MIMO Gantry Commissioning

Application Note 139

This document describes the commissioning of a gantry axis equipped with two motors and two position sensors on each side of the gantry. This description can be applied to stiff or flexible gantries.

Table of Contents

Document AN139_MIMOGantryCommissioning_UserGuide_EP
Version 010, 2025-04-08
Source Q:\doc\ApplicationNotes\
Destination T:\Doc\ApplicationNotes
Owner dg

1 Introduction

The mechanical coupling of the two motors requires special considerations to reach a dynamic and stable behavior of the controller. The controller topology of the *Triamec* servo drives allows to transform the two coupled motor axes x_{Axis0} and x_{Axis1} into a decoupled system, with

- a rotational axis θ indicating the angular position of the gantry, and
- a linear axis x_{Lin} indicating the location of the balance point.

The coordinates are illustrated in Figure 1. The position difference $\Delta x = x_{Axis0} - x_{Axis1}$ is used as input for the rotational controller which is proportional to the angle θ [1].

This transformation into a linear and rotational coordinate system offers the following advantages:

- With the decoupled system, a stiffer and more stable controller can be configured with less complicated tuning.
- This approach can be applied to either stiff or flexible gantries.
- The static or dynamic location of the balance point can be considered easily.

Triamec servo drives ordered with the software option *GY*, support the transformation to a decoupled linear/rotational system. This document describes the required steps for the commissioning of the drive with the MimoGantry controller topology.

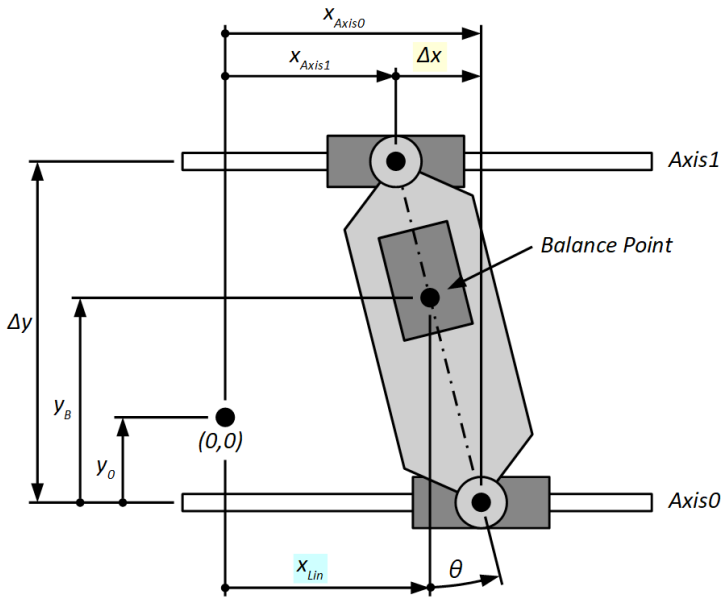

Figure 1: Gantry coordinate systems: Motor related coordinates (x_{Axis0}, x_{Axis1}) and linear/rotational coordinates (θ, x_{Lin}) .

1.1 Preconditions / Restrictions

The following preconditions have to be fulfilled and also some restrictions have to be considered before the commissioning of the gantry can start:

- The servo drive must have the software option *GY*. Check that register General.Signals.Identification-

1 For small values of θ compared to π the position difference can be expressed as $\Delta x \approx \theta\, \Delta y$.

s.Product shows the *GY* tag towards the end of the product name (e.g. *TSD80-10-TH-NONO-**GY***).

- It is possible to add the *GY* software option on an already delivered drive against an additional fee.
- The linear/rotational gantry option requires firmware 4.10.5 or higher and *TAM System Explorer* version 7.15 or higher. The current firmware and *TAM Software* can be downloaded from www.triamec.com.
- Only dual axis drives support the *GY* software option (TSD80, TSD130, TSD350).
- The encoders for the two axes have to be of the same type.
- Serial encoders (e.g. *DigitalEndat*, *DigitalBiss*, *DigitalTamagawa*, *DigitalNikon*) are supported with firmware 4.24 and higher.
- Transformation of positions in a *Tama* program is not supported.
- Modulo is not supported. Both parameters ModuloPositionMaximum and ModuloPositionMinimum must be set to zero.
- In Gantry mode, both encoders must be linked to Encoders[0] of the related axis. If, for example, the two option encoders are used, the parameter EncoderTopology must be set to *OptionA*.
- Gantry with dual-loop control (which involves both Controller[0] and Controller[1]) is not yet implemented. Please contact *Triamec Motion AG* in case you intend to setup a gantry with dual-loop control.
- For the commissioning it is assumed that the mechanical setup allows a low dynamic movement of the gantry with only one active motor.

Important	The encoder and the motor of one side have to be wired to Axis0 and the encoder and the motor of the other side to Axis1. Do not cross encoders and motors!

1.2 Implementation

As explained in the introduction, the position controller of the gantry system is implemented as a linear and a rotational axis. As this topology is integrated into the same register tree as used for the independent control of two axes, some registers have a different meaning when configuring it for a gantry.

The main changes concern the position controller as shown in Figure 2. The position controller of Axis[0] controls the linear motion and the position controller of Axis[1] the rotational motion of the gantry.

Chapter 3 explains usage and properties of all registers which change their meaning when used for a gantry.

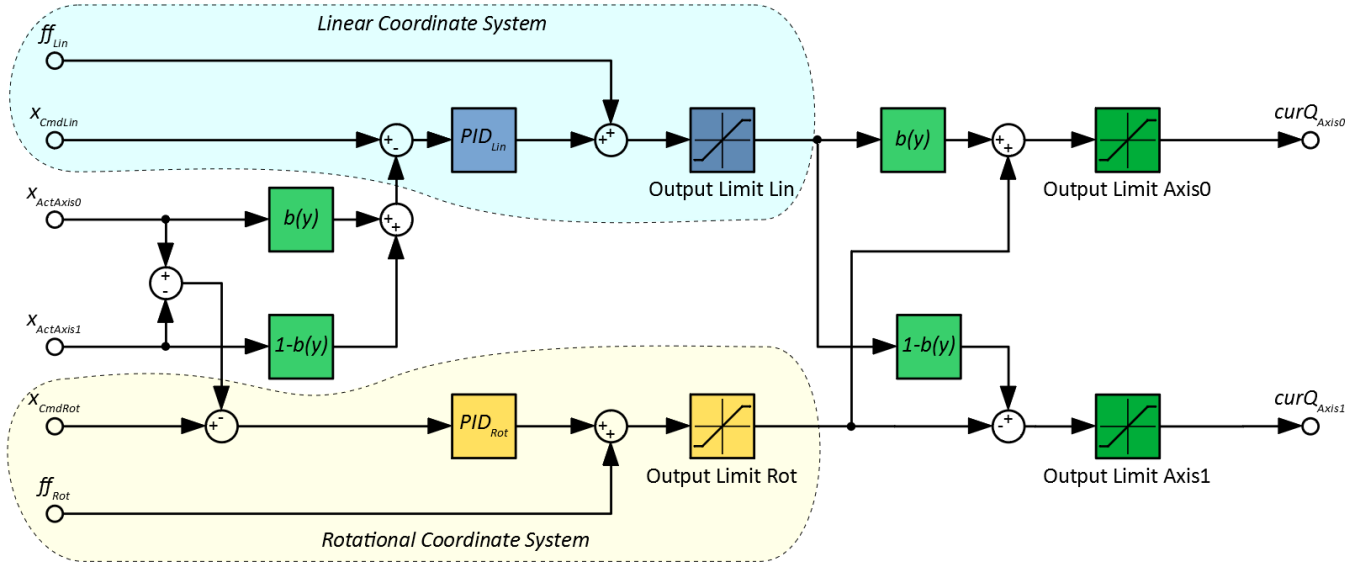

Figure 2: Structure of decoupled gantry controller.

2 Commissioning

This chapter explains the required steps to commission the gantry as a decoupled system with a linear and rotational axis. These steps reflect the best practice for a common gantry setup and may change for the particular system.

Warning	When executing the following steps, one-sided loads might be applied to the gantry. With a wrong parameterisation this could potentially cause damage to the mechanics. Therefore, it is important to set the output limits carefully and to not letting the loads exceed the tolerated values.
Warning	For stiff gantries: None or false compensation of large position offsets between the two encoders can cause forces that potentially damage the mechanics. Refer to chapter 2.7 before executing motion commands.

2.1 TAM Parameter Preparations

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

First, all known parameters (motor data, encoder data, position unit, etc.) need to be set in the register tree for both Axes[0] and Axes[1] – see [1] section 5.1 and 5.2 for the initial setup of the parameters.

For the preparation of the parameters the following registers need special consideration:

General.Parameters.ControllerTopology:

- As properties like the current controller and the commutation are not yet ready, ControllerTopology has to be set to Standard for the first part of the commissioning. At a later stage, this parameter will be set to MimoGantry.

Gantry.Parameters:

- Set the value of BalanceOffset to the fraction of the total linear inertia which affects Axis[0]. If the balance point is centered, BalanceOffset is 0.5.
- As the dependency of another axis is not yet considered, BalanceSlope is set to zero.
- With CurrentLimit0 and CurrentLimit1 the desired current applied to the motors can be restricted. Both parameters are set about 10% below the peak current of the motor or the drive, whichever is smaller.
- The parameter AlignAfterHoming must remain False until the homing sequence is configured and successfully tested.

Axes[].Parameters.PathPlanner:

- The path planner parameters of Axes[0] define the dynamic properties of the linear motion while the internal path planner is used.
- The path planner parameters of Axes[1] define the dynamic properties of the rotational motion. It is recommended to use low dynamics as this path planner is only used, for special tasks, such as aligning the rotational position to zero after homing of a soft gantry.

Axes[].Parameters.PositionController:

- For the initial setup the Controller[] parameters Kp, Ki and Kd should be set to zero for both axes.
- The OutputLimit of Axes[0] governs how much current is available for the linear motion of the gantry. Set this value to Gantry.Parameters.CurrentLimit0 + Gantry.Parameters.CurrentLimit1.
- Set Controllers[0].IntegratorOutputLimit to 40% - 80% of Axes[0].Parameters.Motor.NominalCurrent+ Axes[1].Parameters.Motor.NominalCurrent.
- To protect the mechanical setup against too much torsional tension, the value of the parameter OutputLimit of Axes[1] should be reduced to a tolerated value.
- Also, the value of the PositionErrorLimit of Axes[1], which corresponds to the difference between encoders, should be set to a value within the tolerated rotational deviation of the two encoders.
- The parameter FeedForwardAcceleration of Axes[0] can be calculated based on the total moved mass m_{tot} and the torque constant K_t of the motors (equation 1). For Axes[1], the parameter is zero.

$$FeedForwardAcceleration = \frac{m_{tot}}{K_t} \quad (1)$$

Axes[].Parameters.Commutation:

- For the commissioning of the gantry, the PhasingMethod has to be set to RotorAlignment and EnablingMethod to ForcePhasing. After the position controller is configured, these parameter can be set to AngleSearch and Automatic in order to avoid movement during phasing.
- As long as PhasingMethod is set to RotorAlignment, the RampRiseTime should be set to about 1s to 4s to limit the excitation during phasing. Later this parameter will be reduced to speed up the phasing.

Axes[].Parameters.CurrentController:

- For the initial setup, the parameters Kr and Tn should be set to zero for both axes.

Verification of the Encoder

Before continuing with the commissioning, it is recommended to check the encoder scale and counting direction. This can be done by manually moving the axis by a certain distance in positive direction and observe the encoder position of both axes with the scope.

- Axes[0].Signals.PositionController.Encoders[0].Position
- Axes[1].Signals.PositionController.Encoders[0].Position

In case the direction or the scale does not match, adjust the following parameters:

- Axes[0].Parameters.PositionController.Encoders[0].InvertDirection
- Axes[0].Parameters.PositionController.Encoders[0].Pitch
- Axes[1].Parameters.PositionController.Encoders[0].InvertDirection
- Axes[1].Parameters.PositionController.Encoders[0].Pitch

In case the encoder signal shows no movement at all, check the configuration and the connection of the encoders.

2.2 Current Controller

This section describes the tuning of the current controller.

- Make sure all parameters are set as described in section 2.1. Especially General.Parameters.ControllerTopology must be set to Standard.
- To determine the controller parameters, execute a current controller tuning for both axes as described in [1] section 5.4.
- Parameters Kr and Tn of the current controller can be set for both Axes[0] and Axes[1] individually. In most cases the same parameters can be used for both axes — assuming the two motors are identical.

Warning As soon as the parameters for the current controller are committed, the axis will execute a movement in the range of one pole pair pitch when enabling the axis. Later, this movement will be avoided by setting PhasingMethod to AngleSearch.

Verification of the Phasing, Motor Direction and Magnetic Pitch

To verify if the phasing is functioning as expected, see ‘Verification of the Setup’ in [1].

To check the motor direction and magnetic pitch execute the test sequence described in ‘Verification of the Setup’ – ‘Verification of the Motor Direction and Magnetic Pitch’ in [1] for both, Axis[0] and Axis[1].

2.3 Trimming of the Balance Gain

The balance gain defines the load distribution to the two motors of the gantry and is equivalent to the center of gravity of the gantry axis relative to the distance between the two motors. The parameter BalanceOffset (and in case there is a dependency with an other axis, the BalanceSlope parameter) can be used to configure the balance gain. The value of balance gain is evaluated according to equation (2).

$$b = BalanceOffset + BalanceSlope \cdot y \quad (2)$$

In many cases the balance gain can be determined in advance, for example, based on a CAD model and the mass distribution. For a static symmetric setup the balance gain is 0.5. This is achieved by BalanceOffset set to 0.5 and BalanceSlope to 0.0 accordingly.

Balance Parameters based on Mass Distribution

If we consider a setup according to Figure 3 the parameters BalanceGain and BalanceOffset can be calculated according to the following equations:

$$b = \underbrace{1 - \frac{m_1 y_{m1} + m_0 y_{m0}}{\Delta y (m_0 + m_1)}}_{BalanceOffset} + \underbrace{\frac{-m_0}{\Delta y (m_0 + m_1)}}_{BalanceSlope} y \tag{3}$$

$$BalanceOffset = 1 - \frac{m_1 y_{m1} + m_0 y_{m0}}{\Delta y (m_0 + m_1)} \tag{4}$$

$$BalanceSlope = \frac{-m_0}{\Delta y (m_0 + m_1)} \tag{5}$$

with

- m_0: movable mass of the *y* axis
- m_1: movable mass of gantry axis on *x* direction without m_0
- Δy: distance between the two motors of Axis0 and Axis1 (point of force transmission)
- y_{m0}: distance between the point of force transmission of Axis0 and the center of gravity of m_0 if the y axis is at encoder zero
- Y_{m1}: distance between the point of force transmission of Axis0 and the center of gravity of m_1
- *y*: encoder position of the *y* axis.

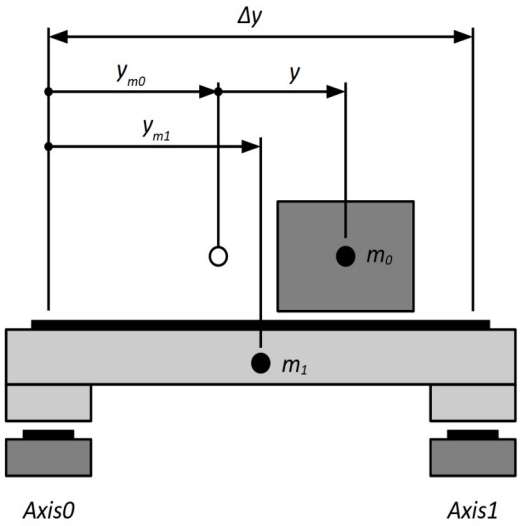

Figure 3: Setup for calculation of the parameters BalanceOffset and BalanceSlope.

Soft Gantry

For soft gantries the balance gain can also be determined with a Bode measurement by executing the following steps:

- Preparation: Make sure the General.Parameters.ControllerTopology is set to Standard.
- Execute a Bode measurement for Axis[0] and Axis[1] each with **Method** set to **Open Loop Bode**.
- Open both measurements in the **Bode Tuning** window and measure at the same frequency the gain g_0 for Axis[0] and g_1 for Axis[1] in the lower frequency range in *dB*. With this the balance gain *b* can be evaluated according to the following equation:

$$b = \frac{1}{10^{\frac{g_0 - g_1}{20}} + 1} \tag{6}$$

- In case there is no dependency with another axis, the parameter BalanceOffset can be set to *b*.
- If there is a dependency to another axis (in the following called axis *y* in accordance with Figure 1), repeat the measurement above for different positions of the axis *y* and fit BalanceOffset and BalanceSlope into the measurement according to the equation (2).

Position of Axis Y

To evaluate equation (2), the servo drive needs to know the position of axis *y*. Therefore, the position of axis *y* has to be written into the following register:

- Axes[1].Commands.PathPlanner.StreamX

Depending on the control system this can be done in several ways:

- Options with *TwinCAT* and *Tria-Link*: (The position is given in SI units)
  - With *Trial-Link*, it is possible to directly cross publish data from one drive to another. See [3] for how to set up cross publishing with *TwinCAT*.
  - With the function block *TL_publishMaster2Slave*, the setpoint of *y* in *SI* units can be transmitted to Axes[1].Commands.PathPlanner.StreamX (see also [4]).
  - If the requirements regarding real-time are low, the asynchronous *TL_MC_RegisterWrite* function block can be used to transmit the position in *SI* units (see also [5]).
- With *EtherCAT*:
  - In the **I/O** interface of the second axis of the drive, set the cyclic data *ModeOfOperation* to 8 and *TargetPosition* to the position of axis *y*. The unit of position *y* must match the unit of the drive configuration and therefore it is the unit of the gantry linear axis (see also chapter 3.2).

2.4 Position Controller

The tuning of the position controller is done in the linear/rotational coordinate system. Therefore, the parameter General.Parameters.ControllerTopology has now to be set to MimoGantry.

Warning	With the change of the ControllerTopology to MimoGantry the assignment of some signals change from the linear to the rotational coordinate system. See chapter 3 for an overview of the affected registers.
Notice	It is recommended to rename your axes at this stage to simplify further commissioning

and especially reduce the probability of errors. Add the suffixes *"linear"* to Axes[0] and "*rotational"* to Axes[1]. Refer to chapter 5.1.2 in [1] on how to change the name.

Tuning of the linear Axis

For the tuning of the position controller the following sequence is recommended:

- Open the **Measure Bode Plot** module of **Axis 0.** First set **Method** to **Gantry Axis[0]: Linear**. Then configure the measurement parameters (see also chapter 4.3.2 in [1]). The additional parameter **ActualCurrentQLin** is set to the same value as **ActualCurrentQ**. As the measurement is running with a closed current loop, make sure that the register OutputLimit of the position controller does not restrict the excitation of the system. Execute the measurement and save it.
- For the tuning of the linear position controller, open the **Tune Controllers** module for **Axis 0** with the corresponding Bode measurement. Setup the controller as described in [1], chapter 5.4.
- See [1] section "Verification of the Setup" - "Position Controller" on how to verify the controller tuning in time domain.

Warning The parameter **ActualCurrentQLin** limits the amplitude of the harmonic excitation in the gantry coordinate system. Also all remarks in chapter 4.3.2 in [1] apply.
Set parameters with caution, as excessive excitation can damage the axis.

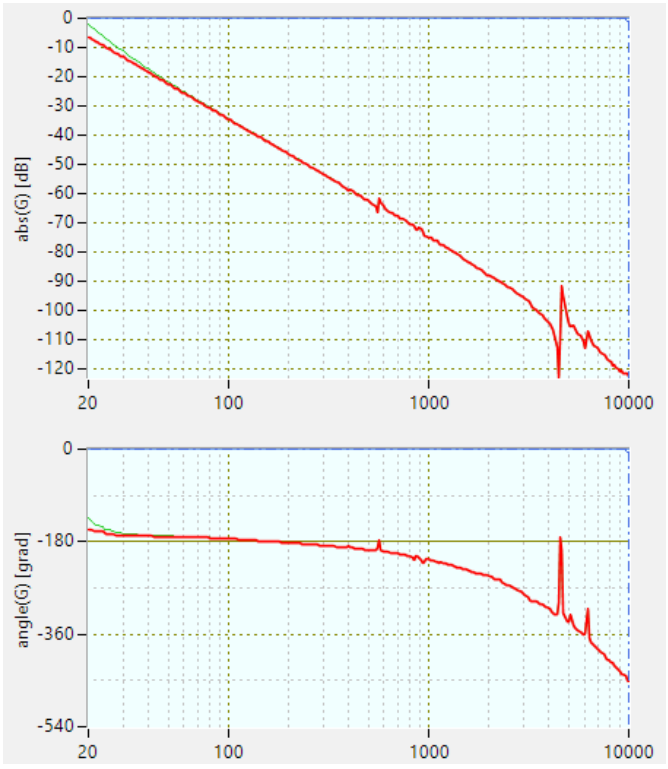

Figure 4: Typical Bode measurement of the linear axis of a gantry.

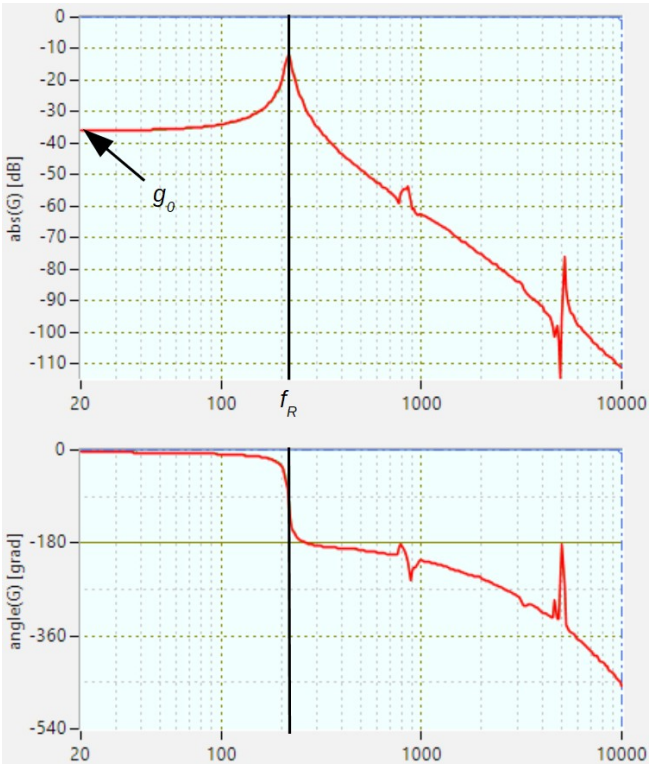

Figure 5: Typical Bode measurement of the rotational axis of a stiff gantry with a gain g0 of -36dB and resonance frequency f_R of 218Hz.

Tuning of the rotational Axis

The following sequence is recommended:

- Open the **Measure Bode Plot** module of **Axis 1.** First set **Method** to **Gantry Axis[1]: Rotational**. Then configure the measurement parameters (see also chapter 4.3.2 in [1]). The additional parameter **ActualCurrentQRot** limits the maximum applied momentum on the rotational axis. Reduce this parameter so that the hardware can sustain the resulting momentum. As the measurement is running with a closed current loop, make sure that the register OutputLimit of the position controller does not restrict the excitation of the system. Execute the measurement and save it.
- For the tuning of the rotational position controller, open the **Bode Tuning** Module for **Axis 1** with the corresponding Bode measurement. Consider the following notes regarding soft and stiff gantry setups.
- See [1] section “Verification of the Setup” - “Position Controller” for how to verify the controller tuning in time domain.

Warning The parameter **ActualCurrentQRot** limits the amplitude of the harmonic excitation of the axis. Also all remarks in chapter 4.3.2 in [1] apply.
Set parameters with caution, as excessive excitation can damage the axis.

Soft gantries show similar characteristics on the rotational axis as a common axis. Therefore, the Bode measurement looks similar to the measurement displayed in Figure 4. The same approach for tuning can be applied as described in [1]. Proceed with commissioning in the next chapter.

Stiff gantries show a different characteristic for the rotational axis which resembles to a spring mass system (see Figure 5). Depending on the relation between the resonance frequency f_R and the desired bandwidth of the controller f_B, a different approach for the tuning has to be used:

- If the resonance frequency f_R is smaller than the desired bandwidth f_B, the same approach for the tuning can be applied as described in [1].
- If the resonance frequency f_R is in range of the desired bandwidth f_B, the settings according to equation 7 are recommended as an initial approach. Use the **Additive, Gain** structure for the controller in this case. See Figure 5 on how to determine the values for the gain g_0 and resonance frequency f_R.

$$\mathrm{Kp}=0\,; \quad \mathrm{Ki}=10^{\frac{-g_0}{20}}\frac{2\pi f_B^3}{2f_B^2+f_R^2}\ldots 10^{\frac{-g_0}{20}}2\pi f_B\,; \quad \mathrm{Kd}=\frac{2\mathrm{Ki}}{(2\pi f_R)^2}\,; \quad \mathrm{T1}=0.0002\,s\,; \tag{7}$$

- If the resonance frequency f_R is significantly higher than the desired bandwidth f_B, it is recommended to apply a notch filter at f_R and then increase the integrator gain Ki until the desired bandwidth is reached.
- Further optimization of the controller may be required based on the criteria of Bode and Nyquist. In most cases the following condition should be maintained.

$$\mathrm{Ki}\leqslant 2(\pi f_R)^2\mathrm{Kd} \tag{8}$$

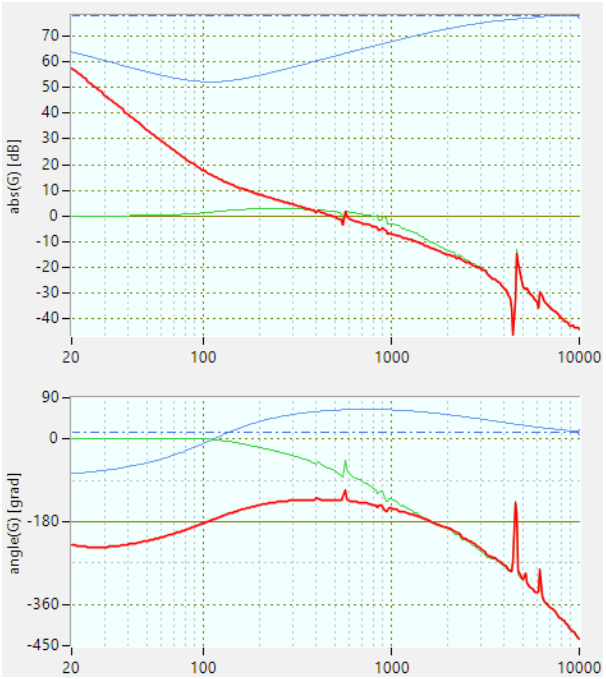

Figure 6: Bode plot of the tuned linear axis.

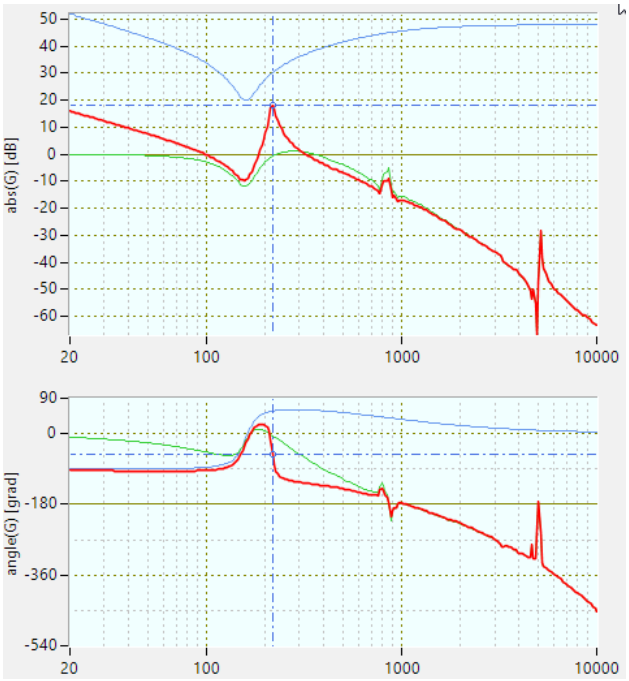

Figure 7: Bode plot of the tuned rotational axis of a stiff gantry.

Acceleration Feed Forward

The acceleration feed forward of the linear axis can also be determined or verified based on the linear Bode measurement and needs to be set in register Axes[0].Parameters.PositionController.FeedForwardAcceleration (see also [1] section 5.4.6 and 5.5.4).

As in normal case no motion is executed with the rotational axis, the FeedForwardAcceleration of Axes[1] is set to zero.

2.5 Completion of the Commissioning

Phasing

With the following adjustments in node Parameters.Commutation of Axes[0] and Axes[1], movement during phasing is avoided:

- Set PhasingMethod to AngleSearch
- Set RampRiseTime to about 0.1s and RampConstTime to 0.5s.

If the phasing with this new setup works as expected (check also [1], chapter 5.5.2), set the EnablingMethod to Automatic. The phasing will now only execute after a restart of the servo drive.

Limits

Before the gantry is used for normal operation, it is recommended to verify the value of the following parameters regarding protection of the mechanics and the dynamic requirement. If some values were reduced during commissioning, i.e. for safety reasons, they can now be reset to the original value.

- Gantry.Parameters.CurrentLimit0
- Gantry.Parameters.CurrentLimit1
- Axes[0|1].Parameters.PositionController.OutputLimit
- Axes[0|1].Parameters.PositionController.Controllers[0|1].PositionErrorLimit
- Axes[0|1].Parameters.PositionController.Controllers[0|1].IntegratorOutputLimit
- Axes[0|1].Parameters.CurrentController.OutputLimit

Save the Configuration

When the configuration is done and tested, it is recommended to persist the configuration on the servo drive and to save it on the PC (see section 3.5 in [1]).

2.6 Homing

The homing sequence for gantry is similar to the homing sequence of a normal axis (see [2]). Homing a gantry is commanded on Axis[0], which will start both axes. The parameters for the second search can be individually set for Axis[0] and Axis[1] to support individual index marks for each axis.

- The Method in Axis[0].Parameters.Homing defines the general homing sequence. The parameters in SecondSearchMove of Axis[0] specify the search of the home position on the encoder of Axis[0]. Parameters in SecondSearchMove of Axis[1] specify the search of the home position on the encoder of Axis[1]. These two searches run simultaneously.
- After the homing moves, the positions of both axes are set in such a way, that the position at the reference of Axis[0] corresponds to the value of Axis[0].Commands.Homing.ReferencePosition and the position of Axis[1] to the value of Axis[1].Commands.Homing.ReferencePosition.
- In case only the encoder of Axis[0] is used for the homing sequence, set Axes[1].Parameters.Homing.SecondSearchMove.EventInput to Skip. In this case, both encoder positions are set to the value in Axis[0].Commands.Homing.ReferencePosition and there will be no position offset correction.

Warning For stiff gantries: Refer to chapter 2.7 before executing a homing routine.

- To start the homing sequence on a soft gantry, select Start in the Axis[0].Commands.Homing.Command register and press Enter.

If the parameter Gantry.AlignAfterHoming is False, when homing is executed, the position offset between the two encoders will remain. If Gantry.AlignAfterHoming is set to True the rotational axis executes a move to zero to align the axis after the homing.

Warning Before Gantry.AlignAfterHoming is set to True the homing sequence has to be configured correctly and tested successfully.

2.7 Encoder Alignment

The rotational controller of the gantry regulates rotational deviations to zero. In case the two encoders show some deviation, this leads to constraining forces on stiff gantries. To avoid this forces, the two encoders have to be aligned to each other.
This section describes, how to setup the *Tama* program `GantryEncoderAlignmentDemo.tama` which measures the deviation with a scan move and applies the resulting correction to the encoder position.

Load the Tama Program

1. The source code with the *Tama* program `GantryEncoderAlignment.cs` can be requested from *Triamec Motion AG*.
2. The *Tama program* might be adjusted depending on the application. The following implementation needs to be verified and might be adjusted:
   - The move range for the scan is defined by the following registers .
     - Axes[0].Parameters.PathPlanner.PositionMinimum
     - Axes[0].Parameters.PathPlanner.PositionMaximum
   - The duration of the scan move is defined by the constant `cScanDuration` which is set to 30s by default for one direction. The scan velocity is evaluated from the move range and the duration.
   - The alignment can be activated when homing and scan are done. As long as the alignment is not activated or homing is not done, the program disables the rotational controller.
   - The internal *Table* Application.Tables.Small1 is used to store the values used for the alignment.
   - Parameter Application.Parameters.Booleans[7] is used to indicate a successful scan.
   - To indicate a valid scan after a reboot with parameter Application.Parameters.Booleans[7], the configuration need to be stored persistent on the drive, after a successful scan.
   - If Application.Parameters.Booleans[7] is set and homing is done, the compensation is activated automatically.
3. To build the *Tama* program Visual Studio (Express) 2017 is recommended.
4. See [1], section 3.7, on how to download the *Tama* program, enable the *isochronous TamaVM* and save it persistent on the drive.

Run the Encoder Alignment

To run the encoder alignment, the gantry axis must be set up correctly and enabled, with the *Tama* program loaded running.

The register Application.TamaControl.IsochronousMainCommand allows to command the encoder alignment program as follows:

Command		Description
0	`NoAlignment`	Encoder alignment is not applied.
1	`HomeScanApply`	With this command the following sequence is initiated: • Homing sequence • Scan movement • Application of the encoder alignment
2	`HomeApply`	With this command only the homing is executed without doing a scan. This command requires a valid previous scan.
3	`ScanApply`	With this command only the scan is executed without doing a homing. This command requires a valid previous homing.
4	`Apply`	The encoder alignment is enabled. This command requires a valid homing and scan.

If the sequence is successfully executed the IsochronousMainCommand will settle to 4, indicating that the alignment is automatically applied.

3 TAM Registers with MIMO Gantry

This chapter explains the usage and meaning of all the registers which are affected, in case the parameter General.Parameters.ControllerTopology is set to MimoGantry. All other parameters can be handled the same as for two individual axes as described in [1].

3.1 The Gantry Node

If the software option *GY* is enabled, the register tree shows a Gantry node with registers related to the gantry setup.

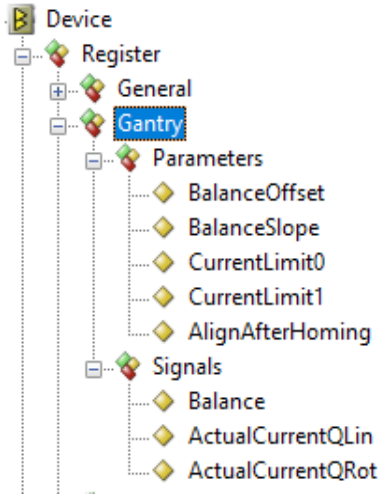

Figure 8: Gantry node in register tree.

Parameters

Gantry.BalanceOffset:

- The value of the parameter BalanceOffset determines the fraction of the total linear inertia which affects Axis[0]. If the balance point is centered, BalanceOffset has to be set to 0.5.
- The fraction of the total linear inertia which affects Axis[1] is 1 minus BalanceOffset.
- In case the position of the balance point depends on the position of another axis *y*, the value of the BalanceOffset is the fraction of the inertia when the position of *y* is zero (see also equation 4).

Gantry.BalanceSlope:

- With the parameter BalanceSlope, the effect of another axis *y* to the position of the balance point can be considered (see also Figure 1).
- The Balance factor *b* is calculated based on the position *y* according to equation (3).
- The position *y* has to be provided, for example, by the superior control system (e.g. TwinCAT) as a setpoint for Axis[1] and has to be written to register Axes[1].Commands.PathPlanner.StreamX.
- In case the balance point is not significantly impacted by the position of another axis or if the posi-

tion of the other axis is not available, the value of BalanceSlope should be set to zero.

Gantry.CurrentLimit0:

- This parameter restricts the absolute setpoint current for the motor connected to Axis[0]. This can be useful to limit the maximum applied force by this motor.

Gantry.CurrentLimit1:

- This parameter restricts the absolute setpoint current for the motor connected to Axis[1]. This can be useful to limit the maximum applied force by this motor.

Gantry.AlignAfterHoming:

- If True, the rotational axis will be aligned to zero after homing.

Signals

Gantry.Balance:

- This register shows the actual balance factor evaluated according to equation (3).

Gantry.ActualCurrentQLin:

- This is the actual current in linear coordinates.

Gantry.ActualCurrentQRot:

- This is the actual current in rotational coordinates.

3.2 Parameters with New Assignment

This section describes the properties of parameters which change their assignment with the gantry setup.

Parameter.PathPlanner:

- The registers in Axis[0].Parameter.PathPlanner define the dynamic properties for the linear movement if the internal path-planner is used.
- The registers in Axis[1].Parameter.PathPlanner define the dynamic properties for the rotational movement e.g. to align the rotational axis after homing.

Parameter.PositionController:

- Encoders: The meaning of the encoder parameters does not change with the gantry topology. They still define the properties of the assigned physical encoder.
- Controllers, FeedForward, MasterPositionSource, ExcentricityCompensation: The parameters of Axis[0] are assigned to the linear axis, the parameters of Axis[1] to the rotational axis.
- PositionUnit: Set the unit of both axes to the unit of the linear axis.

Parameter.Commutation:

- The PhasingMethod parameters of both axes need to be set to the same value.
- In case PhasingMethod is set to RotorAlignment, the phasing of the two axes is done sequentially. First, the phasing is done for Axis[0], followed by Axis[1] using the parametrization of the corresponding registers. With this phasing method, the axis *will execute a movement* during the phasing of up to one pole-pair pitch. This phasing method only requires a proper setup of the current con-

troller for both axes.

- In case PhasingMethod is set to AngleSearch, the phasing of the two axes is done synchronously within the linear-rotational coordinate system. With this method, the axis *will not move* during the phasing. Nevertheless, it requires the configuration of the linear and rotational position controller before it can be used.
- In most cases, the method RotorAlignment should only be used during configuration of the gantry system and changed to AngleSearch when the position controller is configured.
- In general, the commutation parameters for Axis[0] and Axis[1] should be identical.

Parameter.Homing:

- The homing is defined by the parameters Axes[0].Parameters.Homing and Axes[1].Parameters.Homing.SecondSearchMove.
- The homing parameters in Axis[0] define the general homing sequence while parameters in SecondSerachMove of Axis[0] specify the search of the home position for the encoder of Axis[0]. Parameters in SecondSearchMove of Axis[1] specify the search of the home position for the encoder of Axis[1].
- In case the encoder of Axis[1] is not used for the homing sequence, set Axes[1].Parameters.Homing.SecondSearchMove.EventInput to Skip.

3.3 Commands with New Assignment

This section describes the functionality of commands of Axis[0] and Axis[1] which change their assignment with the gantry setup.

Commands.General

- For both Axis[0] and Axis[1], the EnableAxis and DisableAxis commands enable or disable both motors of the gantry.

Commands.PathPlanner:

- The commands of Axis[0] can be used to control the linear motion and the commands of Axis[1] to control the rotational motion e.g. to align the rotational axis after homing.

Commands.PositionController:

- The commands of Axis[0] can be used to handle the linear position controller and the commands of Axis[1] to handle the rotational position controller.
- The Active command can be used to deactivate or activate the linear or the rotational position controller.

Commands.TestGenerator

- The following commands apply to the linear gantry axis in case the command is executed on Axis[0], or to the rotational gantry axis in case the command is executed on Axis[1]:
  - StartPositionSin, StartPositionSquare, StartCurrentSin, StartCurrentSquare.
- The following commands apply to the motor of Axis[0] or Axis[1] respectively:
  - StartVoltageSin, StartCurrentSinStaticVector, StartVoltageSinStaticVector, StartRotatingVectorConstantCurrent, StartRotatingVectorConstantVoltage.

Commands.Homing:

- To start the homing sequence, Start has to be selected in the Command register of Axis[0]. The Start command on Axis[1] is not supported and must not be used.

- The commands used with absolute encoders (Invalidate, SaveEncoder, InvalidateEncoder) apply to the encoder of the corresponding axis.
- ReferencePosition: After homing, the position of both encoders is set in such a way that the position of Axis[0] and Axis[1] at the reference corresponds to the value of ReferencePosition of the related axis. This value is persistently stored on the drive, but *not* in the **TAM Configuration**.

3.4 Signals with New Assignment

This section describes the signals of Axis[0] and Axis[1] which change their assignment with the gantry setup.

Signals.PathPlanner:

- The signals of Axis[0] show the state of the linear path planning and the signals of Axis[1] the state rotational path planning.

Signals.PathInterpolator:

- The signals of Axis[0] show the state of the linear interpolator and the signals of Axis[1] the state of the rotational interpolator.

Signals.PositionController:

- Encoders: The meaning of the encoder signals does not change with the gantry topology and they do still show the states of the assigned physical encoder.
- Controllers, FeedForward, MasterPosition, MasterError, MasterVelocity, DesiredCurrentQ, DesiredCurrentD: The parameters of Axis[0] are assigned to the linear axis, the parameters of Axis[1] to the rotational axis.

Signals.TestGenerator:

- The assignment depends on the selected command. See Commands.TestGenerator in section 3.3.

References

[1] “Servo Drive Setup Guide, TSD and TSP Series”, ServoDrive-SetupGuide_EP022.pdf, Triamec Motion AG, 2022.

[2] “Homing Procedures and Setup”, AN141_HomingProceduresAndSetup_EP003.pdf, Triamec Motion AG, 2022.

[3] "Twincat Library: Cross Publishing data, Application Note", AN106_TwinCAT-CrossPublishing_EP002.pdf, Triamec Motion AG, 2016.

[4] "Twincat Library: Defining cyclic telegrams, Application Note", AN105_TwinCAT-CyclicTelegrams_EP005.pdf, Triamec Motion AG, 2020.

[5] "Twincat Library: Accessing Drive Registers, Application Note", AN109_TwinCAT-AccessingDriveRegisters_EP006.pdf, Triamec Motion AG 2019.

Revision History

Version	Date	Editor	Comment
001	2020-11-11	dg	Initial version
002	2021-10-04	dg	Section encoder alignment added.
003	2021-11-30	sm	General review
004	2022-03-29	dg	Verification of motor direction and phasing moved to setup guide
005	2022-11-16	dg, sm	Update template, general review
006	2023-02-10	sm	Improve encoder alignment guidance for stiff gantries, minor review changes
007	2023-10-30	dg, ab	Added warning for correct wiring, correct some internal links, minor review changes
008	2024-03-19	ab	Added prerequisite for used encoders
009	2024-08-23	yz	Added guidelines for current values on Axis0
010	2025-04-08	dg	Encoder requirements adjusted.

Triamec Motion AG
Lindenstrasse 16
6340 Baar / Switzerland

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

Disclaimer

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

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