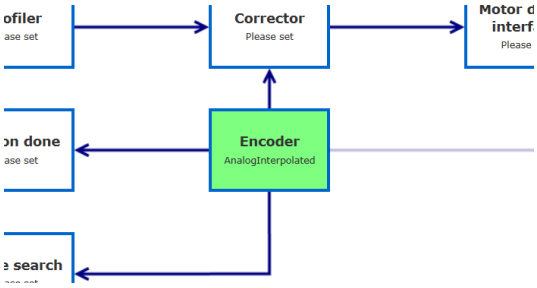
- Once the first level setting for a configuration category is selected, click on the category again.
- A new page opens prompting the parameter values for the second level setting.

The second level settings define all the details for the configuration category. These settings can be done in any order. The type and number of parameters in each of these screens will depend on the first level setting.

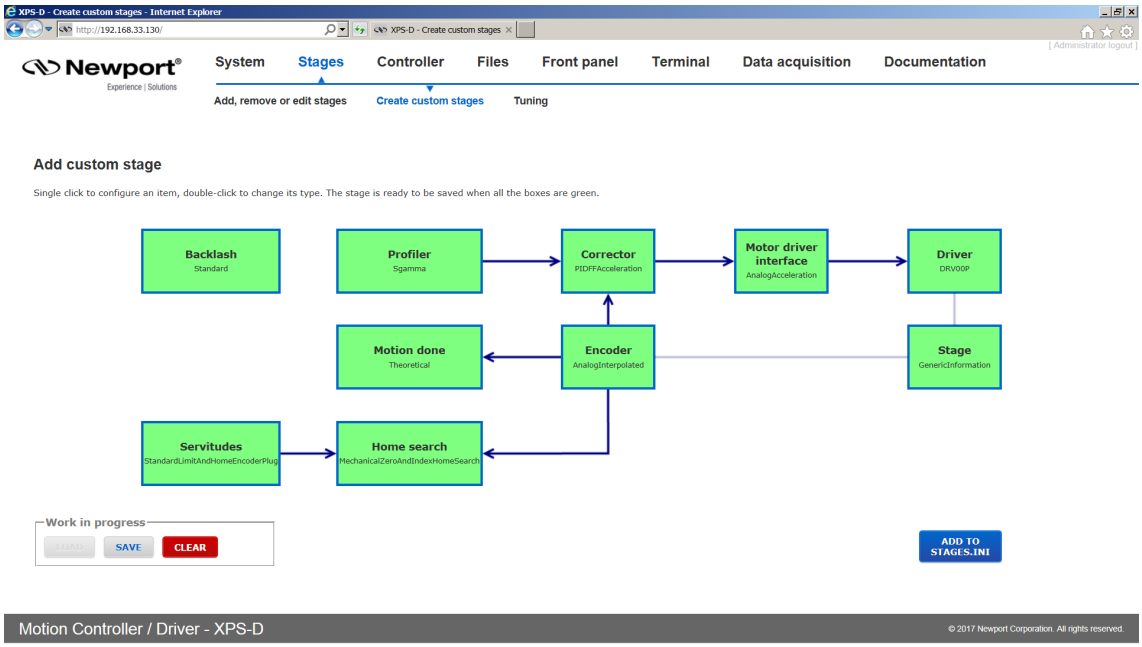
- Define all settings and press the OK button in the bottom of the list to apply them. Example with Encoder second level settings:



- Once the second level settings for a category are set, the category is complete and changes to green color.



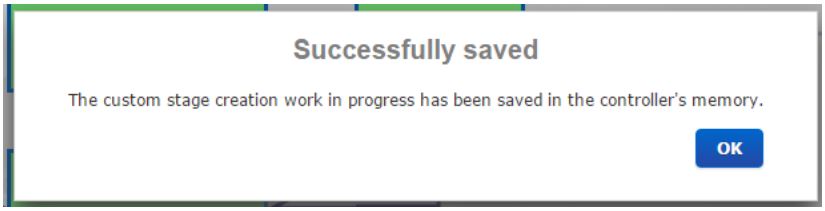
- When all second level settings are complete and green, the stage configuration parameters are set. An "**ADD TO STAGE.INI**" button appears in the lower right-hand corner of the screen.



5.1.3 Save and Configure the Stage

Save Option 1

- Users can save their progress at any time by clicking on the "**SAVE**" button in the lower left-hand corner of the screen.
- A screen appears confirming the progress is stored in the controller's memory.

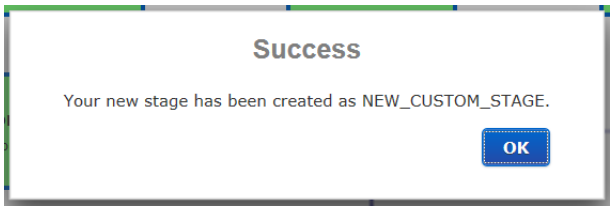


- From here users can move to other webpages and return to **Stages → Create custom stages** to continue the configuration wizard process.

Save Option 2

- When all second level settings are complete and green, an "**ADD TO STAGE.INI**" button appears in the lower right hand corner of the screen.
- Press the "**ADD TO STAGE.INI**" button; the following window appears. Change the default stage name "NEW_CUSTOM_STAGE" if desired and click "**SAVE**". The new stage configuration is saved into the "**stages.ini**" file under the specified name.

- After the stage is added to the stages.ini file a confirmation window appears.



Configure the stage

- After the confirmation that the new stage has been created as "NEW_CUSTOM_STAGE", users can find the stage under **Stages → Add, remove, or edit stages**. The "NEW_CUSTOM_STAGE", once physically connected can now be configured as described in previous sections.

5.2 Encoder

In this configuration category, users are building the Position encoder interface parameters section of the stages.ini file:

Example:

```
; --- Position encoder interface parameters
; --- <Encoder.AquadB>
EncoderType = AquadB
EncoderResolution = 0.001 ; Unit
LinearEncoderCorrection = 0 ; Ppm
PositionerMappingFileName =
PositionerMappingLineNumber = 0
PositionerMappingMaxPositionError = 0 ; Unit
EncoderIndexOffset = 0 ; Unit
```

The XPS controller supports 4 types of position encoder interfaces:

- AquadB differential
- AquadB differential THETA
- Analog Interpolated with sine/cosine 1 Vpp
- Analog Interpolated THETA with sine/cosine 1 Vpp THETA

5.2.1 No Encoder (NoEncoder)

Encoder Type: NoEncoder

This encoder type is used when there is no position feedback from an encoder.

There are no additional parameters to set for this encoder type.

5.2.2 RS422 Differential (AquadB)

Encoder Type: AquadB

This encoder type is used when the position sensor delivers two square waves RS422 A differential signals.

Encoder Resolution

The stage displacement per encoder count, *EncoderResolution* (units), sets the resolution of the position encoder. It must be greater than zero.

NOTE

The encoder resolution is equal to 4-times the signal period (quadrature effect).

The *Stage displacement per encoder count* essentially defines the measurement units of the stage. Many parameters are derived from this value, in particular all parameters with units of length, such as velocities or accelerations. Therefore, it is critical this parameter be set correctly for proper operation.

To calculate the *Stage displacement per encoder count* value all of the following must be taken into account: the number of steps per revolution, screw pitch and any gear reduction in the stage.

Linear Encoder Correction

$$LinearEncoderCorrection = \left(\frac{Real\ increment}{Rounded\ increment} - 1\right) \cdot 1{,}000{,}000$$

The linear correction, *LinearEncoderCorrection* (parts per million), sets the correction applied to the *EncoderResolution* to compensate for linear error effects (see **Features Manual**). This value must be between ±0.5*106. A zero value disables this feature.

The default value is zero.

Encoder Index Offset (In units).

The EncoderIndexOffset is used with the fast PCO functions. It allows defining a preset value to extend the PCO position register when the travel of the stage is larger than the register range. This parameter should be used with care.

(Optional- three entries) Positioner mapping parameters.

Positioner Mapping File Name

The *PositionerMappingFileName* defines the name of the mapping file used for the positioner mapping compensation. No entry for the file name disables the feature.

Positioner Mapping Line Number

The *PositionerMappingLineNumber* defines the number of data lines in the positioner mapping file. This value must be greater than or equal to 3 and less than 200. This parameter is primarily used as a check to confirm the correctness of the mapping file.

Positioner Mapping Max Position Error

The *PositionerMappingMaxPositionError* (units) defines the maximum absolute value of the error corrections in the mapping file. This parameter is primarily used as a check to confirm the correctness of the mapping file.

Please refer to **Features Manual** for further information about the positioner mapping functionality.

5.2.3 RS422 Differential with 3 Encoders (AquadBTheta) PP Version Only

Encoder Type: AquadBTheta

This encoder type is composed of three encoders and is used when the position sensor delivers two square waves RS422 A differential signals.

Encoder Radius

The theta encoder radius, *EncoderRadius (XY *10^{-6}),* is the radius of the theta (three encoders)

Maximum Encoder Correction X

The *MaximumEncoderCorrectionX* sets the maximum allowed correction for the X positioner.

Maximum Encoder Correction Y

The *MaximumEncoderCorrectionY* sets the maximum allowed correction for the Y positioner.

For all other configuration file parameters refer to section 5.2.2: "RS422 Differential (AquadB)".

5.2.4 Sine/Cosine 1 Vpp (AnalogInterpolated)

Encoder Type: AnalogInterpolated

This encoder type is used when the position sensor delivers two 1 Vpp sine/cosine signals that are interpolated by the XPS controller.

For more information on this refer to **Features Manual**.

Encoder Scale Pitch

The stage displacement per encoder period, *EncoderScalePitch* (units), sets the displacement of the stage per encoder period. This parameter essentially defines the measurement units of the stage. Many parameters are derived from this entry, in particular all parameters with units of length, such as velocities or accelerations. Therefore it is critical this parameter value is set correctly for proper operation of the stage.

Encoder Interpolation Factor

The Encoder signal subdivisor, *EncoderInterpolationFactor*, sets the interpolation factor for the encoder signal interpolation. This value defines the resolution of the CurrentPosition and SetpointPosition of the stage as follows:

$$Resolution = EncoderScalePitch/EncoderInterpolationFactor$$

For a better understanding of the meaning of the CurrentPosition and SetpointPosition see **Features Manual**.

The *EncoderInterpolationFactor* must be an integer value greater than or equal to 1 and less than or equal to 65536.

The *EncoderInterpolationFactor* should be set relative to the position noise of the stage. It is not recommended to set a position resolution that is far less than the amplitude of the position noise. It is also important to note that the value of the *EncoderInterpolationFactor* has no impact on the resolution of the position used for the position servo loop. The position servo loop always uses calculations at the maximum position resolution possible. For example, (EncoderScalePitch/65536).

Linear Encoder Correction

Stages.ini file entry: *LinearEncoderCorrection*

LinearEncoderCorrection = ((Real increment/Rounded increment) - 1) * $1e^6$

The linear correction, *LinearEncoderCorrection* (parts per million), sets the correction applied to the *EncoderResolution* to compensate for linear error effects (see **Features Manual**). This value must be between $\pm 0.5*10^6$. A zero value disables this feature.

$$CorrectedResolution = (1 + LinearCorrection \times 10^{-6}) \times EncoderResolution$$

The default value is zero.

Encoder ZM Plug

Stages.ini file entry: *EncoderZMPlug*

The mechanical zero input plug, *EncoderZMPlug*, defines where the mechanical zero signal is input. There are two possible settings:

- *Driver* — for mechanical zero through the driver board plug.
- *Encoder* — for mechanical zero through the encoder driver board plug.

Encoder Sinus Offset & Encoder Cosine Offset

Needed entries in the configuration file:

- sine channel offset correction: *EncoderSinusOffset.*
- sine channel offset correction: *EncoderCosinusOffset.*

The sine/cosine channel offset corrections, *EncoderSinusOffset* (V) and *EncoderCosinusOffset* (V), set the offset values of the two encoder signals for correction. They must be between ±0.1 V. A zero value disables this feature.

The default value is zero.

Encoder Phase Compensation

Stages.ini file entry: *EncoderPhaseCompensation*

The signal phase correction, *EncoderPhaseCompensation* (°), sets the phase correction of the cosine signal phase. This correction is applied after the differential amplitude correction. This value must be between ±10°. A zero value disables this feature.

$$PhaseCorrectedCosine = \frac{Sine \times \sin\left(PhaseCompensation \times \frac{\pi}{180}\right) + CorrectedCosine}{\cos\left(PhaseCompensation \times \frac{\pi}{180}\right)}$$

The default value is zero.

Encoder Differential Gain

Stages.ini file entry: *EncoderDifferentialGain*

The amplitude correction, *EncoderDifferentialGain*, sets the amplitude correction of the cosine signal amplitude. It must be between ±0.1. A zero value disables this feature.

$$CorrectedCosine = (1 + EncoderDifferentialGain) \times Cosine$$

The default value is zero.

Encoder Index Offset

- In units.

The EncoderIndexOffset is used with the fast PCO functions. It allows defining a preset value to extend the PCO position register when the travel of the stage is larger than the register range. This parameter should be used with care**.**

Encoder Hard Interpolator Error Check

Not Applicable — *Only for XPS-Q8*.

Encoder Sin Cos Radius Check

- Enabled or Disabled.

The purpose is to verify in real time that the Lissajou radius is not too small or too high for the interpolation denoting a problem with the stage 1 Volt peak to peak sin cos encoder. The limits are (0.6 to 1.2 V) for a good signal; out of that range, a positioner error is generated**.**

Current Position Filter Select

Frequency of the filter applied on the servo loop position. This filter does not limit the encoder bandwidth because it is applied after sin/cos interpolation however it generates a latency on the position. The ideal value for Curent Position Filter is 3.11. It is the lowest frequency with a latency compatible with the corrector frequency. See table below for more information.

Possible values for Current Position Filter Frequency or PCO Position Filter Frequency with associated latency:

Parameter Value	Filter Frequency	Position Latency
5000	No filter	
600	600 kHz	210 ns
230	230 kHz	540 ns
110	110 kHz	1.14 µs
49.74	49.74 kHz	2.5 µs
24.87	24.87 kHz	5 µs
12.43	12.43 kHz	10 µs
6.22	6.22 kHz	20 µs
3.11	3.11 kHz	40 µs
1.55	1.55 kHz	80 µs
0.78	0.78 kHz	160 µs

PCO Position Filter Select

The ideal value for Curent Position Filter is 3.11. It is the lower frequency with a latency compatible with the corrector frequency. Replace:

Frequency of the filter applied on the position used for PCO, External Gathering and AquadB output

This filter does not limit the encoder bandwidth because it is applied after sin/cos interpolation however it generates a latency on the position. The ideal value for PCO Position Filter depends on the application. See table above for more information.

(Optional- three entries) Positioner mapping parameters.

Positioner Mapping File Name

The *PositionerMappingFileName* defines the name of the mapping file used for the positioner mapping compensation. No entry for the file name disables the feature.

Positioner Mapping Line Number

The *PositionerMappingLineNumber* defines the number of data lines in the positioner mapping file. This value must be greater than or equal to 3 and less than 200. This parameter is primarily used as a check to confirm the correctness of the mapping file.

Positioner Mapping Max Position Error

The *PositionerMappingMaxPositionError* (units) defines the maximum absolute value of the error corrections in the mapping file. This parameter is primarily used as a check to confirm the correctness of the mapping file.

Please refer to **Features Manual** for further information about the positioner mapping functionality.

Example:

```
; --- Position encoder interface parameters
; --- <Encoder.AnalogInterpolated>
EncoderType = AnalogInterpolated
LinearEncoderCorrection = 0 ; Ppm
EncoderZMPlug = Encoder
EncoderInterpolationFactor = 4000
EncoderScalePitch = 0.004 ; Unit
EncoderSinusOffset = 0 ; Volt
EncoderCosinusOffset = 0 ; Volt
EncoderPhaseCompensation = 0 ;--- deg
EncoderDifferentialGain = 0
PositionerMappingFileName =
PositionerMappingLineNumber = 0
PositionerMappingMaxPositionError = 0 ; Unit
EncoderIndexOffset = 0 ; Unit
EncoderHardInterpolatorErrorCheck = Enabled ; # Only used by ISA CIE
EncoderSinCosRadiusCheck = Enabled ; # Only used by PCI CIE
CurrentPositionFilterSelect = 3.1 ; kHz # Only used by PCI CIE
PCOPositionFilterSelect = 5000 ; kHz # Only used by PCI CIE
```

5.2.5 Sine/Cosine 1 Vpp (AnalogInterpolated Theta) PP version only

Encoder Type: AnalogInterploatedTheta

This encoder type is composed of three encoders and is used when the position sensor delivers two 1 Vpp sine/cosine signals that get interpolated by the XPS controller.

Encoder Radius

The theta encoder radius, *EncoderRadius (XY *10^{-6}),* is the radius of the theta (three encoders).

Maximum Encoder Correction X

The *MaximumEncoderCorrectionX* sets the maximum allowed (limit) correction for the X positioner.

Maximum Encoder Correction Y

The *MaximumEncoderCorrectionY* sets the maximum allowed (limit) correction for the Y positioner.

***For all other configuration file parameters refer to section 5.2.4:* "Sine/Cosine 1 Vpp (AnalogInterpolated)".**

5.3 Profiler

In this configuration category, users are building the Profile generator parameters section of the stages.ini file.

Example:

```
; --- Profile generator parameters
; --- <Profiler.Sgamma>
MaximumVelocity = 300 ; Unit/s
JogMaximumVelocity = 300 ; Unit/s
MaximumAcceleration = 2500 ; Unit/s²
JogMaximumAcceleration = 2500 ; Unit/s²
EmergencyDecelerationMultiplier = 4
MinimumJerkTime = 0.02 ; s
MaximumJerkTime = 0.02 ; s
TrackingCutOffFrequency = 25 ; Hz
```

5.3.1 Type: Sgamma

Maximum Velocity

The maximum velocity, *MaximumVelocity* (units/s), sets the maximum velocity the stages can be commanded to move. This value must be greater than zero and less than or equal to the stage velocity at maximum command (see section 5.9.18).

When used with stepper motors and sine/cosine position control the value for the *maximum velocity* must be less than or equal to the *DisplacementPerFullStep* (see section 5.9.17) multiplied by the servo loop frequency. For smooth operation, however, we recommend to not exceed one-quarter of this maximum possible value.

Jog Maximum Velocity

The maximum velocity while in jog mode. *JogMaximumVelocity* (units/s) sets the maximum velocity the stages can be commanded to move in jog mode. This value must be greater than zero and less than or equal to the stage velocity at maximum command (see section 5.9.18).

When used with stepper motors and sine/cosine position control the value for the *jog maximum velocity* must be less than or equal to the *DisplacementPerFullStep* (see section 5.9.17) multiplied by the servo loop frequency. For smooth operation, however, we recommend to not exceed one-quarter of this maximum possible value.

Maximum Acceleration

The maximum acceleration, *MaximumAcceleration* (units/s²), sets the maximum acceleration the stage can be commanded to move. This value must be greater than zero and less than or equal to the stage acceleration at maximum command (see section 5.9.2).

The recommended value for smooth displacement is four times the maximum velocity.

Jog Maximum Acceleration

The maximum acceleration while in jog mode. *MaximumAcceleration* (units/s²) sets the maximum acceleration the stage can be commanded to move in jog mode. This value must be greater than zero and less than or equal to the stage acceleration at maximum command (see section 5.9.2).

The recommended value for smooth displacement is four times the jog maximum velocity.

Emergency Deceleration Multiplier

The emergency deceleration multiplier, *EmergencyDecelerationMultiplier*, defines the ratio between the emergency deceleration during an emergency brake and the normal deceleration during normal brake. This value must be greater than or equal to one.

The default setting is 4.

Minimum/Maximum Jerk Time:

The SGamma profile generator jerk times, *MinimumJerkTime* (s) and *MaximumJerkTime* (s), set the jerk times of the SGamma profile generator. The minimum jerk time must be less than the maximum jerk time and greater than or equal to the profile generator cycle period (0.4 ms).

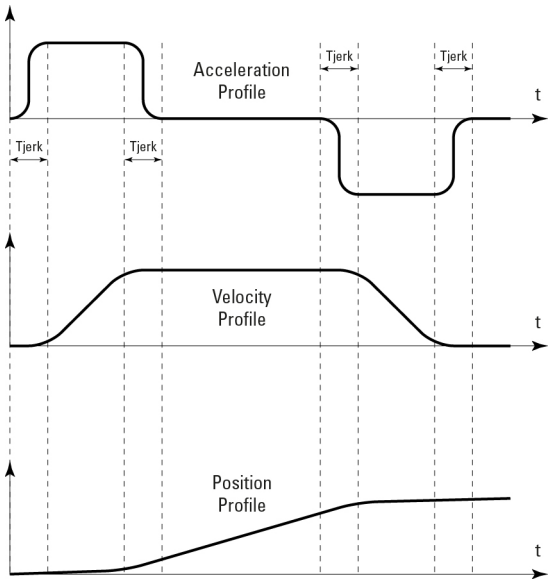


Figure 1: SGamma motion profile.

The recommended values for a smooth displacement are:

- MaximumJerkTime 0.05 s
- MinimumJerkTime 0.005 s

For more information about the SGamma motion profiler, see also **Features Manual**.

Tracking Cut Off Frequency

The tracking mode filter cut off frequency, *TrackingCutOffFrequency* (Hz), sets the cut off frequency of the filter applied to the tracking input. This frequency must be at least ten times less than the servo loop bandwidth to avoid poor tracking. It must be also greater than or equal to zero (disable) and less than or equal to half of the profile generator cycle frequency (2500 Hz).

The default setting is 5 Hz.

5.4 Servitudes

In this configuration category, users are building the Position encoder interface parameters section of the stages.ini file:

Example:

```
; --- Travels and servitudes type parameters
; --- <Servitudes.StandardEORDriverPlug>
ServitudesType = StandardEORDriverPlug
MinimumTargetPosition = -45 ; Unit
MaximumTargetPosition = 45 ; Unit
HomePreset = 0 ; Unit
```

The XPS controller supports 5 settings for the limit sensors input plug:

- No Servitudes
- Driver board
- Encoder board
- None for spindle group
- None for piezo driver

The choice of the setting for the limit sensor input plug depends on where these signals are electrically connected to the XPS controller: either through the plug on the driver board or through the plug on the encoder board. The setting *none for spindle group* is only used for spindle groups.

The limit sensor input plug setting is based on other stage settings, such as: maximum stage travel, maximum drivable speeds and accelerations, etc.

5.4.1 No Servitudes (NoServitudes)

Servitudes Type: NO_SERVITUDES

This setting disregards the end of run detection, even if the signals are present. There are no parameters to set for this servitude type.

Minimum Target Position

The minimum position, *MinimumTargetPosition* (units), sets the minimum allowed position for any move command. This value must be less than the maximum position.

Maximum Target Position

The maximum position, MaximumTargetPosition (units), sets the maximum allowed position for any move command. This value must be greater than the Home Preset.

Home Preset

The home position, *HomePreset* (units), sets the position value of the home reference. This value must be between the minimum position and the maximum position defined above.

5.4.2 Piezo

Servitudes Type: Piezo

This setting can only be used with piezo stages. It disregards the end of run detection, even if the signals are present.

5.4.3 Spindle

Servitudes Type: Spindle

This setting can only be used with stages configured as spindle groups. It disregards the end of run detection, even if the signals are present.

Home Preset

The home position, *HomePreset* (units), sets the position value of the home reference. This value must be greater than zero and less than the *SpindlePeriod.*

Spindle Period

The *SpindlePeriod* (units) sets the period of the spindle rotation. This value must be greater than zero.

5.4.4 Driver Board (StandardEORDriverPlug)

Servitudes Type: StandardEORDriverPlug

This setting is used when the travel limit signals are wired to the XPS controller through the driver board plug.

Minimum Target Position

The minimum position, *MinimumTargetPosition* (units), sets the minimum allowed position for any move command ($Travel_{soft}$-). This value must be less than the maximum position.

Maximum Target Position

The maximum position, *MaximumTargetPosition* (units), sets the maximum allowed position for any move command ($Travel_{soft}$+). This value must be greater than the Home Preset.

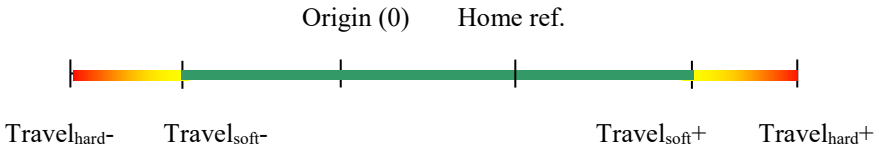


Figure 2: Minimum and maximum travel position.

Home Preset

The home position, *HomePreset* (units), sets the position value of the home reference relative to the origin (zero). This value must be between the minimum position and the maximum position defined above.

5.4.5 Encoder Board (StandardLimitAndLimitEncoderPlug)

Servitudes Type: StandardLimitAndLimitEncoderPlug

This setting is used when **only** the travel limit signals are wired to the XPS controller through the encoder board plug.

For all other configuration file parameters refer to section 5.4.4: "Driver Board (StandardEORDriverPlug)".

5.4.6 Encoder Board (StandardLimitAndHomeEncoderPlug)

Servitudes Type: StandardLimitAndHomeEncoderPlug

This setting is used when **both** the Home signal and the travel limit signals are wired to the XPS controller through the encoder board plug.

For all other configuration file parameters refer to section 5.4.4: "Driver Board (StandardEORDriverPlug)".

5.5 Backlash

In this configuration category, users are building the Backlash compensation parameters section of the stages.ini file.

Example:

```
; --- Backlash parameters
; --- <Backlash.Standard>
Backlash = 0 ; Unit
CurrentVelocityCutOffFrequency = 100 ; Hz
CurrentAccelerationCutOffFrequency = 100 ; Hz
```

5.5.1 Type: Standard

Backlash

The stage backlash, *Backlash* (units), sets the backlash compensation value applied to the target position for a move (see **Features Manual**). This value must be greater than or equal to zero (backlash disabled).

The default value is zero.

Current Velocity Cut Off Frequency

Gathering velocity cut off frequency.

Current Acceleration Cut Off Frequency

Gathering acceleration cut off frequency

The gathering cut off frequencies, *CurrentVelocityCutOffFrequency* (Hz) and *CurrentAcceleration-CutOffFrequency* (Hz), set the cut off frequencies for low-pass filters that are applied to the CurrentVelocity and CurrentAcceleration saved by the gathering feature. These filters reduce derivative noise. They must be greater than zero (filter disabled) and less than half of the servo loop frequency.

The default value is 100 Hz which is about five times greater than the bandwidth of the position servo loop of a typical screw driven stage.

5.6 Home Search

In this configuration category, users are building the Home search process parameters section of the stages.ini file.

Example:

```
; --- Home search process parameters
; --- <HomeSearch.MechanicalZeroAndIndexHomeSearch>
HomeSearchSequenceType = MechanicalZeroAndIndexHomeSearch
HomeSearchMaximumVelocity = 100 ; Unit/s
HomeSearchMaximumAcceleration = 500 ; Unit/s²
HomeSearchTimeOut = 5 ; s
HomingSensorOffset = 0 ; Unit
```

The XPS controller supports 8 different home search processes:

- No home search
- Mechanical zero only
- Mechanical zero and index
- Minus end of run only
- Minus end of run and index
- Current position as home
- Plus end of run only
- Index only

The choice depends on the availability of reference signals such as: a mechanical zero signal, a minus end of run signal or an index signal.

For more information about the possible home search processes, please refer to the Features Manual.

5.6.1 No Home Search (NoHomeSearch)

Home Search Sequence Type: NO_HOME_SEARCH

This home search process is used only used for Dummy configuration.

There are no parameters to set for this home search type.

5.6.2 Mechanical Zero and Index (MechanicalZeroAndIndexHomeSearch)

Home Search Sequence Type: MechanicalZeroAndIndexHomeSearch

This home search process is used when the mechanical zero signal and the index signal are both used for the home search.

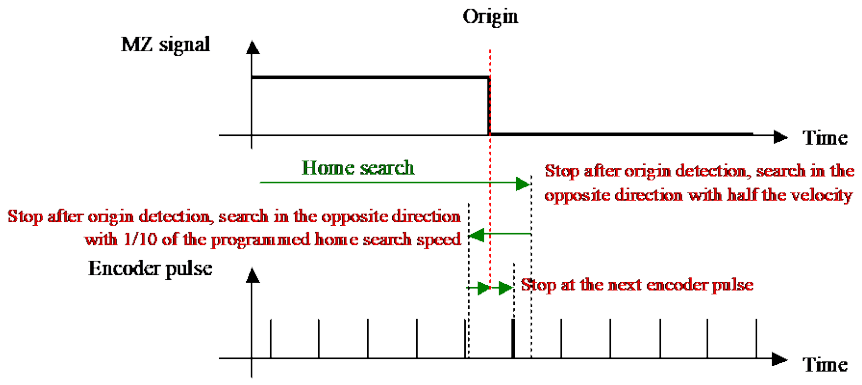


Figure 3: Mechanical zero and index home search process.

Home Search Maximum Velocity

The home search maximum velocity, *HomeSearchMaximumVelocity* (units/s), sets the maximum velocity of the profile generator during the home search process. It must be greater than zero and less than or equal to the *MaximumVelocity* (see section 5.3.1).

The default value is one half of the *MaximumVelocity*.

Home Search Maximum Acceleration

The home search maximum acceleration, *HomeSearchMaximumAcceleration* (units/s²), sets the maximum acceleration of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumAcceleration* (see section 5.3.1).

The default value is one half of the maximum acceleration.

Home Search Time Out

The home search time out, *HomeSearchTimeOut* (s), sets the time out value for the home search process. When the *HomeSearchTimeOut* time is elapsed and the home search process is not yet finished, the XPS controller will generate an emergency stop and will abort the home search process. The *HomeSearchTimeOut* value must be greater than or equal to the profile generator period (400 µs).

Homing Sensor Offset

- In units.
- [0: 1000]

This offset value is used to move the stage *HomingSensorOffset* from the home detected position. Example: With a stage having a HomePreset of zero and a *HomingSensorOffset* of zero will have the stage carriage being at a position. The same stage having a HomePreset of zero and a HomingSensorOffset of ten units will have the stage carriage being ten units away from the previous configuration.

5.6.3 Mechanical Zero Only (MechanicalZeroHomeSearch)

Home Search Sequence Type: MechanicalZeroHomeSearch

This home search process is used only when the mechanical zero signal is used for the home search.

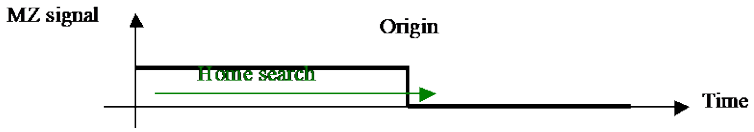


Figure 4: Mechanical zero home search process.

For all other configuration file parameters refer to section 5.6.2:
"Mechanical Zero and Index (MechanicalZeroAndIndexHomeSearch)".

5.6.4 MinusEndOfRunAndIndexHomeSearch

Home Search Sequence Type: MinusEndOfRunAndIndexHomeSearch

This home search process is used when both the minus end of run signal and the index signal are used for the home search.

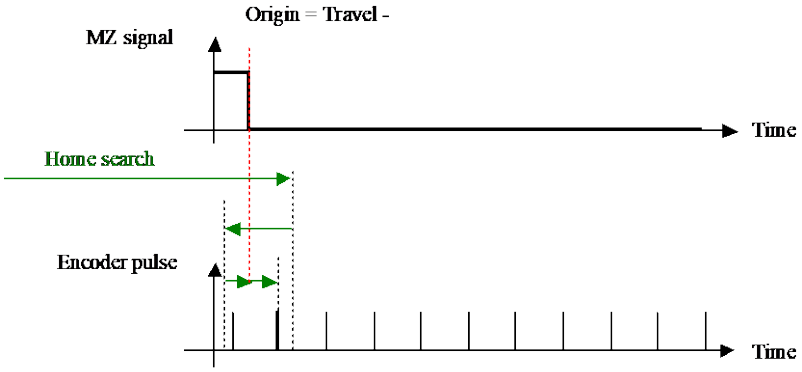


Figure 5: Minus end of run and index home search process.

For all other configuration file parameters refer to section 5.6.2:
"Mechanical Zero and Index (MechanicalZeroAndIndexHomeSearch)".

5.6.5 MinusEndOfRunHomeSearch

Home Search Sequence Type: MinusEndOfRunHomeSearch

This home search process is used only when the minus end of run signal is used for the home search.

Figure 6: Minus end of run home search process.

For all other configuration file parameters refer to section 5.6.2:
"Mechanical Zero and Index (MechanicalZeroAndIndexHomeSearch)".

5.6.6 PlusEndOfRunHomeSearch

Home Search Sequence Type: PlusEndOfRunHomeSearch

This home search process is used only when the plus end of run signal is used for the home search.

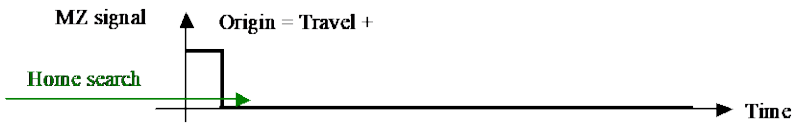


Figure 7: Plus end of run home search process.

For all other configuration file parameters refer to section 5.6.2:
"Mechanical Zero and Index (MechanicalZeroAndIndexHomeSearch)".

5.6.7 IndexHomeSearch

Home Search Sequence Type: IndexHomeSearch

This home search process is used only when the plus end of run signal is used for the home search.

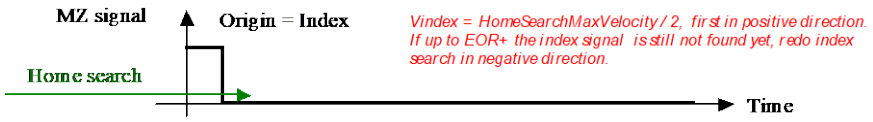


Figure 8: Index home search process.

For all other configuration file parameters refer to section 5.6.2:
"Mechanical Zero and Index (MechanicalZeroAndIndexHomeSearch)".

5.6.8 CurrentPositionAsHome

Home Search Sequence Type: CurrentPositionAsHome

This home search process is used when the current position is defined as the home position.

For all other configuration file parameters refer to section 5.6.2:
"Mechanical Zero and Index (MechanicalZeroAndIndexHomeSearch)".

5.7 Motion done

In this configuration category, users are building the Home search process parameters section of the stages.ini file.

Example:

```
; --- Motion done parameters
; --- <MotionDone.Theoretical>
MotionDoneMode = Theoretical
```

The motion done condition mode defines when a motion is completed. When set to *theoretical motion end,* or a setting *position and velocity checking* can be defined by the user to allow a more precise definition of the motion done by conditioning the motion completion to a number of parameters that take the settling of the positioner into account. The default value for this parameter is *theoretical motion end*.

5.7.1 Theoretical

Motion Done Mode: Theoretical

A theoretical definition of when motion is complete calculated by the motion profiler. Note that this does not take into account the settling of the positioner at the end of the move. See **Features Manual** for more information.

There are no additional parameters to set for this Motion done type.

5.7.2 VelocityAndPositionWindow

Motion Done Mode: VelocityAndPositionWindow

The VelocityAndPositionWindow MotionDone allows a more precise definition of when a motion is complete by specifying the end of the move with a number of parameters that take the settling of the positioner into account. See section **Features Manual** for more information.

Other configuration parameters:

- Motion Done Mode
- Motion Done Position Threshold
- Motion Done Velocity Threshold
- Motion Done Checking Time
- Motion Done Mean Period
- Motion Done Timeout

5.8 Corrector

In this configuration category, users are building the Position servo loop parameters section of the stages.ini file.

Example:

```
; --- Position servo loop parameters
; --- <Corrector.PIDFFAcceleration>
CorrectorType = PIDFFAcceleration
ClosedLoopStatus = Closed
FatalFollowingError = 1 ; Unit
KP = 300000
KI = 10000000
KD = 800
KS = 0.8
GKP = 0
GKD = 0
GKI = 0
KForm = 0 ; Unit
IntegrationTime = 1E+99 ; s
DerivativeFilterCutOffFrequency = 4000 ; Hz
DeadBandThreshold = 0 ; Unit
KFeedForwardAcceleration = 1
NotchFrequency1 = 0 ; Hz
NotchBandwidth1 = 0 ; Hz
NotchGain1 = 0
NotchFrequency2 = 0 ; Hz
NotchBandwidth2 = 0 ; Hz
NotchGain2 = 0
KFeedForwardJerk = 0
```

5.8.1 NoCorrector

Corrector Type: NoCorrector

This servo loop type is only used for Dummy configuration.

There are no additional parameters to set for this servo loop type.

5.8.2 NoEncoderPosition

Corrector Type: NoEncoderPosition

This servo loop type is used for stages without encoder, i.e. stages used always in open loop.

There are no additional parameters to set for this corrector type.

5.8.3 PIDDualFFVoltage

Corrector Type: PIDDualFFVoltage

This servo loop type is used when the position servo loop directly drives the voltage applied to the motor, for instance a DC motor without tachometer connected to a voltage amplifier.

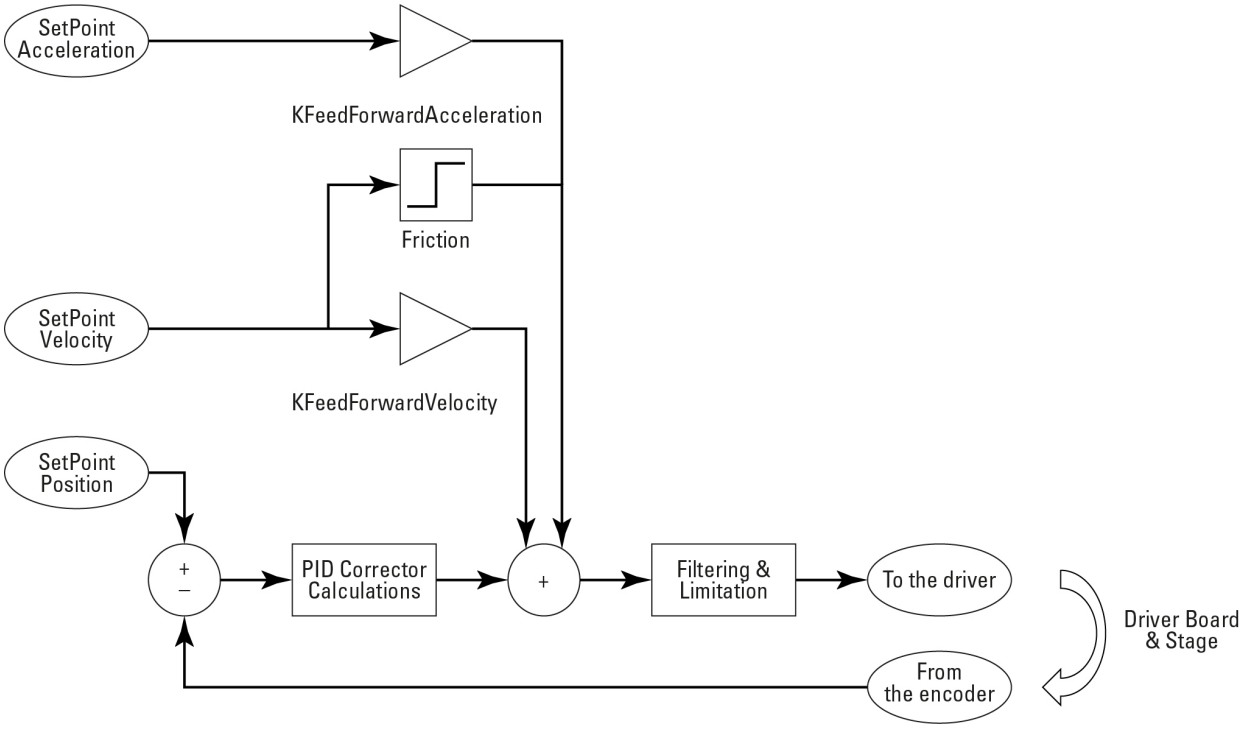


Figure 9: PIDDualFFVoltage corrector.

This servo loop type features a parallel PID servo loop with feed forwards for velocity and acceleration, and two notch filters.

Closed Loop Status

The position servo loop status parameter sets the position servo loop either to open loop i.e. without feedback of a position encoder, or to closed loop, i.e. with feedback from a position encoder. The default value for this parameter is *closed*.

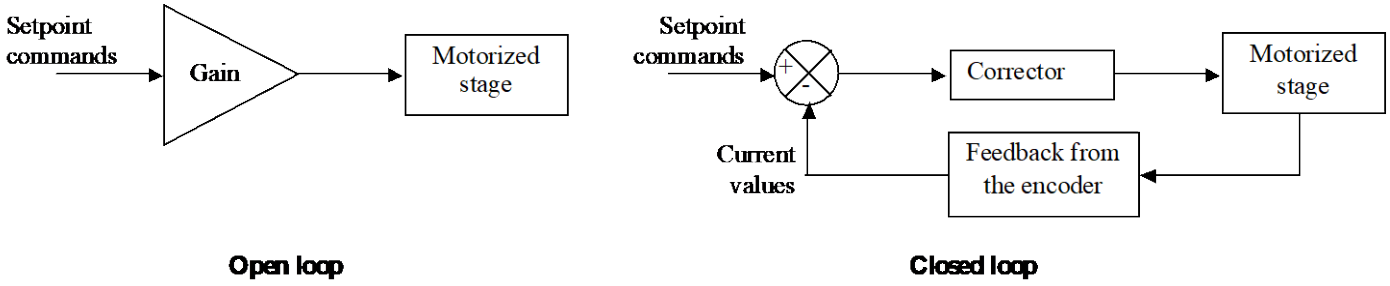


Figure 10: Open and closed loop.

Fatal Following Error

The value for the fatal following error sets the maximum allowed following error of the positioner before generating an error response from the controller. This error is defined as the absolute value of the difference between the setpoint position and the current position. This value is calculated each servo cycle. A following error that exceeds this value will generate the corresponding error code and action. It must be greater than zero.

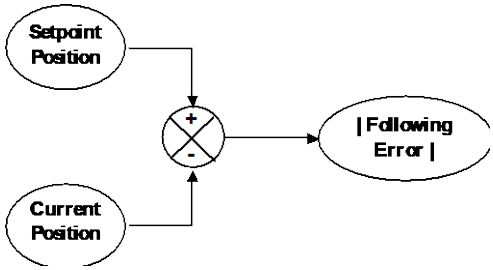


Figure 11: Following error.

Dual Feed Forward (*PIDDualFFVoltage)* PID Parameters entries for the configuration file:

- **K P** — PID servo loop proportional gain
- **K I** — PID servo loop integral gain
- **K D** — PID servo loop derivative gain
- **K S** — PID integral saturation value
- **Integration Time --** PID integration time
- **Derivative Cut Off Frequency --** PID derivative filter cut off frequency
- **K Feed Forward Velocity** -- Velocity feedforward gain
- *K Feed Forward Acceleration* -- Acceleration feedforward gain
- **K Feed Forward Velocity Open Loop --** Velocity feedforward gain in open loop

The PID servo loop parameters *KP*, *KI*, *KD*, *KFeedForwardVelocity* and *FeedForwardAcceleration* define the bandwidth of the servo loop. They must be greater than or equal to zero.

The PID integral saturation value *KS* sets the limit of the integral part of the PID servo loop that is applied to the total servo loop output. The value for *KS* must be between 0 and 1.

The PID *IntegrationTime* (seconds) defines the time span for integration of the residual errors. A small value limits the effect of the integral gain *KI*. It must be greater than or equal to the servo loop period (125 µs).

The PID *DerivativeFilterCutOffFrequency* (Hz) sets the cut-off frequency of the derivative filter. It can be used to reduce the noise introduced by the numerical derivation of the following error. It must be greater than or equal to zero (zero means filter is disabled) and less than or equal to half of the servo loop frequency.

As the output of the position servo loop is neither velocity nor acceleration, individual feed forwards for the velocity and for the acceleration can be set. The *KFeedForwardVelocityOpenLoop* is only used when the position servo loop is in open loop.

All parameters will have an impact on system performance and should be set together. It should be noted that parameters are best set for desired performance according the requirements of the motion application (small following error during trajectory motion, short settling time after a displacement, …). This document will present context specific information on setting PID parameters, but is not intended to be a tutorial on servo loops. Please refer to the common literature for a general treatment of servo loops.

Choice: PID servo loop

$C(p) = KP + \frac{KI}{p} + \frac{KD \times p}{(\tau_d \times p + 1)} \approx KP + \frac{KI}{p} + KD \times p$, where p is the Laplace variable.

Mechanical transfer function: $H(p) = \frac{1}{p \times 60 \times G \times Kv \times (\tau_m \times p + 1)}$

Where: G: ratio between motor rotation and stage displacement (revolution/units)

Kv: motor voltage constant (V/rpm)

$\tau_m = \frac{R_{mot} \times J}{Kt^2}$: stage mechanical time constant (s)

Where: R_{mot}: motor winding resistance per phase (Ω)

J: total inertia on the motor axis (kg.m²). See section 5.10.6: "DRV01AnalogVelocity (with tachometer feedback)" for detail.

Kt: motor torque constant (N.m/A)

Assumptions

- Position closed loop time constants (real and resonance): $\tau_1 = \tau_2$ (s) is between 8 ms and 16 ms (10 Hz and 20 Hz) depending on the first mechanical resonance.
- The damping factor of the closed loop to avoid overshoots (see closed transfer function numerator) is: $\zeta = 1.25$

Notice: $\tau = \frac{1}{2 \times \pi \times f}$

PID parameters

The closed loop transfer function is:

$$T(p) \approx \frac{\frac{KD}{KI} \times p^2 + \frac{KP}{KI} \times p + 1}{\frac{60 \times G \times Kv \times \tau_m}{KI} \times p^3 + \left(\frac{KD}{KI} + \frac{60 \times G \times Kv}{KI}\right) \times p^2 + \frac{KP}{KI} \times p + 1}$$

$$\approx \frac{\left(2 \times \zeta \times \tau_1 \times \tau_2 + \tau_2^2 - \frac{\tau_1 \times \tau_2^2}{\tau_m}\right) \times p^2 + (\tau_1 + 2 \times \zeta \times \tau_2) \times p + 1}{(\tau_1 \times p + 1) \times (\tau_2^2 \times p^2 + 2 \times \zeta \times \tau_2 + 1)}$$

Identification with $(\tau_1 \times p + 1) \times (\tau_2^2 \times p^2 + 2 \times \zeta \times \tau_2 + 1)$ results in:

- PID servo loop proportional gain: $KP = \frac{(\tau_1 + 2 \times \zeta \times \tau_2) \times 60 \times G \times Kv \times \tau_m}{\tau_1 \times \tau_2^2}$
- PID servo loop integral gain: $KI = \frac{60 \times G \times Kv \times \tau_m}{\tau_1 \times \tau_2^2}$
- PID servo loop derivative gain: $KD = \frac{\left(\left(2 \times \zeta \times \tau_1 \times \tau_2 + \tau_2^2\right) \times \tau_m - \tau_1 \times \tau_2^2\right) \times 60 \times G \times Kv}{\tau_1 \times \tau_2^2}$
- PID integral saturation value: default $KS = 0.8$
- PID integration time: $\mathrm{IntegrationTime} = 1e^{99}\,\mathrm{s}$ (full integration)
- PID derivative filter cut off frequency: $\mathrm{DerivativeCutOffFrequency} = 5000\ \mathrm{Hz}$ (maximum)

- velocity feedforward gain: $KFeedForwardVelocity = 60 \times G \times Kv$
- acceleration feedforward gain: $KFeedForwardAcceleration = \frac{R_{mot} \times J}{Kt}$
- velocity feedforward gain in open loop: $KFeedForwardVelocityOpenLoop = 60 \times G \times Kv$

Dual Feed Forward (*PIDDualFFVoltage)* Variable PID parameters entries for the configuration file:

- **G K P** — variable PID proportional gain multiplier
- **G K I** — variable PID integral gain multiplier
- **G K D** — variable PID derivative gain multiplier
- **K Form** — variable PID form coefficient

In addition to the classical gains of the PID servo loop, the XPS controller PID position servo loop features variable gain factors *GKP*, *GKD*, and *GKI*. These gains can be used to reduce settling times of systems exhibiting non-uniform behavior or to tighten the servo loop during the final segment of a move. For example, a stage with a high level of friction will have a response which is dependent on the size of the move: friction is negligible for a large move, but becomes a predominant factor for small moves. For this reason, the required response of the system to reach the commanded position is not the same for small and large moves. The optimum values of PID parameters for small moves are often higher than the optimum values for large moves. Users that do not want to set individual PID gains for different size motions can benefit from variable PID gains. Variable gains are driven by the distance between the target position and the current position. They must be greater than -1.

The parameter PID form coefficient value *KForm* (units) defines the relationship between the distance to the target and the change of the PID gains:

$$Kx = \left(1 + GKx \bullet \frac{KForm}{|TagetPosition - CurrentPosition| + KForm)}\right) \bullet Kx$$

It must be greater than or equal to zero.

The smaller the variable PID form coefficient value is, the sharper the change of the PID gains.

GKp = 10 Kp = 2
Target Position = 0
Encoder Position = -100 to 100

$$Kp_{(Kform,\ Encoder\ Position)} = \left[1 + GK \cdot \left(\frac{Kform}{|\ Target\ Position - Encoder\ Position\ | + Kform}\right)\right] \cdot Kp$$

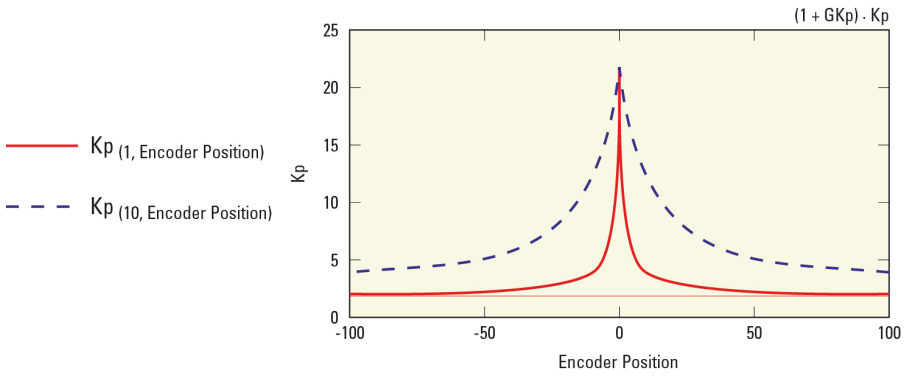


Figure 12: Influence of variable gains.

The default value for these parameters is 0 which disables the variable gains.

Dead Band Threshold

The servo loop dead band threshold sets the dead band value of the position loop. When set to a value other than zero, the position loop is disabled when the following error is less than the value for the dead band threshold AND the theoretical motion is done. In some cases, this can avoid oscillations of stages with backlash or friction. It can also reduce stage settling times, but may result in residual error from the target position. It must be greater than or equal to zero. The default value for this parameter is 0, which disables this feature.

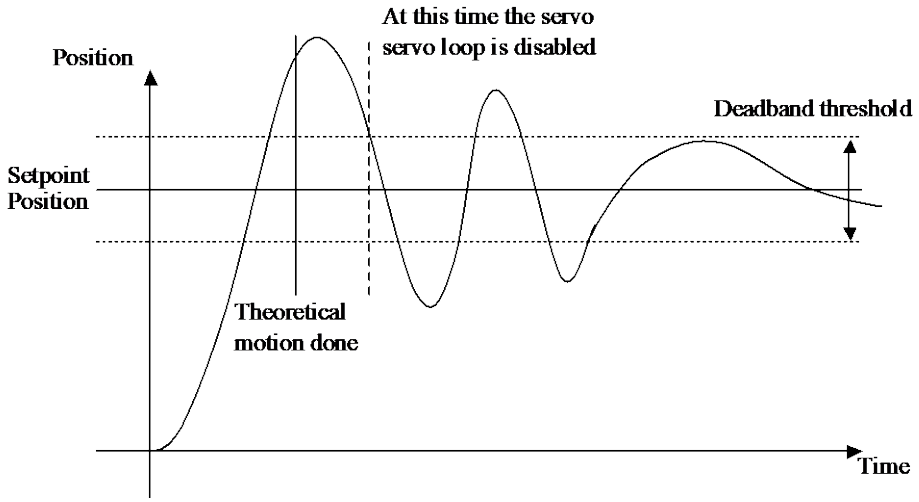


Figure 13: Deadband threshold.

Friction

The friction compensation can be used to compensate for friction of stages during motion. When set to a value other than zero, the friction parameter defines a voltage applied directly to the motor consistent with the direction of motion. The default value for this parameter is 0, which disables this feature.

Dual Feed Forward (*PIDDualFFVoltage)* Notch Filters Parameters entries for the configuration file:

- **Notch Frequency 1** and **Notch Frequency 2** — first and second notch filter center frequency
- **Notch Bandwidth 1** and **Notch Bandwidth 2** — first and second notch filter bandwidth:
- **Notch Gain 1** and **Notch Gain 2** — first and second notch filter gain

The output of the position servo loop is filtered by two notch filters. These filters can be used to avoid the excitation of specific frequencies. They are defined by their center frequencies *NotchFrequency<n°>* (Hz), bandwidths *NotchBandwidth<n°>* (Hz) and their gains *NotchGain<n°>*. The frequencies and bandwidths must be greater than or equal to zero (filter disabled) and less than or equal to half of the servo loop frequency. The gain must be greater than or equal to zero. The default value for these parameters is 0, which disables this feature.

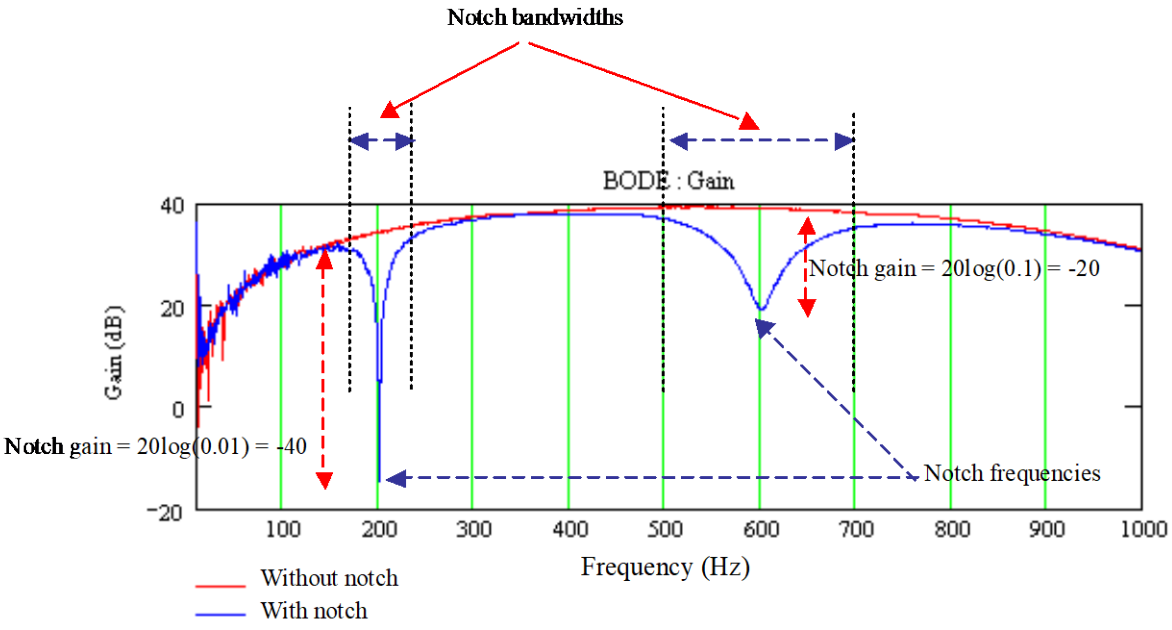


Figure 14: Notch filters.

5.8.4 PIDFFAcceleration

Corrector Type: PIDFFAcceleration

This servo loop type is used when a constant value applied to the driver results in a constant acceleration of the stage.

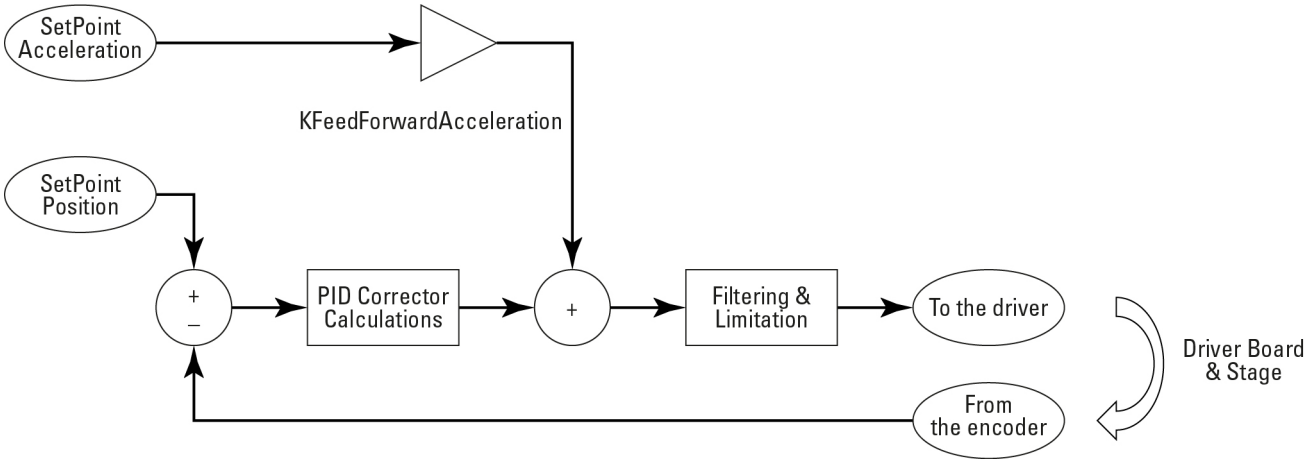


Figure 15: PIDFFAcceleration corrector.

This servo loop type features a parallel PID servo loop with feedforward acceleration and two notch filters.

Closed Loop Status

The position servo loop status parameter sets the position servo loop either to open loop i.e. without feedback of a position encoder, or to closed loop, i.e. with feedback from a position encoder. The default value for this parameter is *closed*.

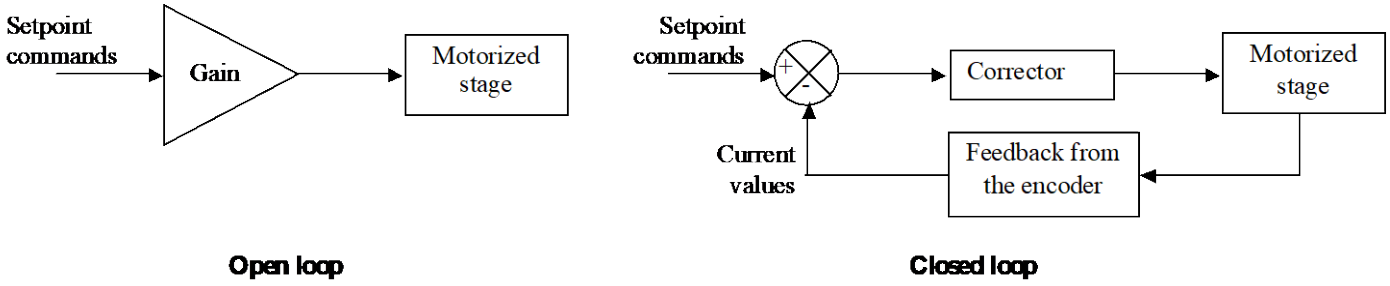


Figure 16: Open and closed loop.

Fatal Following Error

The value for the fatal following error sets the maximum allowed following error of the positioner before generating an error response from the controller. This error is defined as the absolute value of the difference between the setpoint position and the current position. This value is calculated each servo cycle. A following error that exceeds this value will generate the corresponding error code and action. It must be greater than zero.

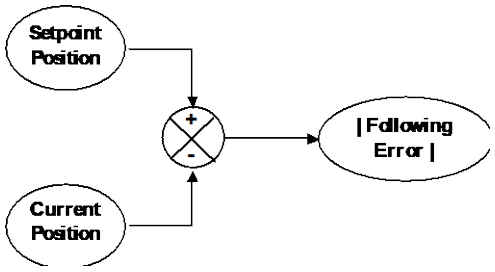


Figure 17: Following error.

Feed Forward (*PIDFFAcceleration*) PID Parameters entries for the configuration file:

- **K P** — PID servo loop proportional gain
- **K I** — PID servo loop integral gain
- **K D** — PID servo loop derivative gain
- **K S** — PID integral saturation value
- **Integration Time** — PID integration time:
- **Derivative Cut Off Frequency** — PID derivative filter cut off frequency
- **K Feed Forward Acceleration** — Acceleration feedforward gain
- **K Feed Forward Jerk** — Jerk feedforward gain

The PID servo loop parameters *KP*, *KI*, *KD* and *KFeedForwardAcceleration, KFeedForwardJerk* define the bandwidth of the servo loop. They must be greater than or equal to zero.

The PID integral saturation value *KS* sets the limit of the integral part of the PID servo loop that is applied to the total servo loop output. The *KS* parameter must be between 0 and 1.

The PID *IntegrationTime* (seconds) defines the time span for integration of the residual errors. A small value limits the effect of the integral gain *KI*. It must be greater than or equal to the servo loop period (125 µs).

The PID *DerivativeFilterCutOffFrequency* (Hz) sets the cut-off frequency of the derivative filter. It can be used to reduce the noise introduced by the numerical derivation of the following error. It must be greater than or equal to zero (zero means filter is disabled) and less than or equal to half of the servo loop frequency.

All parameters will have an impact on system performance and should be set together. It should be noted that parameters are best set for desired performance according the requirements of the motion application (small following error during trajectory motion, short settling time after a displacement, …). This document will present context specific information on setting PID parameters, but is not intended to be a tutorial on servo loops. Please refer to the common literature for a general treatment of servo loops.

Choice: PID servo loop

$C(p) = KP + \frac{KI}{p} + \frac{KD \times p}{(\tau_d \times p + 1)} \approx KP + \frac{KI}{p} + KD \times p$, where p is the Laplace variable.

Mechanical transfer function: $H(p) = \frac{1}{p^2}$ (driver transfer function disregarded)

Assumptions

- Position closed loop time constants (real and resonance), $\tau_1 = \tau_2$ (s) is between 4 ms and 8 ms (20 Hz and 40 Hz) depending on the first mechanical resonance.
- The damping factor of the closed loop to avoid overshoots (see closed transfer function numerator) is: $\zeta = 1.25$

Notice: $\tau = \frac{1}{2 \times \pi \times f}$

PID parameters

The closed loop transfer function is:

$$T(p) \approx \frac{\frac{KD}{KI} \times p^2 + \frac{KP}{KI} \times p + 1}{\frac{1}{KI} \times p^3 + \frac{KD}{KI} \times p^2 + \frac{KP}{KI} \times p + 1}$$

$$\approx \frac{\left(2 \times \zeta \times \tau_1 \times \tau_2 + \tau_2^2\right) \times p^2 + \left(\tau_1 + 2 \times \zeta \times \tau_2\right) \times p + 1}{(\tau_1 \times p + 1) \times \left(\tau_2^2 \times p^2 + 2 \times \zeta \times \tau_2 + 1\right)}$$

Identification with $(\tau_1 \times p + 1) \times \left(\tau_2^2 \times p^2 + 2 \times \zeta \times \tau_2 + 1\right)$ results in:

- PID servo loop proportional gain: $KP = \frac{\tau_1 + 2 \times \zeta \times \tau_2}{\tau_1 \times \tau_2^2} = 1.4e^5$ with $\tau_1 = \tau_2 = 5$ ms
- PID servo loop integral gain: $KI = \frac{1}{\tau_1 \times \tau_2^2} = 8e^6$ with $\tau_1 = \tau_2 = 5ms$
- PID servo loop derivative gain: $KD = \frac{2 \times \zeta \times \tau_1 \times \tau_2 + \tau_2^2}{\tau_1 \times \tau_2^2} = 7e^2$ with $\tau_1 = \tau_2 = 5$ ms
- PID integral saturation value: default $KS = 0.8$
- PID integration time: $\text{IntegrationTime} = 1e^{99}$ s (full integration)
- PID derivative filter cut off frequency: $\text{DerivativeCutOffFrequency} = 5000$ Hz (maximum)
- acceleration feedforward gain: $\text{KFeedForwardAcceleration} = 1$

Feed Forward (*PIDFFAcceleration*) Variable PID parameters entries for the configuration file:

- **G K P** — variable PID proportional gain multiplier
- **G K I** — variable PID integral gain multiplier
- **G K D** — variable PID derivative gain multiplier
- **K Form** — variable PID form coefficient

In addition to the classical gains of the PID servo loop, the XPS controller PID position servo loop features variable gain factors *GKP*, *GKD*, and *GKI*. These gains can be used to reduce settling times of systems exhibiting non-uniform behavior or to tighten the servo loop during the final segment of a move. For example, a stage with a high level of friction will have a response which is dependent on the size of the move: friction is negligible for a large move, but becomes a predominant factor for small moves. For this reason, the required response of the system to reach the commanded position is not the same for small and large moves. The optimum values of PID parameters for small moves are often higher than the optimum values for large moves. Users that do not want to set individual PID gains for different size motions can benefit from variable PID gains. Variable gains are driven by the distance between the target position and the current position. They must be greater than -1.

The parameter PID form coefficient value *KForm* (units) defines the relationship between the distance to the target and the change of the PID gains:

$$K_x = \left(1 + GK_x \bullet \frac{KForm}{|TagetPosition - CurrentPosition| + KForm)}\right) \bullet K_x$$

It must be greater than or equal to zero.

The smaller the variable PID form coefficient value is, the sharper the change of the PID gains.

GKp = 10 Kp = 2
Target Position = 0
Encoder Position = -100 to 100

$$Kp_{(Kform,\ Encoder\ Position)} = \left[1 + GK \cdot \left(\frac{Kform}{|\ Target\ Position - Encoder\ Position\ | + Kform}\right)\right] \cdot Kp$$

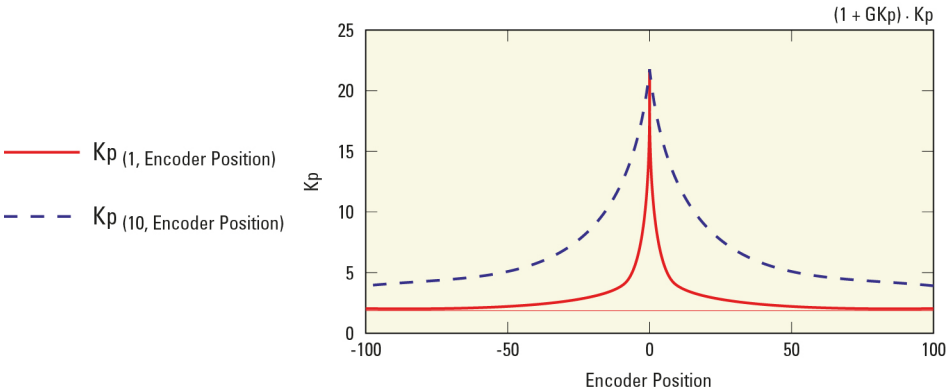


Figure 18: Influence of variable gains.

The default setting for these parameters is 0 which disables the variable gains.

Dead Band Threshold

The servo loop dead band threshold sets the dead band value of the position loop. When set to a value other than zero, the position loop is disabled when the following error is less than the value for the dead band threshold AND the theoretical motion is done. In some cases, this can avoid oscillations of stages with backlash or friction. It can also reduce stage settling times, but may result in residual error from the target position. It

must be greater than or equal to zero. The default value for this parameter is 0, which disables this feature.

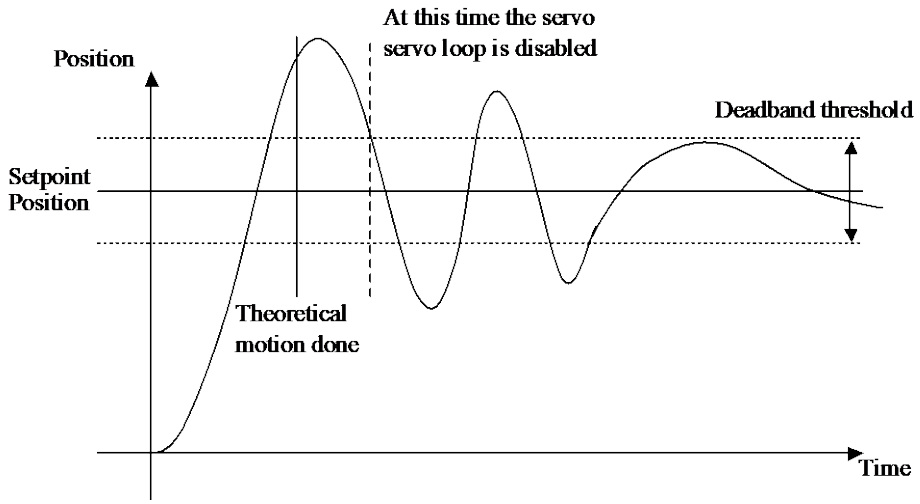


Figure 19: Deadband threshold.

Feed Forward (*PIDFFAcceleration*) Notch filters parameters entries for the configuration file:

- **Notch Frequency 1** and **Notch Frequency 2** — first and second notch filter center frequency
- **Notch Bandwidth 1** and **Notch Bandwidth 2** — first and second notch filter bandwidth:
- **Notch Gain 1** and **Notch Gain 2** — first and second notch filter gain

The output of the position servo loop is filtered by two notch filters. These filters can be used to avoid the excitation of specific frequencies. They are defined by their center frequencies *NotchFrequency<n°>* (Hz), bandwidths *NotchBandwidth<n°>* (Hz) and their gains *NotchGain<n°>*. The frequencies and bandwidths must be greater than or equal to zero (filter disabled) and less than or equal to half of the servo loop frequency. The gain must be greater than or equal to zero. The default value for these parameters is 0, which disables this feature.

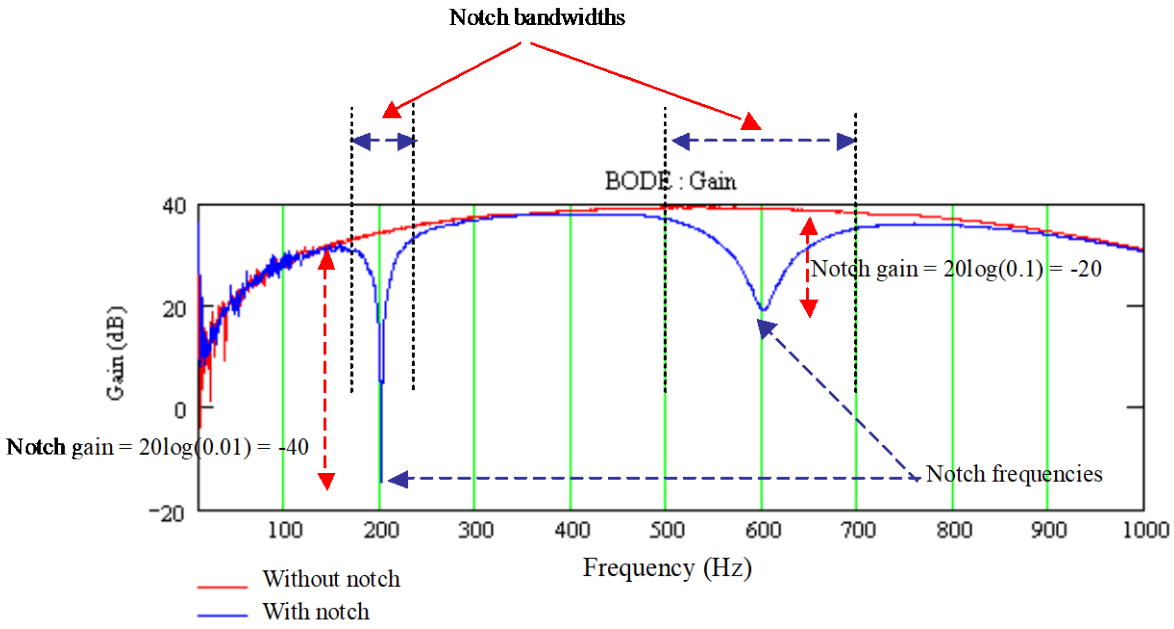


Figure 20: Notch filters.

5.8.5 PIDFFVelocity

Corrector Type: PIDFFVelocity

This servo loop type is used when a constant value applied to the driver results in constant velocity of the stage, for instance a DC motor with tachometer connected to a driver with internal speed loop. This servo loop type features a parallel PID servo loop with a feed forward velocity and two notch filters.

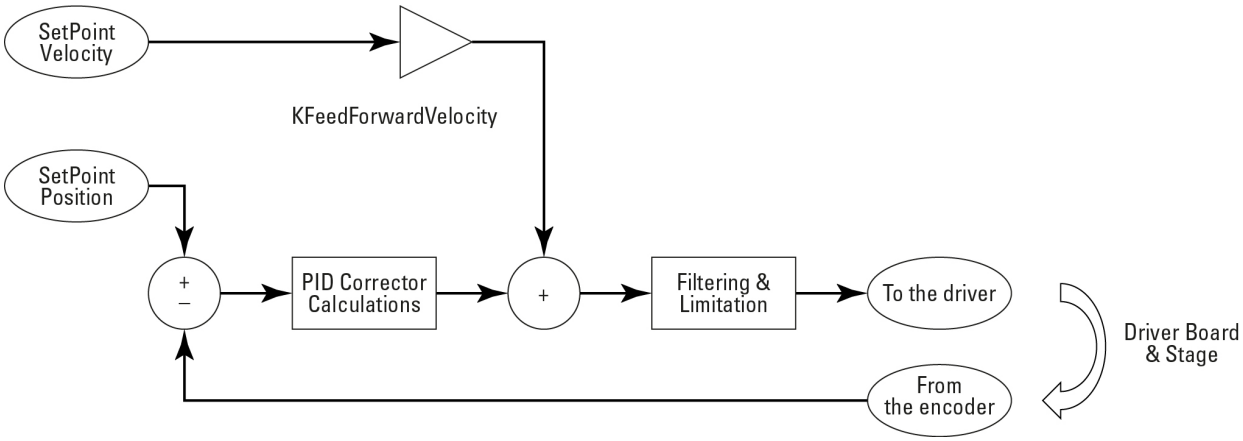


Figure 21: PIDFFVelocity corrector.

Closed Loop Status

The position servo loop status parameter sets the position servo loop either to open loop i.e. without feedback of a position encoder, or to closed loop, i.e. with feedback from a position encoder. The default value for this parameter is *closed*.

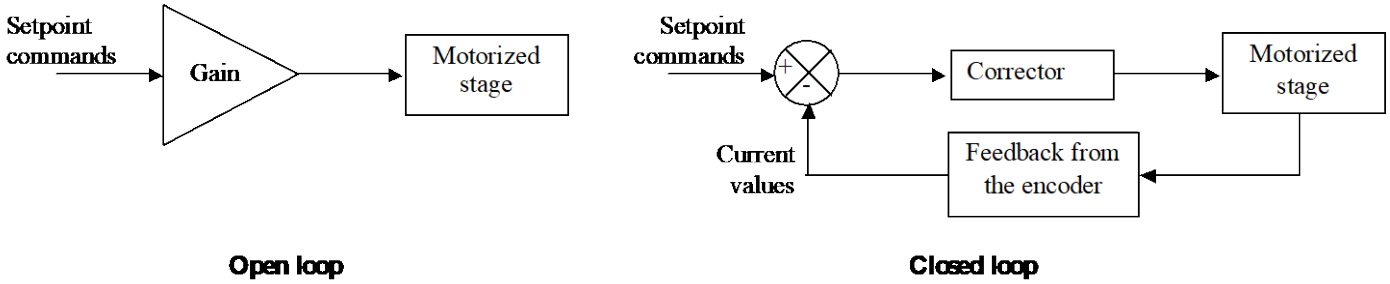


Figure 22: Open and closed loop.

Fatal Following Error

The value for the fatal following error sets the maximum allowed following error of the positioner before generating an error response from the controller. This error is defined as the absolute value of the difference between the setpoint position and the current position. This value is calculated each servo cycle. A following error that exceeds this value will generate the corresponding error code and action. It must be greater than zero.

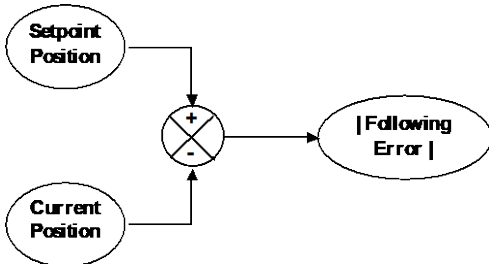


Figure 23: Following error.

Feed Forward (*PIDFFVelocity*) PID Parameters entries for the configuration file:

- **K P** — PID servo loop proportional gain
- **K I** — PID servo loop integral gain
- **K D** — PID servo loop derivative gain
- **K S** — PID integral saturation value
- **Integration Time** — PID integration time:
- **Derivative Cut Off Frequency** — PID derivative filter cut off frequency
- **K Feed Forward Velocity** — Velocity feedforward gain

The PID servo loop parameters *KP*, *KI*, *KD* and *KFeedForwardVelocity* define the bandwidth of the servo loop. They must be greater than or equal to zero.

The PID integral saturation value *KS* sets the limit of the integral part of the PID servo loop that is applied to the total servo loop output. The value for *KS* must be between 0 and 1.

The PID *IntegrationTime* (seconds) defines the time span for integration of the residual errors. A small value limits the effect of the integral gain *KI*. The value in seconds must be greater than or equal to the servo loop period (125 µs).

The PID *DerivativeFilterCutOffFrequency* (Hz) sets the cut-off frequency of the derivative filter. It can be used to reduce the noise introduced by the numerical derivation of the following error. It must be greater than or equal to zero (zero means filter is disabled) and less than or equal to half of the servo loop frequency.

All parameters will have an impact on system performance and should be set together. It should be noted that parameters are best set for desired performance according the requirements of the motion application (small following error during motion, short settling time after a displacement…). This document will present context specific information on setting PID parameters, but is not intended to be a tutorial on servo loops. Please refer to the common literature for a general treatment of servo loops.

Choice: PI servo loop (Kd = 0)

$C(p) = KP + \frac{KI}{p}$, where p is the Laplace variable.

Mechanical transfer function: $H(p) = \frac{1}{p \times (\tau_v \times p + 1)}$, where τ_v is the time constant of the velocity transfer function (s). See section 5.10.6: “DRV01AnalogVelocity (with tachometer feedback)”.

Assumption

The position closed loop resonance time constant τ_p is much greater than τ_v. This allows disregarding the velocity servo loop transfer function. By experience, τ_p is about 10-20 ms for most screw driven stages and τ_p / τ_v ranges between 10 and 20. The damping factor of the closed loop ζ is set to $\zeta = 1.25$ to avoid overshoots, see closed transfer function numerator.

Notice: $\tau = \frac{1}{2 \times \pi \times f}$

PID parameters

By taken into account that $\tau_p >> \tau_v$, the closed loop transfer function is:

$$T(p) \approx \frac{\frac{KP}{KI} \times p + 1}{\frac{1}{KI} \times p^2 + \frac{KP}{KI} \times p + 1} \approx \frac{2 \times \zeta \times \tau_p \times p + 1}{\tau_p^2 \times p^2 + 2 \times \zeta \times \tau_p \times p + 1}$$

This results in:

- Closed loop cut off frequency: $Fc = \frac{\sqrt{1 + 2 \times \zeta^2 + \sqrt{\left(1 + 2 \times \zeta^2\right)^2 + 1}}}{2 \times \pi \times \tau_p}$
- PID servo loop proportional gain: $KP = \frac{2 \times \zeta}{\tau_p} = 156.25$ with $\tau_p = 16ms$
- PID servo loop integral gain: $KI = \frac{1}{\tau_p^2} = 3906.25$ with $\tau_p = 16ms$
- PID servo loop derivative gain: $KD = 0$
- PID integral saturation value: default $KS = 0.8$
- PID integration time: $\mathrm{IntegrationTime} = 1e^{99}\,s$ (full integration)
- PID derivative filter cut off frequency: $\mathrm{DerivativeCutOffFrequency} = 0$ (disabled)
- velocity feedforward gain: $\mathrm{KFeedForwardVelocity} = 1$
- Variable PID parameters

Feed Forward (*PIDFFVelocity*) Variable PID parameters entries for the configuration file:

- **G K P** — variable PID proportional gain multiplier
- **G K I** — variable PID integral gain multiplier
- **G K D** — variable PID derivative gain multiplier
- **K Form** — variable PID form coefficient

In addition to the classical gains of the PID servo loop, the XPS controller PID position servo loop features variable gain factors *GKP*, *GKD*, and *GKI*. These gains can be used to reduce settling times of systems exhibiting non-uniform behavior or to tighten the servo loop during the final segment of a move. For example, a stage with a high level of friction will have a response which is dependent on the size of the move: friction is negligible for a large move, but becomes a predominant factor for small moves. For this reason, the required response of the system to reach the commanded position is not the same for small and large moves. The optimum values of PID parameters for small moves are often higher than the optimum values for large moves. Users that do not want to set individual PID gains for different size motions can benefit from variable PID gains. Variable gains are driven by the distance between the target position and the current position. They must be greater than -1.

The parameter PID form coefficient value *KForm* (units) defines the relationship between the distance to the target and the change of the PID gains:

$$K_x = \left(1 + GK_x \bullet \frac{KForm}{|TagetPosition - CurrentPosition| + KForm)}\right) \bullet K_x$$

It must be greater than or equal to zero.

The smaller the variable PID form coefficient value is, the sharper the change of the PID gains.

GKp = 10 Kp = 2
Target Position = 0
Encoder Position = -100 to 100

$$Kp_{(Kform,\ Encoder\ Position)} = \left[1 + GK \cdot \left(\frac{Kform}{|\ Target\ Position - Encoder\ Position\ | + Kform} \right) \right] \cdot Kp$$

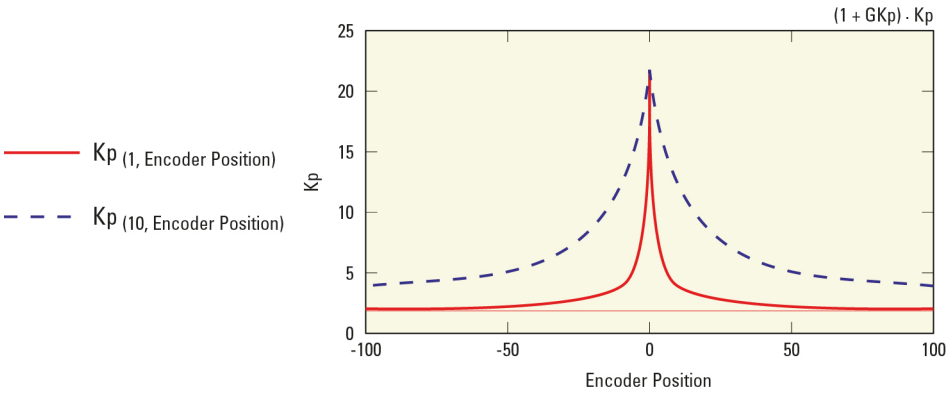


Figure 24: Influence of variable gains.

Dead Band Threshold

The servo loop dead band threshold sets the dead band value of the position control loop. When set to a value other than zero, the position loop is disabled when the following error is less than the value for the dead band threshold AND the theoretical motion is done. In some cases, this can avoid oscillations of stages with backlash or friction. It can also reduce stage settling times, but may result in a residual error from the target position. It must be greater than or equal to zero.

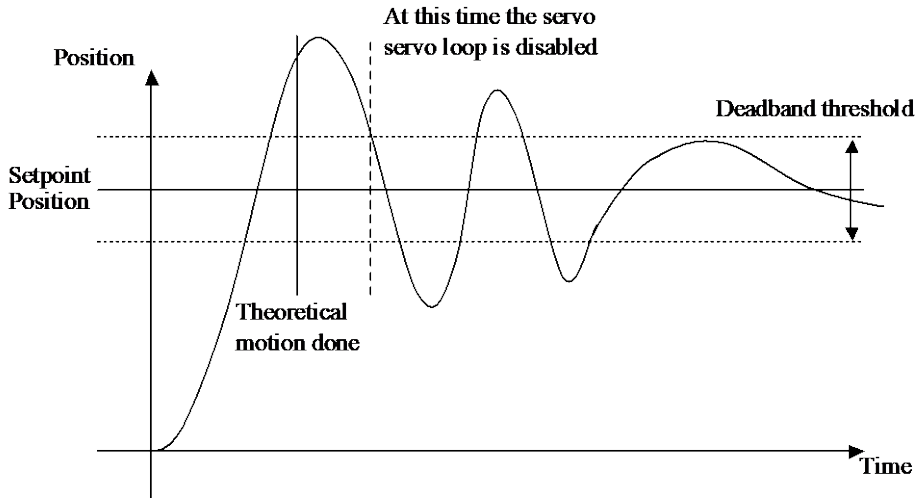


Figure 25: Deadband threshold.

Feed Forward (*PIDFFAcceleration*) Notch filters parameters entries for the configuration file:

- **Notch Frequency 1** and **Notch Frequency 2 —** first and second notch filter center frequency
- **Notch Bandwidth 1** and **Notch Bandwidth 2 —** first and second notch filter bandwidth:
- **Notch Gain 1** and **Notch Gain 2 —** first and second notch filter gain

The output of the position servo loop is filtered by two notch filters. These filters can be used to avoid the excitation of specific frequencies. They are defined by their center frequencies *NotchFrequency<n°>* (Hz), bandwidths *NotchBandwidth<n°>* (Hz) and their gains *NotchGain<n°>*. The frequencies and bandwidths must be greater than or

equal to zero (filter disabled) and less than or equal to half of the servo loop frequency. The gain must be greater than or equal to zero. The default value for these parameters is 0, which disables this feature.

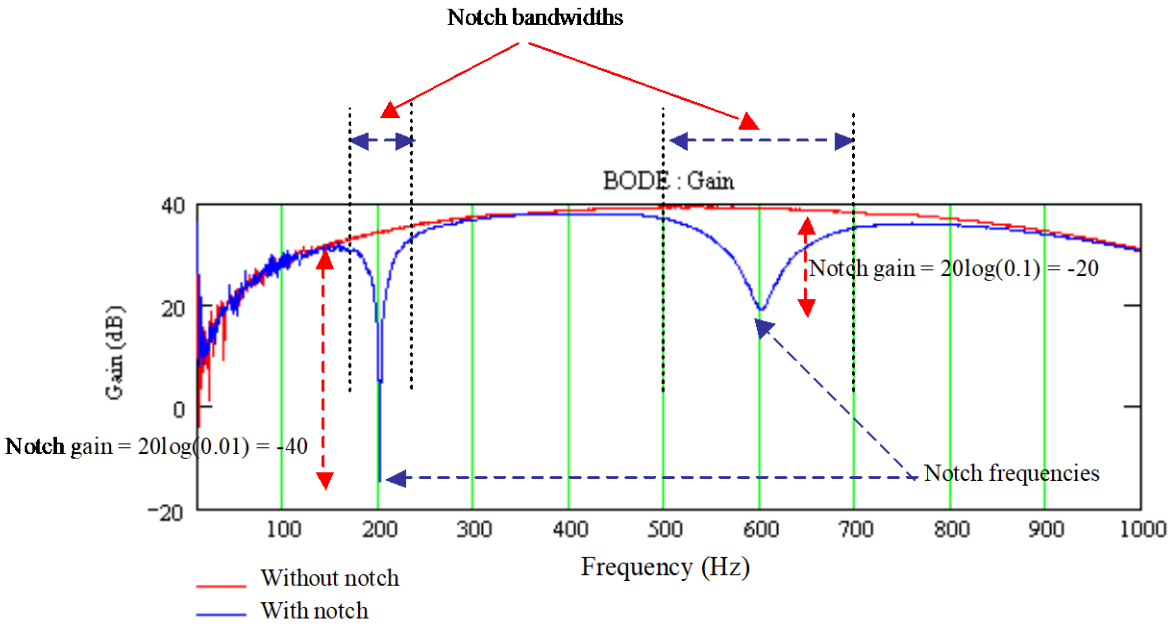


Figure 26: Notch filters.

5.8.6 PIPosition

Corrector Type: PIPosition

This servo loop type is used when the position servo loop directly outputs a position value. This servo loop type features a parallel PI servo loop with two notch filters.

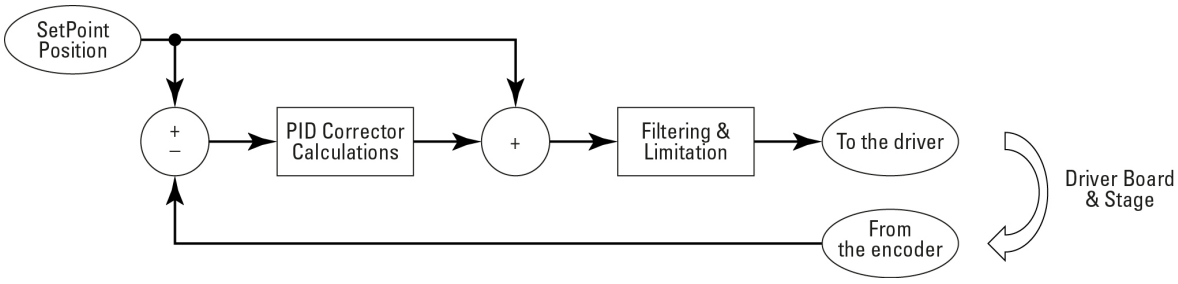


Figure 27: PIPosition corrector.

Closed Loop Status

The position servo loop status parameter sets the position servo loop either to open loop i.e. without feedback of a position encoder, or to closed loop, i.e. with feedback from a position encoder. The default value for this parameter is *closed*.

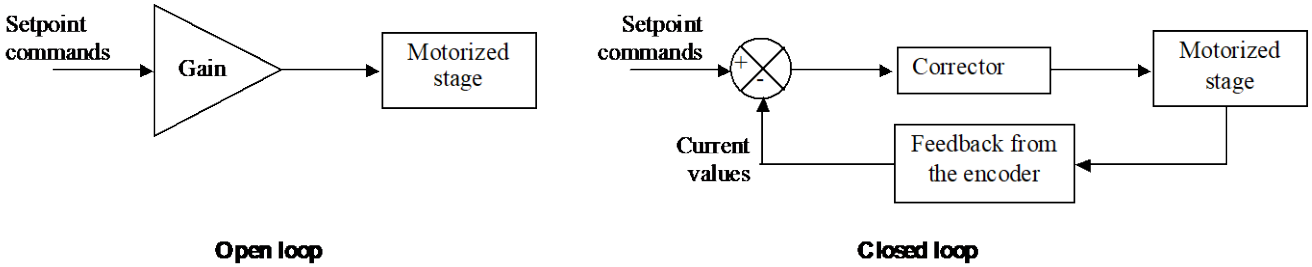


Figure 28: Open and closed loop.

Fatal Following Error

The value for the fatal following error sets the maximum allowed following error of the positioner before generating an error response from the controller. This error is defined as the absolute value of the difference between the setpoint position and the current position. This value is calculated each servo cycle. A following error that exceeds this value will generate the corresponding error code and action. It must be greater than zero.

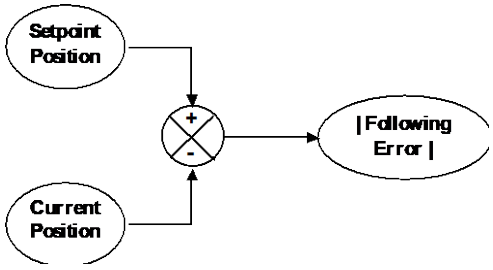


Figure 29: Following error.

Position (*PIPosition*) PI Parameters entries for the configuration file:

- **K P** — PI servo loop proportional gain:
- **K I** — PI servo loop integral gain
- **Integration Time** — PI integration time

The PI servo loop parameters *KP* and *KI* define the bandwidth of the servo loop. They must be greater than or equal to zero.

The PI *IntegrationTime* (seconds) defines the time span for integration of the residual errors. A small value limits the effects of the integral gain *KI*. It must be greater than or equal to the servo loop period (125 µs).

All parameters will have an impact on system performance and should be set together. It should be noted that parameters are best set for desired performance according the requirements of the motion application (small following error during trajectory motion, short settling time after a displacement, …). This document will present context specific information on setting PID parameters, but is not intended to be a tutorial on servo loops. Please refer to the common literature for a general treatment of servo loops.

Choice: I servo loop (no value for *KP*)

$C(p) = \frac{KI}{p}$, where p is the Laplace variable.

Mechanical transfer function: $H(p) = 1$

Assumption

The position closed loop real time constant τ_p (s) is between 40 ms and 80 ms (2 Hz to 4Hz)

Notice: $\tau = \frac{1}{2 \times \pi \times f}$

PI parameters

The closed loop transfer function is:

$$T(p) = \frac{1}{\frac{1}{KI} \times p + 1}$$

This results in:

- PI servo loop proportional gain: $KP = 0$

- PI servo loop integral gain: $KI = \frac{1}{\tau_p} = 12.5$ with $\tau_p = 80ms$
- PID integration time: $\mathrm{IntegrationTime} = 1e^{99}s$ (full integration)

Dead Band Threshold

The servo loop dead band threshold sets the dead band value of the position control loop. When set to a value other than zero, the position loop is disabled when the following error is less than the value for the dead band threshold AND the theoretical motion is done. In some cases, this can avoid oscillations of stages with backlash or friction. It can also reduce stage settling times, but may result in a residual error from the target position. It must be greater than or equal to zero.

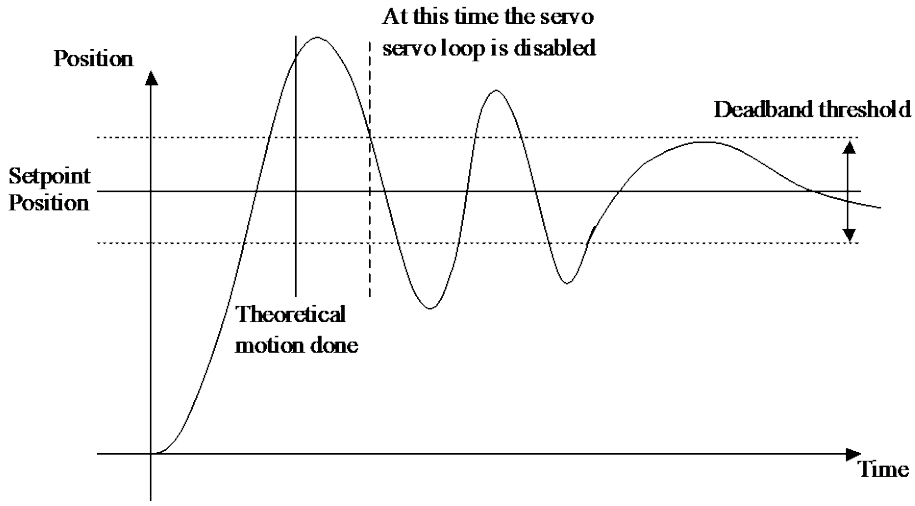


Figure 30: Deadband threshold.

Position (*PIPosition*) Notch Filters Parameters entries for the configuration file:

- **Notch Frequency 1** and **Notch Frequency 2** — first and second notch filter center frequency
- **Notch Bandwidth 1** and **Notch Bandwidth 2** — first and second notch filter bandwidth:
- **Notch Gain 1** and **Notch Gain 2** — first and second notch filter gain

The output of the position servo loop is filtered by two notch filters. These filters can be used to avoid the excitation of specific frequencies. They are defined by their center frequencies *NotchFrequency<n°>* (Hz), bandwidths *NotchBandwidth<n°>* (Hz) and gains *NotchGain<n°>*. The frequencies and bandwidths must be greater than or equal to zero (filter disabled) and less than or equal to half of the servo loop frequency. The gain must be greater than or equal to zero. The default setting for these parameters is 0 which disables the filters.

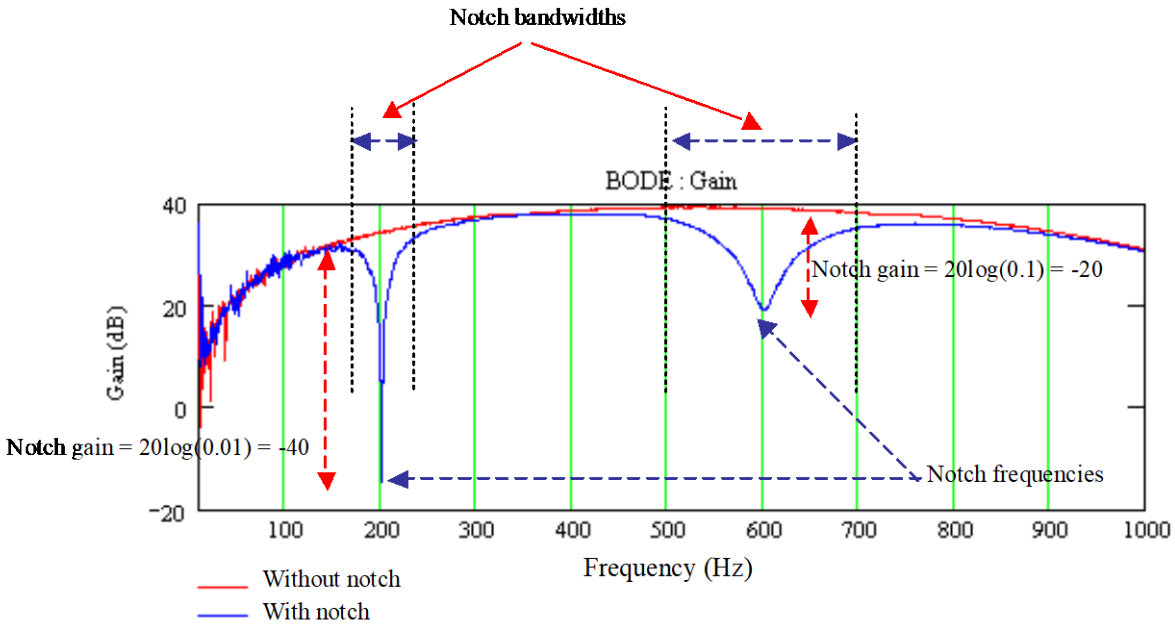


Figure 31: Notch filters.

5.9 Motor Driver Interface

In this configuration category, users are building the Drive command interface parameters section of the stages.ini file.

Example:

```
; --- Driver command interface parameters
; --- <MotorDriverInterface.AnalogSin120AccelerationLMI>
MotorDriverInterface = AnalogSin120AccelerationLMI
DelayAfterMotorOnToSetClosedLoop = 0.05 ; s
ScalingAcceleration = 30590 ; Unit/s²
AccelerationLimit = 13960 ; Unit/s²
MagneticTrackPeriod = 30 ; Unit
MagneticTrackPositionAtHomeMode = Disabled
MagneticTrackPositionAtHome = 0 ; Units
InitializationAccelerationLevel = 7 ; Percent
InitializationCycleDuration = 5 ; s
```

The XPS controller features 7 different motor driver interfaces:

- Velocity control
- Voltage control
- Acceleration control
- Sine/cosine position control
- Position control
- 60/90/120 deg UV phase acceleration control
- 60/90/120 deg UV phase dual output acceleration control

The choice of the motor driver interface depends on the position servo loop type, the driver type and the motor type.

5.9.1 NoMotorInterface

Motor Driver Interface Type: NoMotorInterface

This driver command interface is only used for Dummy configuration.

There are no additional parameters to set for this motor interface type.

5.9.2 AnalogAcceleration

Motor Driver Interface Type: AnalogAcceleration

This driver command interface is used when the output of the position servo loop refers to an acceleration value and when the driver input is an analog acceleration value.

Scaling Acceleration

The stage acceleration at maximum command, *ScalingAcceleration* (units/s²), scales the output of the controller. This value corresponds to the theoretical acceleration reached by the stage with a +10 V input signal to the driver. The value for the *ScalingAcceleration* is stage and driver dependent and must be greater than zero.

See section 5.10.10: “DRV03AnalogAcceleration” for XPS-DRV03 driver board settings.

Delay After Motor On To Set Closed Loop (Not used by standard XPS-Q firmware)

- In seconds.

This delay enables to wait after a motor ON to give time with certain external motor amplifiers to stabilize before the controller closes the position control loop. The default value is 0.050 s.

Acceleration Limit

This parameter should not be confused with the *profile generator maximum acceleration*, which is the maximum acceleration a stage can be commanded to move (see section 5.3.1: “Type: Sgamma”). In order to decrease following errors, a positioner must be allowed to move at greater acceleration than the *profile generator maximum acceleration*. Its maximum value is defined by the maximum allowed stage acceleration, *AccelerationLimit* (units/s²). The higher the dynamic bandwidth of a system, the greater the margin between *maximum allowed stage acceleration* and the *profile generator maximum acceleration*.

The value for the *maximum allowed stage acceleration* must be less than the *stage acceleration at maximum command* and greater than or equal to the *profile generator maximum acceleration.* The recommended value is 1.5 times the value of the *profile generator maximum acceleration*.

Initialization Acceleration Level (percent)

The following relation: is the acceleration used during the stage auto-scaling process.

$$\mathrm{InitializationAcceleration} = \frac{\mathrm{ScalingAcceleration} \bullet \mathrm{InitializationAccelerationLevel}}{100}$$

[InitializationAcceleration] = units/s²

The recommended starting value for this parameter is equal to 20% of the scaling acceleration. If the auto-scaling process does not work properly with this setting (for example, an acceleration that is too low during auto-scaling combined with bad efficiency of the drive chain, could result in failure of the auto-scaling), this value has to be increased step by step. If the displacement during auto-scaling is too large, the *stage initialization acceleration* can be decreased.

5.9.3 AnalogPosition

Motor Driver Interface Type: AnalogPosition

This driver command interface is used when the output of the position servo loop is a position value and when the driver input is an analog position value. This is the case for some drivers of piezo stages or galvanometric mirrors or voice coils.

Delay After Motor On To Set Closed Loop (Not used by standard XPS-Q firmware)

- In seconds.

This delay enables to wait after a motor ON to give time with certain external motor amplifiers to stabilize before the controller closes the position control loop.

Minimum Target Position Voltage

The command voltage at minimum target position, *MinimumTargetPositionVoltage* (V), sets the analog output voltage of the controller when the stage is at its minimum travel limit. This value must be between ±10 V and less than the maximum target position voltage.

Maximum Target Position Voltage

The command voltage at maximum target position, *MaximumTargetPositionVoltage* (V), sets the analog output voltage of the controller when the stage is at its maximum travel limit. This value must be between ±10 V and greater than the minimum target position voltage.

5.9.4 AnalogPositionPiezo

Motor Driver Interface Type: AnalogPositionPiezo

This type of motor driver interface is used specially for piezo stages (DRVP1, …). It is compatible only with *NoEncoderPosition* or *PIPosition* corrector type.

Delay After Motor On To Set Closed Loop (Not used by standard XPS-Q firmware)

- In seconds.

This delay enables to wait after a motor ON to give time with certain external motor amplifiers to stabilize before the controller closes the position control loop.

Default value: 0.050 s.

5.9.5 AnalogSin60Acceleration

Motor Driver Interface Type: AnalogSin60Acceleration

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value and when the driver inputs are analog modulated signals. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels is 60° between phases.

Note: This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

Delay After Motor On To Set Closed Loop (Not used by standard XPS-Q firmware)

- In seconds.

This delay enables to wait after a motor ON to give time with certain external motor amplifiers to stabilize before the controller closes the position control loop.

Default value: 0.050 s.

Scaling Acceleration

The stage acceleration at maximum command, *ScalingAcceleration* (units/s²), scales the analog output of the controller. This value corresponds to the theoretical acceleration reached by the stage with the maximum voltage input to the driver (+10 V for the amplitude of the sine signal). The *ScalingAcceleration* must be greater than zero and is stage and driver dependent. See also section 5.10.8 for parameter definition with the XPS driver board XPS-DRV02.

Acceleration Limit

This parameter should not be confused with the *profile generator maximum acceleration*, which is the maximum acceleration a stage can be commanded to move, see section 5.3.1: “Type: Sgamma”. In order to decrease following errors, a positioner must be allowed to move at greater acceleration than the *profile generator maximum acceleration*. Its maximum value is defined by the maximum allowed stage acceleration, *AccelerationLimit* (units/s²). The higher the dynamic bandwidth of a system, the greater the margin between *maximum allowed stage acceleration* and the *profile generator maximum acceleration*.

The value for the *maximum allowed stage acceleration* should be set in relation to the application and not solely in relation to the motor capability. For correct parameter determination with the driver board XPS-DRV02, see also section 5.10.8: “DRV02”. The value must be less than the *stage acceleration at maximum command* and greater than or equal to the *profile generator maximum acceleration.*

Magnetic Track Period

The stage displacement per motor period, *MagneticTrackPeriod* (units), is the magnetic pitch between two consecutive north poles of the magnetic track. Its value must be greater than zero.

Magnetic Track Position At Home Mode (Not used by standard XPS-Q firmware)

When *MagneticTrackPositionAtHome* is Enabled, the phasing of the motor commutation (previously detected during initialization sequence) is adjusted to the parameter *MagneticTrackPositionAtHome* during the home sequence.

Magnetic Track Position At Home (Not used by standard XPS-Q firmware)

The phasing of the motor commutation (previously detected during initialization sequence) is adjusted to the parameter *MagneticTrackPositionAtHome* during the home sequence, when *MagenticTrackPositionAtHome* is Enabled.

Initialization Acceleration Level (percent)

This parameter is the acceleration used during the motor initialization and the stage auto-scaling processes with the following relation:

$$\mathrm{InitializationAcceleration} = \frac{\mathrm{ScalingAcceleration} \bullet \mathrm{InitializationAccelerationLevel}}{100}$$

[InitializationAcceleration] = units/s²

The recommended starting value for this parameter is 20% of the scaling acceleration and must be greater than or equal to the maximum acceleration of the profile generator, see section 5.4.5: “Encoder Board (StandardLimitAndLimitEncoderPlug)”. If the initialization or auto-scaling process does not work properly with this setting (for example, an acceleration that is too low during auto-scaling combined with bad efficiency of the drive chain, could result in failure of the initialization or auto-scaling),

this value has to be increased step by step. If the displacement during initialization or auto-scaling is too large, the *stage initialization acceleration* can be decreased.

Initialization Frequency

Default value: 50 Hz.

5.9.6 AnalogSin60AccelerationLMI

Motor Driver Interface Type: AnalogSin60AccelerationLMI

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value and when the driver inputs are analog modulated signals. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels is 60° between phases. Large Move Initialization (LMI) will be enabled under this configuration allowing for a larger, user-defined move during the initialization process as compared to standard initialization process.

NOTE

This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

Initialization Cycle Duration

The stage initialization cycle duration, InitializationCycleDuration (s), is the time period of the motor initialization process.

This parameter is used only with the LMI (Large Move Initialization) initialization process.

The recommended starting value of this parameter varies from 1 to 20 seconds and is inversely related to the stage friction. For example, a stage with minimal friction requires a longer stage initialization cycle to stabilize the stage during initialization.

For all other configuration file parameters refer to section 5.9.5: "AnalogSin60Acceleration".

5.9.7 AnalogDualSin60Acceleration

Motor Driver Interface Type: AnalogDualSin60Acceleration

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value balanced on two controller axes and when both driver inputs are analog modulated signals. This is a specific interface for driving two motors via two drivers in parallel. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels is 60° between phases.

NOTE

This interface does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

Delay After Motor On To Set Closed Loop (Not used by standard XPS-Q firmware)

Default value: 0.050 s

Scaling Acceleration

The stage acceleration at maximum command value *ScalingAcceleration* (units/s²) is used to scale the analog output of the controller. This value corresponds to the theoretical acceleration reached by the stage with both motors when the maximum voltage (+10 V for the amplitude of the sine signal) is output by the controller. It must be greater than zero. See also section 5.10.8 for parameter definition with the XPS driver board XPS-DRV02.

Acceleration Limit

The stage acceleration at maximum command, ScalingAcceleration (units/s²), scales the analog output of the controller. This value corresponds to the theoretical acceleration reached by the stage with a maximum voltage input to the driver (+10 V for the amplitude of the sine signal). The ScalingAcceleration must be greater than zero and is stage and driver dependent. See also section 5.10.8 for parameter definition with the XPS driver board XPS-DRV02.

Magnetic Track Period

The stage displacement per motor period, *MagneticTrackPeriod* (units), is the magnetic pitch between two consecutive north poles of the magnetic track. This value must be greater than zero.

Magnetic Track Position At Home Mode (Not used by standard XPS-Q firmware)

When *MagneticTrackPositionAtHome* is Enabled, the phasing of the motor commutation (previously detected during initialization sequence) is adjusted to the parameter *MagneticTrackPositionAtHome* during the home sequence.

Magnetic Track Position At Home (Not used by standard XPS-Q firmware)

The phasing of the motor commutation (previously detected during initialization sequence) is adjusted to the parameter *MagneticTrackPositionAtHome* during the home sequence, when *MagenticTrackPositionAtHome* is Enabled.

Initialization Acceleration Level (percent)

This parameter is the acceleration used during the motor initialization and the stage auto-scaling processes with the following relation:

$$\mathrm{InitializationAcceleration} = \frac{\mathrm{ScalingAcceleration} \bullet \mathrm{InitializationAccelerationLevel}}{100}$$

[InitializationAcceleration] = units/s²

The recommended starting value for this parameter is 20% of the scaling acceleration and must be greater than or equal to the maximum acceleration of the profile generator (see section 5.3.1: “Type: Sgamma”). If the initialization or auto-scaling process does not work properly with this setting (for example, an acceleration that is too low during auto-scaling combined with bad efficiency of the drive chain, could result in failure of the initialization or auto-scaling), this value has to be increased step by step. If the displacement during initialization or auto-scaling is too large, the *stage initialization acceleration* can be decreased.

First Motor Balance and Second Motor Balance

The motor command input balance, *FirstMotorBalance* and *SecondMotorBalance*, scale the outputs of the two drivers to apply the acceleration to the center of gravity. The values for both parameters must be between 0 and 1.

Initialization Frequency

Default value: 50 Hz.

5.9.8 AnalogDualSin60AccelerationLMI

Motor Driver Interface Type: AnalogDualSin60AccelerationLMI

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value balanced on two controller axes and when both driver inputs are analog modulated signals. This is a specific interface for driving two motors via two drivers in parallel. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels is 60° between phases. Large Move Initialization (LMI) will be enabled under this configuration allowing for a larger, user-defined move during the initialization process as compared to standard initialization process.

Note: This interface does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

Initialization Cycle Duration

The stage initialization cycle duration, *InitializationCycleDuration* (s), is the time period of the motor initialization process.

This parameter is used only with the LMI (Large Move Initialization) initialization process.

The recommended starting value of this parameter varies from 1 to 20 seconds and is inversely related to the stage friction. For example, a stage with minimal friction requires a longer stage initialization cycle to stabilize the stage during initialization.

For all other configuration file parameters refer to section 5.9.7: "AnalogDualSin60Acceleration".

5.9.9 AnalogSin90Acceleration

Motor Driver Interface Type: AnalogSin90Acceleration

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value and when the driver inputs are analog modulated signals. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels are 90° between phases for two phase motors.

Note: This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

For all other configuration file parameters refer to section 5.9.5: "AnalogSin60Acceleration".

5.9.10 AnalogSin90AccelerationLMI

Motor Driver Interface Type: AnalogSin90AccelerationLMI

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value and when the driver inputs are analog modulated signals. The modulation is based on the position and suitable for synchronous linear or rotary

brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels are 90° between phases for two phase motors. Large Move Initialization (LMI) will be enabled under this configuration allowing for a larger, user-defined move during the initialization process as compared to standard initialization process.

NOTE

This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

Initialization Cycle Duration

The stage initialization cycle duration, *InitializationCycleDuration* (s), is the time period of the motor initialization process.

This parameter is used only with the LMI (Large Move Initialization) initialization process.

The recommended starting value of this parameter varies from 1 to 20 seconds and is inversely related to the stage friction. For example, a stage with minimal friction requires a longer stage initialization cycle to stabilize the stage during initialization.

For all other configuration file parameters refer to section 5.9.5: "AnalogSin60Acceleration".

5.9.11 AnalogDualSin90Acceleration

Motor Driver Interface Type: AnalogDualSin90Acceleration

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value balanced on two controller axes and when both driver inputs are analog modulated signals. This is a specific interface for driving two motors via two drivers in parallel. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels are 90° between phases for two phase motors.

NOTE

This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

For all other configuration file parameters refer to section 5.9.7: "AnalogDualSin60Acceleration".

5.9.12 AnalogDualSin90AccelerationLMI

Motor Driver Interface Type: AnalogDualSin90AccelerationLMI

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value balanced on two controller axes and when both driver inputs are analog modulated signals. This is a specific interface for driving two motors via two drivers in parallel. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels are 90° between phases for

two phase motors. Large Move Initialization (LMI) will be enabled under this configuration allowing for a larger, user-defined move during the initialization process as compared to standard initialization process.

NOTE

This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

Initialization Cycle Duration

The stage initialization cycle duration, *InitializationCycleDuration* (s), is the time period of the motor initialization process.

This parameter is used only with the LMI (Large Move Initialization) initialization process.

The recommended starting value of this parameter varies from 1 to 20 seconds and is inversely related to the stage friction. For example, a stage with minimal friction requires a longer stage initialization cycle to stabilize the stage during initialization.

For all other configuration file parameters refer to section 5.9.7: "AnalogDualSin60Acceleration".

5.9.13 AnalogSin120Acceleration

Motor Driver Interface Type: AnalogSin120Acceleration

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value and when the driver inputs are analog modulated signals. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels are 120° between phases for three phase motors.

NOTE

This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

For all other configuration file parameters refer to section 5.9.5: "AnalogSin60Acceleration".

5.9.14 AnalogSin120AccelerationLMI

Motor Driver Interface Type: AnalogSin120AccelerationLMI

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value and when the driver inputs are analog modulated signals. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels are 120° between phases for three phase motors. Large Move Initialization (LMI) will be enabled under this configuration allowing for a larger, user defined move during the initialization process as compared to standard initialization process.

NOTE

This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

Initialization Cycle Duration

The stage initialization cycle duration, *InitializationCycleDuration* (s), is the time period of the motor initialization process.

This parameter is used only with the LMI (Large Move Initialization) initialization process.

The recommended starting value of this parameter varies from 1 to 20 seconds and is inversely related to the stage friction. For example, a stage with minimal friction requires a longer stage initialization cycle to stabilize the stage during initialization.

For all other configuration file parameters refer to section 5.9.5: "AnalogSin60Acceleration".

5.9.15 AnalogDualSin120Acceleration

Motor Driver Interface Type: AnalogDualSin120Acceleration

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value balanced on two controller axes and when both driver inputs are analog modulated signals. This is a specific interface for driving two motors via two drivers in parallel. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels are 120° between phases for three phase motors.

NOTE

This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

For all other configuration file parameters refer to section 5.9.7: "AnalogDualSin60Acceleration".

5.9.16 AnalogDualSin120AccelerationLMI

Motor Driver Interface Type: AnalogDualSin120AccelerationLMI

This motor driver interface is used when the output of the position servo loop refers to a modulated acceleration value balanced on two controller axes and when both driver inputs are analog modulated signals. This is a specific interface for driving two motors via two drivers in parallel. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors. Use this Motor interface type for motors whose phase difference between the two output channels are 120° between phases for three phase motors. Large Move Initialization (LMI) will be enabled under this configuration allowing for a larger move during the initialization process as compared to standard initialization process.

NOTE

This interface type does not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

Initialization Cycle Duration:

The stage initialization cycle duration, *InitializationCycleDuration* (s), is the time period of the motor initialization process.

This parameter is used only with the LMI (Large Move Initialization) initialization process.

The recommended starting value of this parameter varies from 1 to 20 seconds and is inversely related to the stage friction. For example, a stage with minimal friction requires a longer stage initialization cycle to stabilize the stage during initialization.

For all other configuration file parameters refer to section 5.9.7: "AnalogDualSin60Acceleration".

5.9.17 AnalogStepperPosition

Motor Driver Interface Type: AnalogStepperPosition

This driver command interface is used when the output of the position servo loop is a position value and when the driver inputs are two channels of analog sine/cosine signals. For example, this would be the case for a stepper motor connected to a driver board XPS-DRV01 and a position loop setting to either *PI with position output* or to *No servo loop with position output*.

Delay After Motor On To Set Closed Loop (Not used by standard XPS-Q firmware)

- In seconds.

This delay enables to wait after a motor ON to give time with certain external motor amplifiers to stabilize before the controller closes the position control loop.

Default value: 0.050 s

Scaling Current

The motor current at maximum command, *ScalingCurrent* (A), scales the output of the controller. Its value corresponds to the current output of the driver with a +10 V input signal. It must be greater than zero.

When used with driver board XPS-DRV01, this value must be set to 3 A.

Displacement Per Full Step

The stage displacement per motor full step, *DisplacementPerFullStep* (units), defines the stage displacement generated by one full step of the motor. Note: One full step displacement corresponds to ¼ of the electrical period.

The *DisplacementPerFullStep* defines the measurement units of the stage and many parameters are derived from this value, essentially all parameters with units of length, such as velocities or accelerations. Therefore, it is critical this parameter be set correctly for proper operation.

To calculate the *DisplacementPerFullStep* value all of the following must be taken into account: the number of steps per revolution, screw pitch and any gear reduction in the stage.

NOTE

The value for the *maximum velocity* (see section 5.3.1: "Type: Sgamma") must be less than or equal to the *DisplacementPerFullStep* multiplied by the servo loop frequency. This is a requirement for smooth operation; however, we recommend not exceeding one quarter of this maximum possible value.

Peak Current Per Phase:

The stepper motor peak current per phase, *PeakCurrentPerPhase* (A), sets the amplitude of the sine/cosine modulated output current of the driver. This corresponds either to the nominal current per phase (1 phase on) or to the nominal current per phases (2 phases on) multiplied by $\sqrt{2}$. This value can be less than the capability of the motor to reduce motor heating. It must be greater than zero and less than or equal to the *motor current at maximum command*.

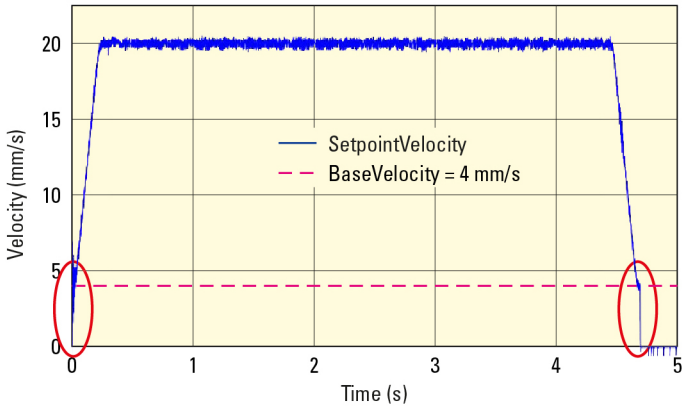
Standby Peak Current Per Phase

The stepper motor standby peak current per phase, *StandByPeakCurrentPerPhase* (A), sets the amplitude of the sine/cosine modulated output current of the driver when the stage has stopped for 5 s. This parameter allows further reduction of motor heating after a motion. It must be greater than zero and less than or equal to the *stepper motor peak current per phase*.

The default value for this parameter is one half of the *stepper motor peak current per phase*.

Base Velocity

The stepper motor start/stop velocity, *BaseVelocity* (units/s), sets the start/stop velocity of the stepper motor. It must be greater than or equal to zero and less than or equal to the maximum velocity, see section 5.3.1: "Type: Sgamma".



In this example, the *BaseVelocity* is set to 4 mm/s: during start/stop periods, the velocity passes through the 4 mm/s velocity step:

- Start period: Starts motion with 4 mm/s velocity and increases from 4 mm/s to the max velocity according to the settings of the motion profiler
- Stop period: Decreases from the max velocity to 4 mm/s velocity according to the settings of the motion profiler and then decreases to null velocity immediately.

The default value for the *BaseVelocity* is zero.

5.9.18 AnalogVelocity

Motor Driver Interface Type: AnalogVelocity

This driver command interface is used when the output of the position servo loop refers to a velocity value and when the driver input is an analog velocity value. For instance, this is the case with a DC motor with tachometer connected to a driver with internal speed loop and a Position servo loop type setting to *PID with a velocity output*.

This driver command interface also provides a configurable current limitation output.

Delay After Motor On To Set Closed Loop (Not used by standard XPS-Q firmware)

- In seconds.

This delay enables to wait after a motor ON to give time with certain external motor amplifiers to stabilize before the controller closes the position control loop.

Default value: 0.050 s.

Scaling Velocity

The stage velocity at maximum command, *ScalingVelocity* (units/s), scales the output of the controller. The value corresponds to the velocity of the positioner with a +10 V input signal to the driver. For the XPS-DRV03 driver board, it is recommended to set this value equal to the maximum allowed stage velocity. For XPS-DRV01 driver board, see section 5.10.6: "DRV01AnalogVelocity (with tachometer feedback)" with the driver board settings.

The value for the *ScalingVelocity* must be greater than zero.

Velocity Limit

This parameter should not be confused with the *profile generator maximum velocity*, which is the maximum velocity a stage can be commanded to move, see section 5.3.1: "Type: Sgamma". In order to decrease following errors, a positioner must be capable of moving faster than the *profile generator maximum velocity*. The maximum value is defined by the maximum allowed stage velocity, *VelocityLimit* (units/s).

The higher the dynamic bandwidth of a system, the greater the margin between the *maximum allowed stage velocity* and the *profile generator maximum velocity*.

The value for the *maximum allowed stage velocity* must be less than the *stage velocity at maximum command* and greater than or equal to the *profile generator maximum velocity.* The recommended value is 1.2 times the value for the *profile generator maximum velocity*.

Scaling Current

The motor current at maximum command, *ScalingCurrent* (A), scales the output of the controller for the current limitation setting.

- When used with driver board XPS-DRV01, this value must be set to 3 A.
- When used with driver board XPS-DRV03, this value must be set to 5 A.
- When used with driver board XPS-DRV00P, this value scales the 10 V analog output of the output channel B, pin 13 on the XPS-DRV00P (see also next entry ***Current Limit***). This value must be greater than zero.

Current Limit

The maximum allowed motor current, *CurrentLimit* (A), defines the current limitation of the motor driver. This values must be less than or equal to the *motor current at maximum command* and greater than zero.

When used with the driver board XPS-DRV00, this value defines the voltage that gets output on the analog output channel B, pin 13, in relation to the *motor current at*

maximum command as follows: Output voltage = 10 V * CurrentLimit (A)/ScalingCurrent (A).

The *CurrentLimit* can be determined as follows:

Motor data:

- motor torque constant: Kt (N.m/A)
- maximum allowed motor current: *MotorCurrentLimit* (A)

Driver data:

- motor current at maximum command: *ScalingCurrent* (A) (for instance 3A for XPS-DRV01 or 5A for XPS-DRV03)

Stage data:

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with: J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

User Performance:

- maximum stage acceleration (see section 5.3.1: “Type: Sgamma”): $MaximumAcceleration$ (units/s²)

Maximum Allowed Motor Current:

- $MaximumCurrent = \frac{MaximumAcceleration \times J \times 2 \times \pi \times G}{Kt}$
- $CurrentLimit = \min(MaximumCurrent \times 1.5, MotorCurrentLimit, ScalingCurrent)$ (A)

NOTE

It is recommended that the CurrentLimit be 1.5 times the motion profiler maximum current to meet the motion requirements of the default stage dynamics.

See also section 5.10.11: “DRV03AnalogVelocity” for XPS-DRV03 driver board with a RMS limitation.

5.9.19 AnalogVoltage

Motor Driver Interface Type: AnalogVoltage

This driver command interface is used when the output of the position servo loop refers to a motor voltage value and when the driver input is a motor voltage value. For instance, this is the case for a DC motor without tachometer connected to a voltage amplifying driver and a Position servo loop type setting to *PID with motor voltage output*.

This driver command interface also provides a configurable current limitation output.

Delay After Motor On To Set Closed Loop (Not used by standard XPS-Q firmware)

- In seconds.

This delay enables to wait after a motor ON to give time with certain external motor amplifiers to stabilize before the controller closes the position control loop.

Default value : 0.05 s.

Scaling Voltage

The motor voltage at maximum command, *ScalingVoltage* (V), scales the analog output of the controller. This value corresponds to the voltage output of the driver with a +10 V input signal. It must be greater than zero.

When used with the driver board XPS-DRV01 or XPS-DRV03, this value must be set to 48 V.

Voltage Limit

The maximum allowed motor voltage, *VoltageLimit* (V), sets the maximum allowed output voltage of the driver. This value must be less than or equal to the *motor voltage at maximum command* and greater than zero.

This parameter can be determined as follows:

Motor data:

- motor winding resistance per phase: R_{mot} (Ω)
- motor torque constant: Kt (N.m/A)
- motor voltage constant: Kv (V/rpm)
- maximum allowed motor current: *MotorCurrentLimit* (A)
- maximum allowed motor voltage: *MotorVoltageLimit* (V)

Driver data:

- motor current at maximum command: *ScalingCurrent* (A)
  (for instance 3A for XPS-DRV01 or 5A for XPS-DRV03)
- motor voltage at maximum command; *ScalingVoltage* (V)
  (for instance 48 V for XPS-DRV01 or XPS-DRV03)

Stage data:

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with: J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

User performance:

- maximum stage velocity (see section 5.3.1: “Type: Sgamma”): $MaximumVelocity$ (units/s)
- maximum stage acceleration (see section 5.3.1: “Type: Sgamma”): $MaximumAcceleration$ (units/s²)

Maximum Allowed Motor Voltage:

- $MaximumCurrent = \min\left(\frac{MaximumAcceleration \times J \times 2 \times \pi \times G}{Kt}, MotorCurrentLimit, ScalingCurrent\right)$
- $MaximumVoltage = R_{mot} \times MaximumCurrent + MaximumVelocity \times 60 \times G \times Kv$

$$VoltageLimit = \min(MaximumVoltage \times 1.5, MotorVoltageLimit, ScalingVoltage)$$

NOTE

It is recommended that the VoltagaeLimit be 1.5 times the motion profiler maximum voltage to meet the motion requirements of the default stage dynamics.

Scaling Current

The motor current at maximum command, *ScalingCurrent* (A), scales the output of the controller for the current limitation setting.

- When used with the driver board XPS-DRV01, this value must be set to 3 A.
- When used with the driver board XPS-DRV03, this value must be set to 5 A.

When used with the driver board XPS-DRV00, this value scales the 10 V analog output of the output channel B, pin 13 on the XPS-DRV00 (see also section 5.9.18: "AnalogVelocity"). Its value must be greater than zero.

Current Limit

The maximum allowed motor current, *CurrentLimit* (A), defines the current limit of the motor driver. This value must be less than or equal to the *motor current at maximum command* and greater than zero.

When used with the driver board XPS-DRV00, this value defines the voltage that gets output on the analog output channel B, pin 13, in relation to the *motor current at maximum command* as follows: Output voltage = 10 V * CurrentLimit (A)/ScalingCurrent (A).

The *CurrentLimit* can be determined as follows:

Motor data:

- motor torque constant: Kt (N.m/A)
- maximum allowed motor current: *MotorCurrentLimit* (A)

Driver data:

- motor current at maximum command: *ScalingCurrent* (A)
  (for instance 3A for XPS-DRV01 or 5A for XPS-DRV03)

Stage data:

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{rmotor} + J_{load} + J_{mec}$

with: J_{rmotor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

User performance:

- maximum stage acceleration (see section 5.3.1: "Type: Sgamma"): $MaximumAcceleration$ (units/s²)

Maximum allowed motor current:

- $MaximumCurrent = \dfrac{MaximumAcceleration \times J \times 2 \times \pi \times G}{Kt}$
- $CurrentLimit = \min(MaximumCurrent \times 1.5, MotorCurrentLimit, ScalingCurrent)$ (A)

NOTE

It is recommended that the CurrentLimit be 1.5 times the motion profiler maximum current to meet the motion requirements of the default stage dynamics.

See also section 5.10.12: "DRV03AnalogVoltage" for XPS-DRV03 driver board with a RMS limitation.

5.9.20 DigitialStepperPosition

Motor Driver Interface Type: DigitalStepperPosition

Required entries in the configuration file: MotorInterfaceType = *PulseDir*

This motor interface drives external stepper motors. The output signals are named PLS_OUT and DIR_OUT. To modify the logic of these signals, two modes are available:

- PLS_OUT = pulses generation only.
- DIR_OUT = direction information only.

DigitalStepperDirectionLogic	**Negative**				**Positive**			
DigitalStepperPulseLogic	**Positive**		**Negative**		**Positive**		**Negative**	
DIR_OUT	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
direction	+	-	+	-	-	+	-	+

PLS_OUT	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
pulse	□	■	■	□	□	■	■	□

■ = pulse □ = no pulse + = positive direction - = negative direction

Delay After Motor On To Set Closed Loop (Not used by standard XPS-Q firmware)

- In seconds.

This delay enables to wait after a motor ON to give time with certain external motor amplifiers to stabilize before the controller closes the position control loop.

Default value: 0.050 s.

Digital Stepper Pulse Logic

Stages.ini file entry: *DigitalStepperPulseLogic*

The logic pulse can be **Negative** (PLS_OUT: 1 = no pulse and 0 = pulse) or **Positive** (PLS_OUT: 1 = pulse and 0 = no pulse).

PLS_OUT	**0**	**1**
DigitalStepperPulseLogic = **Positive**	No pulse	Pulse
DigitalStepperPulseLogic = **Negative**	Pulse	No pulse

Digital Stepper Direction Logic

Stages.ini file entry: *DigitalStepperDirectionLogic*

The logic direction can be **Negative** (DIR_OUT: 1 = negative direction and 0 = positive direction) or **Positive** (DIR_OUT: 1 = positive direction and 0 = negative direction).

The PLS_OUT represents the pulse generation.

DIR_OUT	0	1
DigitalStepperDirectionLogic = **Positive**	Direction -	Direction +
DigitalStepperDirectionLogic = **Negative**	Direction +	Direction -

Displacement Per Full Step

Stages.ini file entry: *DisplacementPerFullStep*

The stage displacement per motor full step, *DisplacementPerFullStep* (units), defines the stage displacement generated by one full step of the motor.

NOTE

One full step displacement corresponds to ¼ of the electrical period.

The *DisplacementPerFullStep* defines the measurement units of the stage and many parameters are derived from this value, essentially all parameters with units of length, such as velocities or accelerations. It is critical this value be set correctly for proper operation of the stage.

To calculate the *DisplacementPerFullStep* value all of the following must be taken into account: the number of steps per revolution, screw pitch and any gear reduction in the stage.

Micro Steps Per Full Step

Stages.ini file entry: *MicroStepsPerFullStep*

The *MicroStepsPerFullStep* defines the number of micro steps in the displacement per one full step of the stepper motor.

5.10 Driver

In this configuration category, users are building the Motor driver parameters section of the stages.ini file.

Example:

```
; --- Motor driver parameters
; --- <Driver.DRV02>
DriverName = XPS-DRV02
DriverMotorResistance = 5.5 ; Ohm
DriverMotorInductance = 0.0018 ; Henry
DriverMaximumPeakCurrent = 2.51 ; Amp
DriverMaximumRMSCurrent = 1.14 ; Amp
DriverRMSIntegrationTime = 15 ; s
DriverThermistanceThreshold = 1000 ; Ohm
DriverCutOffFrequency = 400 ; Hz
```

The XPS controller supports the following settings for the motor driver model:

- No Driver
- Non Configurable Driver
- XPS-DRV00 for non-configurable external drivers
- XPS-DRV00P for configurable external drivers
- XPS-DRV01 with tachometer feedback
- XPS-DRV01 without tachometer feedback
- XPS-DRV01 for stepper motors
- XPS-DRV02 / 02P for linear/brushless motors
- XPS-DRV03 with tachometer feedback
- XPS-DRV03 for acceleration control
- XPS-DRV03 for voltage control
- XPS-DRVPx (x = 1, 2, …) for piezo stage drivers

The choice of a driver board setting depends on the driver board used, the driver command interface, position servo loop type, and motor type.

5.10.1 No Driver (NoDriver)

Motor Driver Type: NoDriver

This type of Driver Name is used when there is not a driver card in the XPS controller and the controller is not connected to a stage.

Driver Name: NO_DRIVER

There are no additional parameters to set for this motor interface type.

5.10.2 Non Configurable Driver

Motor Driver Type: NonConfigurableDriver

This type of Driver Name is used when directly connecting the XPS controller to a non-configurable external motor driver without use of a XPS-DRV00 or XPS-DRV00P pass-through card. This setting of the motor driver type is compatible with all driver command interfaces.

Driver Name: NON_CONFIGURABLE_DRV

There are no additional parameters to set for this motor interface type.

5.10.3 DRV00 (for Non-Configurable External Driver)

Motor Driver Type: DVR00

The XPS-DRV00 is a pass-through board needed for connecting the XPS controller to an external motor driver. This setting of the motor driver type is compatible with all driver command interfaces.

Driver Name: XPS-DRV00

There are no additional parameters to set for this motor interface type.

5.10.4 DRV00P

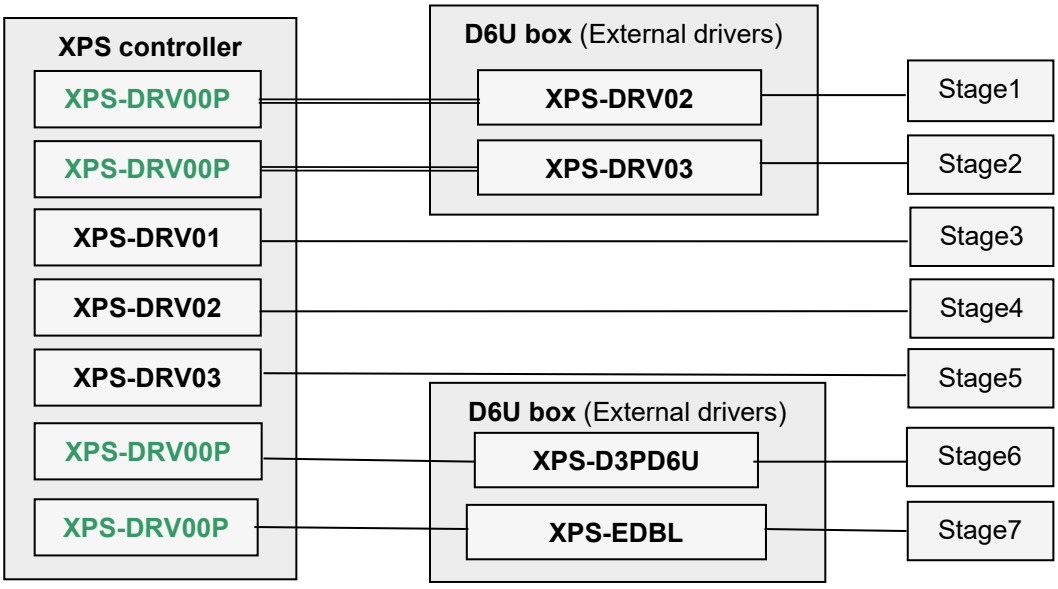
Motor Driver Type: DRV00P

The XPS-DRV00P (DRV00 Version 2) is a pass through board developed for the CIE05/CIE08 board. It is an interconnect board for an external amplifier to connect to the XPS controller. The XPS-DRV00P is similar to an XPS-DRV00 card with the added capability of Pulse/Direction outputs for stepper motor control (Pulse/Dir or Pulse+/Pulse- mode). The I2C communication link allows the user to configure the configurable external drivers (for example external XPS-DRV02, XPS-DRV03, XPS-D3PD6U or XPS-EDBL Newport drivers).

Driver Name: XPS-DRV00P

How to use a configurable external driver?

A configurable external driver must be connected to the XPS controller with an XPS-DRV00P board.



NOTE

In the stages.ini file, the "DriverName" of your stage configuration must be declared as XPS-DRV02, XPS-DRV03, XPS-D3PD6U or XPS-EDBL, but not XPS-DRV00P.

[Stage1]
DriverName = XPS-DRV02

[Stage2]
DriverName = XPS-DRV03

[Stage6]
DriverName = XPS-D3PD6U

[Stage7]
DriverName = XPS-EDBL

There are no additional parameters to set for this motor interface type.

5.10.5 DRV01AnalogStepperPosition (for stepper motors)

Motor Driver Type: DRV01AnalogStepperPosition

This type of motor driver model is used for stepper motors. The motor driver interface must be set to sine/cosine position control (see section 5.9.17: "AnalogStepperPosition") and the position servo loop type to either PI with a position output (see section 5.8.6: "PIPosition") or to no servo loop with a position output (see section 5.8.2: "NoEncoderPosition"). The XPS-DRV01 driver board supplies a maximum output of 3 A and 48 V.

Driver Name: XPS-DRV01

Driver PWM Frequency: *{int, 50, 52, 54, 57, 65, 69, 73, 78, 96, 104, 113, 125, 178, 200, 250, 300} Hz*

The pulse width modulation frequency, *DriverPWMFrequency* (Hz), sets the frequency of the pulse width modulation output of the XPS-DRV01 driver.

The default value is 50 kHz.

Driver Stepper Winding: {string, Full, Half}

The stepper motor winding connection, *DriverStepperWinding*, specifies the stepper wiring. If the stepper is wired between pin 1-4 and pin 5-8, the parameter must be set to *Full.* If the stepper is wired between pin 1-4 and 11-12 as well as pin 5-10, the parameter must be set to *Half.*

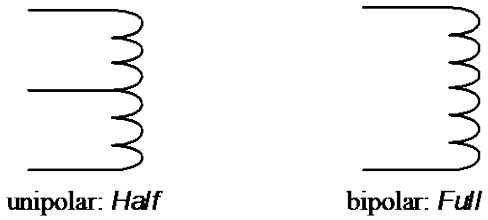


Figure 32: Stepper motor winding connection.

5.10.6 DRV01AnalogVelocity (with tachometer feedback)

Motor Driver Type: DRV01AnalogVelocity

This type of motor driver model is used for DC motors with tachometer. The motor driver interface must be set to *velocity control* (see section 5.9.18: "AnalogVelocity") and the position servo loop type to *PID with velocity output* (see section 5.8.5: "PIDFFVelocity"). The XPS-DRV01 driver board supplies a maximum output of 3 A and 48 V.

Driver Name: XPS-DRV01

Driver PWM Frequency: *{int, 50, 52, 54, 57, 65, 69, 73, 78, 96, 104, 113, 125, 178, 200, 250, 300} Hz*

The pulse width modulation frequency, *DriverPWMFrequency* (Hz), sets the frequency of the pulse width modulation output of the XPS-DRV01 driver.

The default value is 50 kHz.

- **Driver Error Amplifier Gain**: {int, 1, 5, 9, 13, 17, 21, 25, 29}
- **Driver Tachometer Gain**: {int, 0, 25, 27, 30, 33, 50, 60, 75, 100}

The velocity servo loop proportional gain and the tachometer gain should be set together to optimize the bandwidth of the velocity loop.

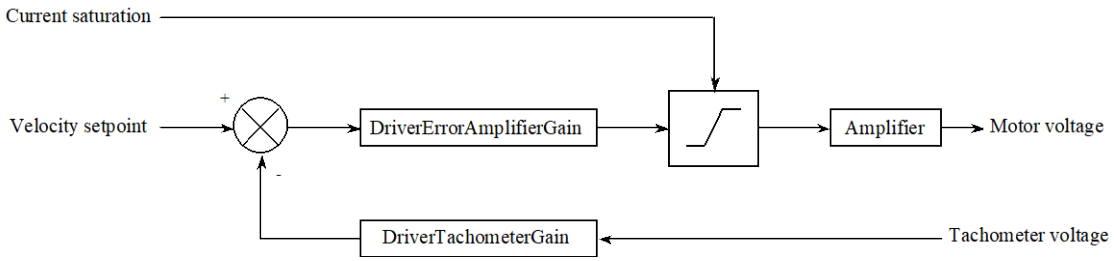


Figure 33: Velocity servo loop.

Motor data

- motor maximum allowed velocity: $V_{motor\,max}$ (rpm)
- motor winding resistance per phase: R_{mot} (Ω)
- motor torque constant: Kt (N.m/A)
- motor voltage constant: Kv (V/rpm)

Notice: $Kt = Kv$ when there are given in these units

Tachometer data

- tachometer voltage constant: K_{GT} (V/rpm)

Driver data

- DC power supply voltage: $U_{DC} = 48V$
- DAC maximum voltage: $U_{DAC\,max} = 10V$
- maximum allowed current at maximum command: $ScalingCurrent = 3A$
- tachometer feedback maximum voltage: $U_{tachometer\,max} = 10V$
- tachometer gain (*DriverTachometerGain*): $K_{tachometer} \in \{100, 75, 60, 50, 33, 30, 27, 25, 0\}$
- velocity servo loop proportional gain (*DriverErrorAmplifierGain*): $Kp_{velocity} \in \{1, 5, 9, 13, 17, 21, 25, 29\}$

Stage data

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with: J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

- stage mechanical time constant: τ_m (s)

Notice: $\tau_m = \frac{R_{mot} \times J}{Kt^2}$

User performance

maximum allowed stage velocity (see section 5.9.18: "AnalogVelocity"): $VelocityLimit$ (units/s)

- maximum stage acceleration (see section 5.3.1: "Type: Sgamma"): $MaximumAcceleration$ (units/s²)
- velocity servo loop cut off frequency: Fc (Hz)

Notice: Usually this cut off frequency is between 100 Hz and 200 Hz

Preliminary tachometer gain

- maximum motor velocity: $V_{motor\,limit} = 60 \times G \times V_{limit} \le V_{motor\,max}$ (rpm)
- raw tachometer gain: $K_{tachometer\,raw} = \frac{U_{tachometer\,max}}{V_{motor\,limit} \times K_{GT}} \times 100$
- tachometer gain: $K_{tachometer}$ closest lower value

Maximum allowed motor current

- $CurrentLimit = \frac{MaximumAcceleration \times 2 \times \pi \times J \times G}{Kt} \times 1.5 \le ScalingCurrent$ (A)

Notice: The coefficient 1.5 is the margin for parameters uncertainty and servo loop transient.

Velocity servo loop proportional gain

- raw velocity servo loop proportional gain:

$$Kp_{velocit\,raw} = \frac{U_{DACmax} \times Kv \times (\tau_m \times 2 \times \pi \times Fc - 1)}{U_{DC} \times K_{GT} \times \frac{K_{tachometer}}{100}}$$

- velocity servo loop proportional gain: $Kp_{velocity}$ closest higher value

Cut off frequency recalculation due to quantification

- raw cut off frequency: $Fc_{raw} = \frac{\frac{Kp_{velocity} \times U_{DC} \times K_{GT}}{U_{DACmax} \times Kv} \times \frac{K_{tachometer}}{100} + 1}{\tau_m \times 2 \times \pi}$ (Hz)
- if the value is too far from the velocity servo loop cut off frequency, the tachometer gain $K_{tachometer}$ can be decreased.

Stage velocity at maximum command (see section 5.9.18: "

AnalogVelocity”):

- $$ScalingVelocity = \frac{\frac{Kp_{velocity} \times U_{DC}}{60 \times G \times Kv}}{\frac{Kp_{velocity} \times U_{DC} \times K_{GT}}{U_{DAC\,max} \times Kv} \times \frac{K_{tachometer}}{100} + 1} \geq VelocityLimit$$ (units/s)

5.10.7 DRV01AnalogVoltage (without tachometer feedback)

Motor Driver Type: DRV01AnalogVoltage

This type of motor driver model is used with DC motors without tachometer. The motor driver interface must be set to *voltage control* (see section 5.9.19: “AnalogVoltage”) and the position corrector type to *PID with voltage output* (see section 5.8.3: “PIDDualFFVoltage”). The XPS-DRV01 driver board supplies a maximum output of 3 A and 48 V.

Driver Name: XPS-DRV01

Driver PWM Frequency: *{int, 50, 52, 54, 57, 65, 69, 73, 78, 96, 104, 113, 125, 178, 200, 250, 300}*

The pulse width modulation frequency, *DriverPWMFrequency* (Hz), sets the frequency of the pulse width modulation output of the XPS-DRV01 driver.

The default value is 50 kHz.

5.10.8 DRV02

Motor Driver Type: DRV02

This type of motor driver model is used for brushless linear or rotary motors. The motor driver interface must be set to *120° UV phase acceleration control* (see section 5.9.13: “AnalogSin120Acceleration”) or *120° UV phase dual output acceleration control* (see section 5.9.15: “AnalogDualSin120Acceleration”) and the position corrector type to *PID with acceleration output* (see section 5.8.4: “PIDFFAcceleration”).

The XPS-DRV02 driver board supplies a maximum output of 5 A and 44 Vpp.

Driver Name: XPS-DRV02

Driver Motor Resistance:

The motor winding resistance per phase, *DriverMotorResistance* (Ω), is the motor resistance of each phase. It must be greater than zero and less than or equal to 65.535 Ω.

Driver Motor Inductance:

The motor winding induction per phase, *DriverMotorInductance* (H), is the motor induction of each phase. It must be greater than zero and less than or equal to 65.535 mH.

Driver Cut Off Frequency:

The current servo loop cut off frequency, *DriverCutOffFrequency* (Hz), sets the bandwidth of the internal driver current servo loop. The internal driver PI parameters are calculated automatically. This value has to be set relative to the bandwidth of the position servo loop (see section 5.8.4: “PIDFFAcceleration”). It must be greater than zero and less than or equal to 3 kHz.

For a stage with a position servo loop cut off frequency between 20 and 50 Hz, the *current servo loop cut off frequency* should be set between 200 and 500 Hz.

Current Monitoring Parameters entries in the configuration file:

- Driver Maximum Peak Current — peak current limit
- Driver Maximum RMS Current — RMS current limit
- Driver RMS Integration Time — RMS integration time

The peak current limit, *DriverMaximumPeakCurrent* (A), is the maximum allowed motor current. If the motor current goes beyond this value, a driver fault is generated. This value must be greater than zero and less than or equal to 5 A.

The RMS current limit, *DriverMaximumRMSCurrent* (A), is the maximum allowed RMS motor current. The RMS integration time *DriverRMSIntegrationTime* (s) defines the integration time span. If the RMS motor current goes beyond this value, a driver fault is generated. The *RMS Current limit* must be greater than zero and less than or equal to 5 A. The *RMS integration time* must be greater than zero and less than or equal to 480 s.

There are many different methods possible to define these values. Here is a description of one method used by Newport for standard products.

Application input:

- Due to the high accuracy requirements of Newport testing, a maximum temperature increase of the motor coils by 20°C is set to avoid thermal effects impacting the motion precision: $\Delta T = 20\ ^{o}C$
- A ratio of 2 between the peak current limit and the RMS current limit is used. This has been found to be a realistic approach in numerous precision motion applications as a good balance between throughput and precision: r = 2.

Motor data

- motor force/torque constant: Kt (N/Arms) or (N.m/Arms)
- motor constant or steepness: S (N²/W) or (N².m²/W)
- thermal resistance: R_{th} (°C/W)
- thermal time constant: τ_{th} (s)

Driver data

- maximum driver current: $\mathrm{I_{max}} = 5\ \mathrm{A}$

Stage data

- load (linear direct drive stage only) : *Load* (kg)
- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{rotor} + J_{load} + J_{rec}$

with: J_{rotor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{rec}: bearing, lead screw, … inertia (kg.m²)

Calculations

- $Kt = \sqrt{3 \times Rf \times S}$, where Rf (Ω) is the motor resistance per phase
- $\tau_{th} = \dfrac{R_{th} \times F_p^2}{\theta\tau \times S}$ or $\tau_{th} = \dfrac{R_{th} \times C_p^2}{\theta\tau \times S}$,

  where: F_p (N) is the motor peak force, C_p (N.m) is the motor peak torque and $\theta\tau$ (°C/s) is the temperature rise at motor peak force
- *DriverMaximumRMSCurrent*: $I_{RMS} = \min\left(\sqrt{\dfrac{\Delta T \times S}{R_{th} \times (1 + 0.004 \times \Delta t)}} \times \dfrac{\sqrt{2}}{Kt}, \mathrm{I_{max}}\right)$
- *DriverMaximumPeakCurrent*: $I_{peak} = \min(I_{RMS} \times r \times 1.1, \mathrm{I_{max}})$

Notice: The coefficient 1.1 is the margin for the servo loop transient.

- DriverRMSIntegrationTime: $\min(\tau_{th}, 15s)$

- ScalingAcceleration: $A_{scaling} = \frac{I_{max}}{\sqrt{2}} \times \frac{Kt}{Load}$ (m/s²) or

  $A_{scaling} = \frac{I_{max}}{\sqrt{2}} \times \frac{Kt}{2 \times \pi \times J \times G}$ (units/s²)

- AccelerationLimit: $A_{limit} = \min\left(\frac{I_{RMS}}{\sqrt{2}} \times r \times \frac{Kt}{Load}, \Lambda_{scaling}\right)$ (m/s²) or

  $A_{limit} = \min\left(\frac{I_{RMS}}{\sqrt{2}} \times r \times \frac{Kt}{2 \times \pi \times J \times G}, \Lambda_{scaling}\right)$ (units/s²)

Driver Thermistance Threshold

The thermistor threshold, *DriverThermistanceThreshold* (Ω), sets the threshold for the driver fault in the event of overheating. This value must be greater than 100 Ω and less than or equal to 9 kΩ.

For a 1 kΩ PTC temperature sensor, the thermistor threshold value is 1000, corresponding to the resistor value of the transition edge. The temperature that corresponds to that threshold is determined by the type of sensor (see color code of the wire).

5.10.9 DRV02P

Motor Driver Type: DVR02P

This type of motor driver model is used for brushless linear or rotary motors. The motor driver interface must be set to *120 deg UV phase acceleration control* (see section 5.9.13: "AnalogSin120Acceleration") or *120 deg UV phase dual output acceleration control* (see section 5.9.15: "AnalogDualSin120Acceleration") and the position corrector type to *PID with acceleration output* (see section 5.8.4: "PIDFFAcceleration").

The XPS-DRV02P driver board supplies a maximum output of 7 A and 44 Vpp.

Driver Name: XPS-DRV02P

For all other configuration file parameters refer to section 5.10.8: "DRV02".

5.10.10 DRV03AnalogAcceleration

Motor Driver Type: DRV03AnalogAcceleration

This type of motor driver model is used for DC motors controlled by acceleration. The moto driver interface must be set to *acceleration control* (see section 5.9.2: "AnalogAcceleration") and the position corrector type to *PID with acceleration output* (see section 5.8.4: "PIDFFAcceleration"). The XPS-DRV03 driver board supplies a maximum output of 5 A and 48 V. The XPS-DRV03H driver board supplies a maximum output of 1.58 A and 48 V.

Driver Name: XPS-DRV03

Driver Motor Resistance

The motor winding resistance, *DriverMotorResistance* (Ω), is the motor resistance. This value must be greater than zero and less than or equal to 65.535 Ω.

Driver Motor Inductance

The motor winding inductance, *DriverMotorInductance* (H), is the motor inductance. This value must be greater than zero and less than or equal to 65.535 mH.

Driver Current Cut Off Frequency

The current servo loop cut off frequency, *DriverCurrentCutOffFrequency* (Hz), sets the bandwidth of the internal driver current servo loop. The internal driver corrector parameters are calculated automatically. This value has to be set in relation to the bandwidth of the position servo loop (see section 5.8.4: "PIDFFAcceleration"). This value must be greater than zero and less than or equal to 3 kHz.

For a stage with a position servo loop cut off frequency between 20 and 50 Hz, the *current servo loop cut off frequency* should be set between 200 and 500 Hz.

Current Monitoring Parameters entries for the configuration file:

- Driver Maximum Peak Current — peak current limit
- Driver Maximum RMS Current — RMS current limit
- Driver RMS Integration Time — RMS integration time

The peak current limit, *DriverMaximumPeakCurrent* (A), is the maximum allowed motor current. If the motor current exceeds this value, a driver fault is generated. This value must be greater than zero and less than or equal to 5 A.

The RMS current limit, *DriverMaximumRMSCurrent* (A), is the maximum allowed RMS motor current. The RMS integration time *DriverRMSIntegrationTime* (s) defines the integration time span. If the RMS motor current exceeds this value, a driver fault is generated. The *RMS Current limit* must be greater than zero and less than or equal to 5 A. The *RMS integration time* must be greater than zero and less than or equal to 60 s.

There are many different methods to define these values. Here is a description of a method used by Newport for standard products.

Application input:

- Because high accuracy needs, a maximum temperature raise of the motor coils by 20°C is set to avoid thermal effects impacting the motion precision:
- A ratio of 2 between the peak current limit and the RMS current limit is used. This has been found to be a good approach in numerous precision motion applications offering a good balance between throughput and precision: = 2

Motor data

- motor torque constant: Kt (N.m/Arms)
- motor constant or steepness: S (N²m²/W)
- thermal resistance: R_{th} (°C/W)
- thermal time constant: τ_{th} (s)

Driver data

- maximum driver current: $I_{max} = 5$ A

Stage data

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with: J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

Calculations

- $Kt = \sqrt{Rf \times S}$, where Rf (Ω) is the motor resistance per phase
- $\tau_{th} = \frac{R_{th} \times C_p^2}{\theta\tau \times S}$,

where: C_p (N.m) is the motor peak torque and
$\theta\tau$ (°C/s) is the temperature rise at motor peak torque

- *DriverMaximumRMSCurrent*: $I_{RMS} = \min\left(\sqrt{\frac{\Delta T \times S}{R_{th} \times (1 + 0.004 \times \Delta t)}} \times \frac{1}{Kt}, I_{max}\right)$
- *DriverMaximumPeakCurrent*: $I_{peak} = \min(I_{RMS} \times r \times 1.1, I_{max})$

Notice: The coefficient 1.1 is the margin for the servo loop transient.

- DriverRMSIntegrationTime: $\min(\tau_{th}, 3s)$
- ScalingAcceleration: $A_{scaling} = I_{max} \times \frac{Kt}{J \times G \times 2\pi}$ (units/s²)
- AccelerationLimit: $A_{limit} = \min\left(I_{RMS} \times r \times \frac{Kt}{J \times G \times 2\pi}, A_{scaling}\right)$ (units/s²)

Driver Maximum Motor Voltage

The maximum allowed motor voltage, *DriverMaximumMotorVoltage* (V), sets the maximum allowed output voltage of the driver.

5.10.11 DRV03AnalogVelocity

Motor Driver Type: DRV03AnalogVelocity

This type of motor driver model is used for DC motors with tachometer. The driver command interface must be set to *velocity control* (see section 5.9.18: "AnalogVelocity") and the position servo loop type to *PID with velocity output* (see section 5.8.5: "PIDFFVelocity"). The XPS-DRV03 driver board supplies a maximum output of 5 A and 48 V. The XPS-DRV03H driver board supplies a maximum output of 1.58 A and 48 V.

Driver Name: XPS-DRV03

Driver Motor Resistance

The motor winding resistance, *DriverMotorResistance* (Ω), is the motor resistance. This value must be greater than zero and less than or equal to 655.35 Ω.

Driver Motor Inductance

The motor winding induction, *DriverMotorInductance* (H), is the motor inductance. This value must be greater than zero and less than or equal to 65.535 mH.

Driver Motor Voltage Constant

The motor voltage constant parameter, *DriverMotorVoltageConstant* (Volt/rpm), sets the back EMF constant of the motor. This value must be greater than zero and less than or equal to $65.535e^{-3}$ V/rpm.

Driver Tacho Generator Voltage

The tachometer generator voltage parameter, *DriverTachoGeneratorVoltage* (Volt/rpm), set the voltage constant of the tachometer generator. This value must be greater than zero and less than or equal to $65.535e^{-3}$ V/rpm.

Driver Stage Inertia

The stage inertia, *DriverStageInertia* (kg.m²), is the total inertia (J) on the motor axis. It must be greater or equal to 10^{-9} kg.m² and less than or equal to 1 kg.m².

Notice: $J = J_{rotor} + J_{load} + J_{rec}$

with: J_{rmotor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

Driver Gear Ratio

The gear ratio, *DriverGearRatio* (revolution/unit), sets the ratio between the motor rotation and the stage displacement. It must be greater than 0.

Driver Current Cut Off Frequency

The current servo loop cut off frequency, *DriverCurrentCutOffFrequency* (Hz), sets the bandwidth of the internal driver current servo loop. The driver internal corrector parameters are calculated automatically. This value has to be set in relation to the bandwidth of the velocity servo loop (see section 5.8.5: “PIDFFVelocity”). This value must be greater than zero and less than or equal to 3 kHz.

For a stage with a velocity servo loop cut off frequency between 100 and 200 Hz, the current servo loop cut off frequency should be set between 500 and 1000 Hz.

Driver Velocity Cut Off Frequency

The velocity servo loop cut off frequency, *DriverVelocityCutOffFrequency* (Hz), sets the bandwidth of the internal driver velocity servo loop. The driver internal corrector parameters are calculated automatically. This value has to be set in relation to the bandwidth of the velocity servo loop (see section 5.8.5: “PIDFFVelocity”). This value must be greater than zero and less than or equal to 300 Hz.

For a stage with a velocity servo loop cut off frequency between 20 and 50 Hz, the velocity servo loop cut off frequency should be set between 100 and 200 Hz.

Current Monitoring Parameters entries for the configuration file:

- Driver Maximum RMS Current — RMS current limit
- Driver RMS Integration Time — RMS integration time

The RMS current limit, *DriverMaximumRMSCurrent* (A), is the maximum allowed RMS motor current. The RMS integration time *DriverRMSIntegrationTime* (s) defines the integration time span. If the RMS motor current goes beyond this value, a driver fault is generated. The RMS Current limit must be greater than zero and less than or equal to 5 A. The RMS integration time must be greater than zero and less than or equal to 60 s.

There are many different methods to define these values. Here is a description of a method used by Newport for standard products.

Application input:

- Due to the high accuracy requirements of Newport testing, a maximum temperature increase of the motor coils by 20°C is set to avoid thermal effects impacting the motion precision: $\Delta T = 20°C$.
- A ratio of 2 between the current limit (defined in the motor interface section) and the RMS current limit is used. This has been found to be a good approach in numerous precision motion applications offering a good balance between throughput and precision: $r = 2$.

Motor data

- motor torque constant: Kt (N.m/Arms)
- motor constant or steepness: S (N².m²/W)
- thermal resistance: R_{th} (°C/W)
- thermal time constant: τ_{th} (s)

Driver data

- maximum driver current: $I_{max} = 5\ A$

Calculations

- $Kt = \sqrt{Rf \times S}$, where Rf (Ω) is the motor resistance per phase
- $\tau_{th} = \frac{R_{th} \times C_p^2}{\theta\tau \times S}$,

  where: C_p (N.m) is the motor peak torque and

  $\theta\tau$ (°C/s) is the temperature rise at motor peak torque
- *DriverMaximumRMSCurrent*: $I_{RMS} = \min\left(\sqrt{\frac{\Delta T \times S}{R_{th} \times (1 + 0.004 \times \Delta t)}} \times \frac{1}{Kt}, I_{max}\right)$
- $CurrentLimit = \min(I_{RMS} \times r, I_{max})$
- DriverRMSIntegrationTime: $\min(\tau_{th}, 3s)$

Driver Maximum Motor Voltage

The maximum allowed motor voltage, *DriverMaximumMotorVoltage* (V), sets the maximum allowed output voltage of the driver. It must be greater than zero and less than or equal to 48 V.

This parameter can be determined as follows:

Motor data

- motor winding resistance per phase: R_{mot} (Ω)
- motor torque constant: Kt (N.m/A)
- motor voltage constant: Kv (V/rpm)
- maximum allowed motor current: *MotorCurrentLimit* (A)
- maximum allowed motor voltage: *MotorVoltageLimit* (V)

Driver data

- motor current at maximum command: ScalingCurrent (A) (5A for XPS-DRV03)
- motor voltage at maximum command; ScalingVoltage (V) (48 V for XPS-DRV03)

Stage data

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with: J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

User performance

- maximum stage velocity (see section 5.3.1: “Type: Sgamma”): $MaximumVelocity$ (units/s)
- maximum stage acceleration (see section 5.3.1: “Type: Sgamma”): $MaximumAcceleration$ (units/s²)

Maximum allowed motor voltage

- $MaximumCurrent = \min\left(\frac{MaximumAcceleration \times J \times 2 \times \pi \times G}{Kt}, MotorCurrentLimit, ScalingCurrent\right)$
- $MaximumVoltage = R_{mot} \times MaximumCurrent + MaximumVelocity \times 60 \times G \times Kv$
- $VoltageLimit = \min(MaximumVoltage \times 1.5, MotorVoltageLimit, ScalingVoltage)$

NOTE

It is recommended that the VoltageLimit be 1.5 times the motion profiler maximum voltage to meet the motion requirements of the default stage dynamics.

5.10.12 DRV03AnalogVoltage

Motor Driver Type: DRV03AnalogVoltage

This setting of the motor driver model is used for DC motors controlled directly by the motor voltage. The motor driver interface must be set to *voltage control* (see section 5.9.19: "AnalogVoltage") and the position corrector type to *PID with voltage output* (see section 5.8.3: "PIDDualFFVoltage"). The XPS-DRV03 driver board supplies a maximum output of 5 A and 48 V.

Driver Name: XPS-DRV03

Current Monitoring Parameters entries for the configuration file:

- Driver Maximum RMS Current — RMS current limit
- Driver RMS Integration Time — RMS integration time

The RMS current limit, *DriverMaximumRMSCurrent* (A), is the maximum allowed RMS motor current. The RMS integration time *DriverRMSIntegrationTime* (s) defines the integration time span. If the RMS motor current goes beyond this value, a driver fault is generated. The *RMS Current limit* must be greater than zero and less than or equal to 5 A. The *RMS integration time* must be greater than zero and less than or equal to 60 s.

There are many different methods to define these values. Here is a description of a method used by Newport for standard products.

Application input:

- Because high accuracy needs, a maximum temperature raise of the motor coils by 20°C is set to avoid thermal effects impacting the motion precision: $\Delta T = 20°C$.
- A ratio of 2 between the current limit (defined in the motor interface section) and the RMS current limit is used. This has been found to be a good approach in numerous precision motion applications offering a good balance between throughput and precision: r = 2.

Motor data

- motor torque constant: Kt (N.m/Arms)
- motor constant or steepness: S (N².m²/W)
- thermal resistance: R_{th} (°C/W)
- thermal time constant: τ_{th} (s)

Driver data

- maximum driver current: $I_{max} = 5$ A

Calculations

- $Kt = \sqrt{Rf \times S}$,

  where Rf (Ω) is the motor resistance per phase

- $\tau_{th} = \frac{R_{th} \times C_p^2}{\theta\tau \times S}$,

  where: C_p (N.m) is the motor peak torque and

  $\theta\tau$(°C/s) is the temperature rise at motor peak torque

- *DriverMaximumRMSCurrent*: $I_{RMS} = \min\left(\sqrt{\frac{\Delta T \times S}{R_{th} \times (1 + 0.004 \times \Delta t)}} \times \frac{1}{Kt}, I_{max}\right)$
- $CurrentLimit = \min(I_{RMS} \times r, I_{max})$
- DriverRMSIntegrationTime: $\min(\tau_{th}, 3s)$

5.10.13 DRV03HAnalogAcceleration

Motor Driver Type: DRV03HAnalogAcceleration

This type of motor driver model is used for DC motors controlled by acceleration. The motor driver interface must be set to *acceleration control* (see section 5.9.2: "AnalogAcceleration") and the position corrector type to *PID with acceleration output* (see section 5.8.4: "PIDFFAcceleration"). The XPS-DRV03H driver board supplies a maximum output of 1.58 A and 48 V.

Driver Name: XPS-DRV03H

For all other configuration parameters, refer to section 5.10.10: "DRV03AnalogAcceleration".

5.10.14 DRV03HAnalogVelocity

Motor Driver Type: DRV03HAnalogVelocity

This type of motor driver model is used for DC motors with tachometer. The driver command interface must be set to *velocity control* (see section 5.9.18: "AnalogVelocity") and the position servo loop type to *PID with velocity output* (see section 5.8.5: "PIDFFVelocity").The XPS-DRV03H driver board supplies a maximum output of 1.58 A and 48 V.

Driver Name: XPS-DRV03H

For all other configuration parameters, refer to section 5.10.11: "DRV03AnalogVelocity".

5.10.15 DRV03HAnalogVoltage

Motor Driver Type: DRV03HAnalogVoltage

This setting of the motor driver model is used for DC motors controlled directly by the motor voltage. The motor driver interface must be set to *voltage control* (see section 5.9.19: "AnalogVoltage") and the position corrector type to *PID with voltage output* (see section 5.8.3: "PIDDualFFVoltage"). The XPS-DRV03H driver board supplies a maximum output of 1.58 A and 48 V.

Driver Name: XPS-DRV03H

For other configuration file parameters, refer to section 5.10.12: "DRV03AnalogVoltage".

5.10.16 DRV11AnalogVoltage

Motor Driver Type: DRV11AnalogVoltage

This setting of the motor driver model is used for DC motors controlled directly by the motor voltage. The motor driver interface must be set to *voltage control* (see section 5.9.19: "AnalogVoltage") and the position corrector type to *PID with voltage output* (see section 5.8.3: "PIDDualFFVoltage"). The XPS-DRV11 driver board supplies a maximum output of 6.4 A and 48 V.

Driver Name: XPS-DRV11

For other configuration file parameters, refer to section 5.10.12: "DRV03AnalogVoltage".

5.10.17 DRV11AnalogStepperPosition (for stepper motors)

Motor Driver Type: DRV11AnalogStepperPosition

This type of motor driver model is used for stepper motors. The motor driver interface must be set to sine/cosine position control (see section 5.9.17: "AnalogStepperPosition") and the position servo loop type to either PI with a position output (see section 5.8.6: "PIPosition") or to no servo loop with a position output (see section 5.8.2: "NoEncoderPosition"). The XPS-DRV11 driver board supplies a maximum output of 6.4 A and 48 V.

Driver Name: XPS-DRV11

Driver Motor Resistance

The motor winding resistance, *DriverMotorResistance* (Ω), is the motor resistance. This value must be greater than zero and less than or equal to 655.35 Ω.

Driver Motor Inductance

The motor winding induction, *DriverMotorInductance* (H), is the motor inductance. This value must be greater than zero and less than or equal to 65.535 mH.

Driver Current Cut Off Frequency

The current servo loop cut off frequency, *DriverCurrentCutOffFrequency* (Hz), sets the bandwidth of the internal driver current servo loop. The driver internal corrector parameters are calculated automatically. This value has to be set in relation to the bandwidth of the velocity servo loop (see section 5.8.5: "PIDFFVelocity"). This value must be greater than zero and less than or equal to 3 kHz.

For a stage with a velocity servo loop cut off frequency between 100 and 200 Hz, the current servo loop cut off frequency should be set between 500 and 1000 Hz.

Driver Bemf Compensation

The motor parameters are used to configure the current loop bandwith. The DriverBemfCompensationGain parameter is used to have a constant torque versus velocity.

5.10.18 DRV11AnalogAcceleration

Motor Driver Type: DRV11

This type of motor driver model is used for brushless linear or rotary motors. The motor driver interface must be set to *120 deg UV phase acceleration control* (see section 5.9.13: "AnalogSin120Acceleration") or *120 deg UV phase dual output acceleration control* (see section 5.9.15: "AnalogDualSin120Acceleration") and the position

corrector type to *PID with acceleration output* (see section 5.8.4: “PIDFFAcceleration”).

The XPS-DRV11 driver board supplies a maximum output of 6.4 A and 44 Vpp.

Driver Name: XPS-DRV11

For all other configuration file parameters refer to section 5.10.8: “DRV02”.

5.10.19 DRVP1AnalogPositionPiezo

Motor Driver Type: DRVP1AnalogPositionPiezo

This type of DriverName is used with a piezo driver card. This setting of the motor driver model is compatible only with:

- AnalogPositionPiezo driver command interfaces,
- NoEncoderPosition or PIPosition corrector type.

Driver Name: XPS-DRVP1

Driver parameters

- DriverNotchFrequency
- DriverNotchBandwidth
- DriverNotchGain
- DriverLowpassFrequency
- DriverKI
- DriverFatalFollowingError
- DriverStagePositionOffset
- DriverTravelCorrection

For more information, refer to the XPS-DRVP1 User’s Manual.

5.10.20 EDBL

Motor Driver Type: EDBL

This type of motor driver model is used for brushless linear or rotary motors. The motor driver interface must be set to *120° UV phase acceleration control* (see section 5.9.13: “AnalogSin120Acceleration”) or *120° UV phase dual output acceleration control* (see section 5.9.15: “AnalogDualSin120Acceleration”) and the position corrector type to *PID with acceleration output* (see section 5.8.4: “PIDFFAcceleration”). The XPS-EDBL, external driver module, supplies a maximum output of 25 A and 96 Vpp.

Driver Name: XPS-EDBL

Driver Supply Voltage

For the XPS-EDBL driver, the parameter “Driver Supply Voltage” is fixed to 96 V and is not modifiable.

For all other configuration file parameters refer to section 5.10.8: “DRV02”.

5.11 Stage

In this configuration category, users are building the Global stage parameters section of the stages.ini file.

Example:

```
; --- Global stage parameters
; --- <Stage.GenericInformation>
SmartStageName = XMS160
Unit = mm
ConfigurationComment = No load
```

5.11.1 Type: GenericInformation

Configuration Comment

Use this parameter to write a comment in the stages.ini file

Unit

Use this parameter to display the unit (such as mm) that is reference in the configuration file for parameters with units of length.

Smart Stage Name

Use this parameter of the stage as a Newport's exclusive ESP technology to define the EEPROM smart stage name.

Service Form

Your Local Representative

Tel.: _____

Fax: _____

Name: _____

Company: _____

Address: _____

Country: _____

P.O. Number: _____

Item(s) Being Returned: _____

Model#: _____

Return authorization #: _____
(Please obtain prior to return of item)

Date: _____

Phone Number: _____

Fax Number: _____

Serial #: _____

Description: _____

Reasons of return of goods (please list any specific problems): _____

Newport®

Visit Newport Online at:
www.newport.com

North America & Asia
Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA

Sales
Tel.: (800) 222-6440
e-mail: sales@newport.com

Technical Support
Tel.: (800) 222-6440
e-mail: tech@newport.com

Service, RMAs & Returns
Tel.: (800) 222-6440
e-mail: service@newport.com

Europe
MICRO-CONTROLE Spectra-Physics S.A.S
9, rue du Bois Sauvage
91055 Évry CEDEX
France

Sales
Tel.: +33 (0)1.60.91.68.68
e-mail: france@newport.com

Technical Support
e-mail: tech_europe@newport.com

Service & Returns
Tel.: +33 (0)2.38.40.51.55

mks

Newport® Ophir® Spectra-Physics®