Gantry Control

Flexible and Rigid Gantry

Application Note

agito

www.agito-akribis.com

Member of Akribis Systems group

Revision History

Version	Description	Date
1.0	Initial Release	06 Jun 2022

Contact Information

Manufacturer	Agito Akribis Systems Ltd., Member of Akribis Systems Group
Address	6 Yad-Harutsim St., P.O.Box 7172, Kfar-Saba 4464103
Telephone	+972-9-8909797
Website	www.agito-akribis.com

Copyright Notice

Products Rights

AGDx, AGCx, AGMx, AGAx, AGIx, and AGLx are products designed by Agito Akribis Systems Ltd. in Israel. Sales of the products are licensed to Akribis Systems Pte Ltd. under intercompany license agreement.

Agito Akribis Systems Ltd. has full rights to distribute above products worldwide.

Disclaimer

This product documentation was accurate and reliable at the time of its release.

Agito Akribis Systems Ltd. reserves the right to change the specifications of the product described in this manual without notice at any time.

Trademarks

Agito PCSuite is a trademark of Agito Akribis Systems Ltd..

Warranty

This product is warranted to be free of defects in material and workmanship and conforms to the specifications listed in this manual, for a period of 12 months from the shipment date from factory.

Contents

1 Introduction

1.1 Background

A motion-centric system designed for multi-axis operation with an overhead bridge is regularly called Gantry System. You will find them in applications related to Scanning, Digital Printing, Electronics Assembly, AOI (Automatic Optical Inspection) and Automation. In general, using Gantry System is ideal for dynamic and high duty-cycles processes, which requires high throughput and high precision.

1.2 Scope

This document provides the user with a basic and detailed explanation of how to use Agito PCSuite Gantry Control function. It is assumed that you are familiar with the use of similar software. It is also assumed that you are familiar with common concepts of motion systems and motion control.

This manual focuses only on concepts that may not be self-explanatory. Interface elements that are simple enough for the user to understand are not documented in this manual.

Agito PCSuite tools (screens) that refer to specific features of Agito controllers (such as motions and control filters) are explained in detail in the *Agito Controllers User Manual*.

This manual is based on Agito PCSuite version 10.7.0-4.0.

2 Setup

2.1 Equipment and overview

The typical setup topology is presented in the following figure:

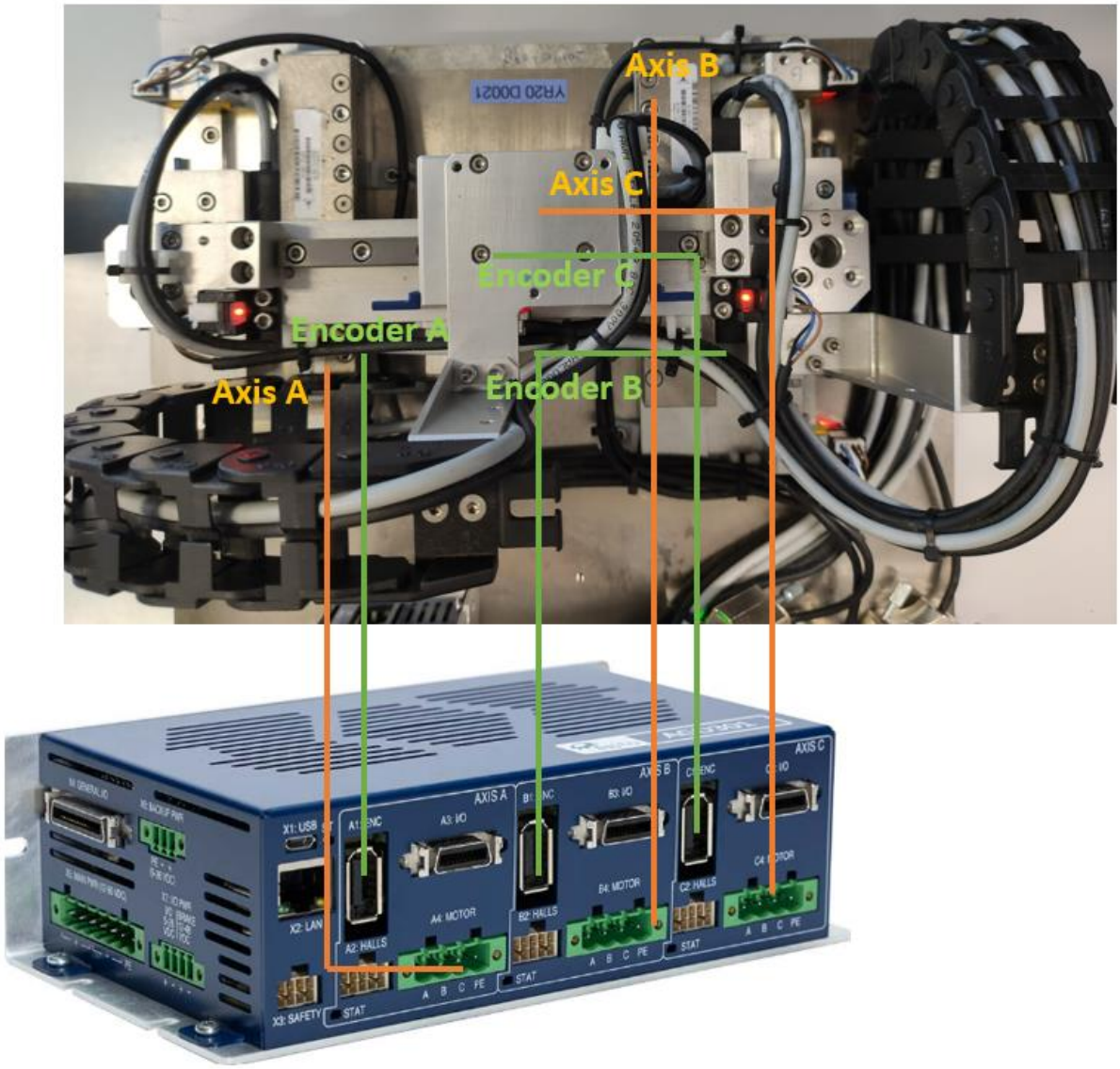

Figure 1. Standard setup topology

The example setup includes:

1. AGD301 motion controller with integrated drive, which act as motor drive and controller to perform gantry control.
2. Two AUM1 DC brushless linear motors with feedback act as gantry X1 (Axis A) and X2 (Axis B).
3. One AUM1 DC brushless linear motor with feedback act as gantry Axis C.

2.2 Wiring

AGD301: A4, B4 and C4 – Motor Power

Function	Pin Name	Pin #	Remarks
Motor Phase A	Phase A, M1	1	Motor Power
Motor Phase B	Phase B, M2	2	Motor Power
Motor Phase C	Phase C, M3	3	Motor Power, NC for voice coils
PE	PE	4	Motor PE

AGD301: A1, B1 and C1 – Encoder Port (Example shows pinout for Incremental encoder)

Function	Pin Name	Pin #	Remarks
5V	5V	1	Power for encoder
GND	GND	2	Power return for encoder
CLOCK+	Encoder_1+	3	Encoder signal input
CLOCK-	Encoder_1-	4	Encoder signal input
A+/SIN+	Encoder_2+	5	Encoder signal input
A-/SIN-	Encoder_2-	6	Encoder signal input
B+/COS+	Encoder_3+	7	Encoder signal input
B-/COS-	Encoder_3-	8	Encoder signal input
Z+/DATA+	Encoder_4+	9	Encoder signal input
Z-/DATA-	Encoder_4-	10	Encoder signal input

Note – Wiring for other controllers

This example uses Agito controller, AGD301 for the example. Wiring information for other controllers or encoder protocols can be found in their respective Product Manuals.

2.3 Motor setup and tuning

2.3.1 Setting of motor parameters in non-Gantry mode

Open Agito PCSuite software. Select CFG 

in CONFIG 

below and setting parameters.

Click ‘Next’ and set the motor type and number of pole pairs according to the test motor.

For linear motor, the encoder Resolution parameter depends on the motor’s electrical cycle (one pole-pair distance) and encoder resolution. Click ‘Next’ and fill in the relevant data.

Fill in the current, voltage and speed limit according to the motor manual, and make sure the parameters are smaller than their limits for safety protection.

2.3.2 Adjust the current loop, velocity loop and position loop separated

Select Auto-Phasing 

in the tune option 

, set Auto-Phasing mode, method, step time, step voltage and step accuracy. Perform Auto-Phasing.

Select current control 

in the tune option, set motor resistance and inductance according to the motor manual, adjust parameters of current PI such that MotoCurr follows CurrRef as close as possible in the recorder graphs, below is one example of tuning.

Select PIV 

in the tune option, adjust parameters of velocity PI, make the graphic of Vel approach to the graphic of VelRef as close as possible.

On the same page, adjust parameters of position PI, make the graphic of Pos approach to the graphic of PosRef as close as possible.

Note

Please notice that the above steps perform separately on the A axis and B axis!

2.4 Gantry tuning and homing

2.4.1 Gantry performance tuning

Once ready for gantry motion, switch to A axis, select GNTR 

in TUNE tab 

, click ‘Gantry On’.

If success, as shown here:

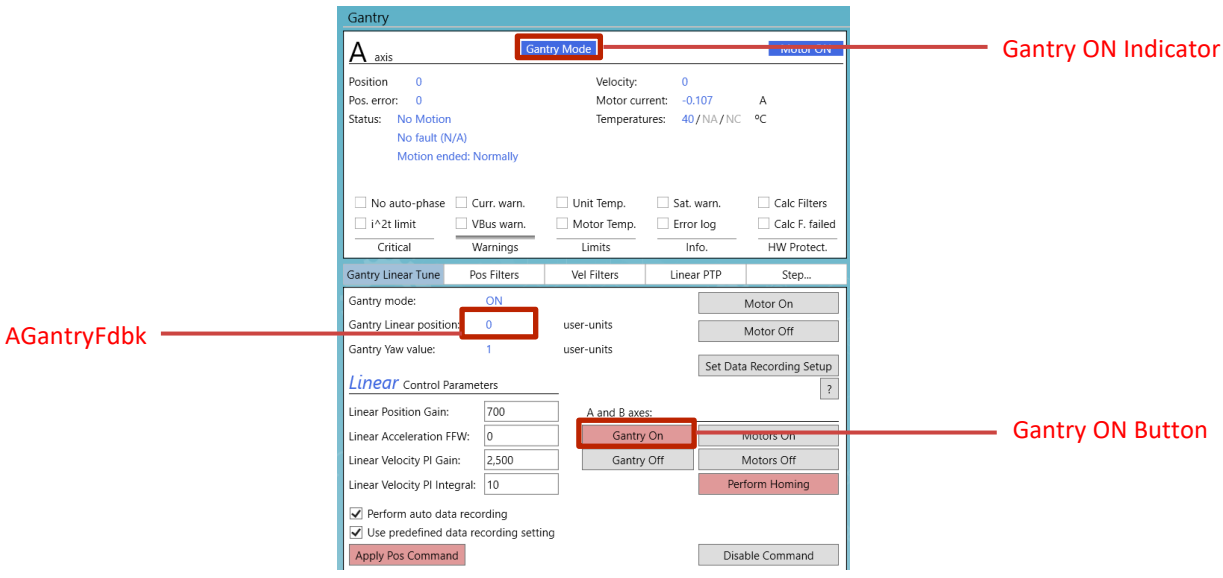

Figure 2. Gantry ON.

Apply Pos Command, and adjust parameters of Linear PI in A axis page, make the graphic of AGantryFdbk approach to the graphic of APosRef as close as possible.

AGantryFdbk = (APos + BPos+AGantryOffset) / 2

AGantryOffSet = APos - BPos when you do gantry ON.

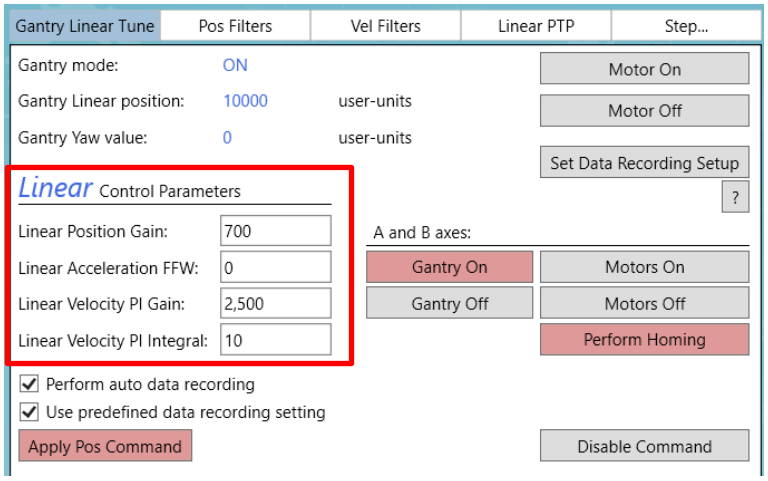

Figure 3. Gantry Linear Tuning.

Note

AGantryPosGain control the gain of X1&2 axis shown below, and BGantryPosGain control the gain of Yaw axis, and so on!

Note

When Gantry Mode is on, it better to reduce the parameters of single motor's PI, or it might be high frequency noise appear!

After the adjustment, turn to PTP mode, select repeat movement between Target1 and Target2. Observe the graphs, adjust the parameters of Gantry PI in B axis page to minimize the value of BGantryFdbk and AposErr.

BGantryFdbk = APos – BPos - AGantryOffSet

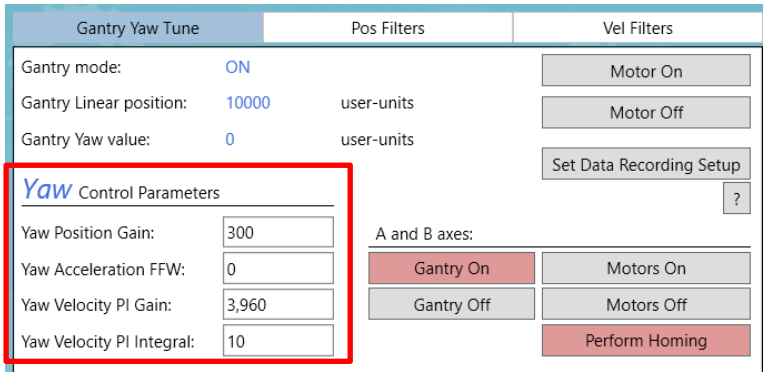

Figure 4. Gantry Yaw Tuning.

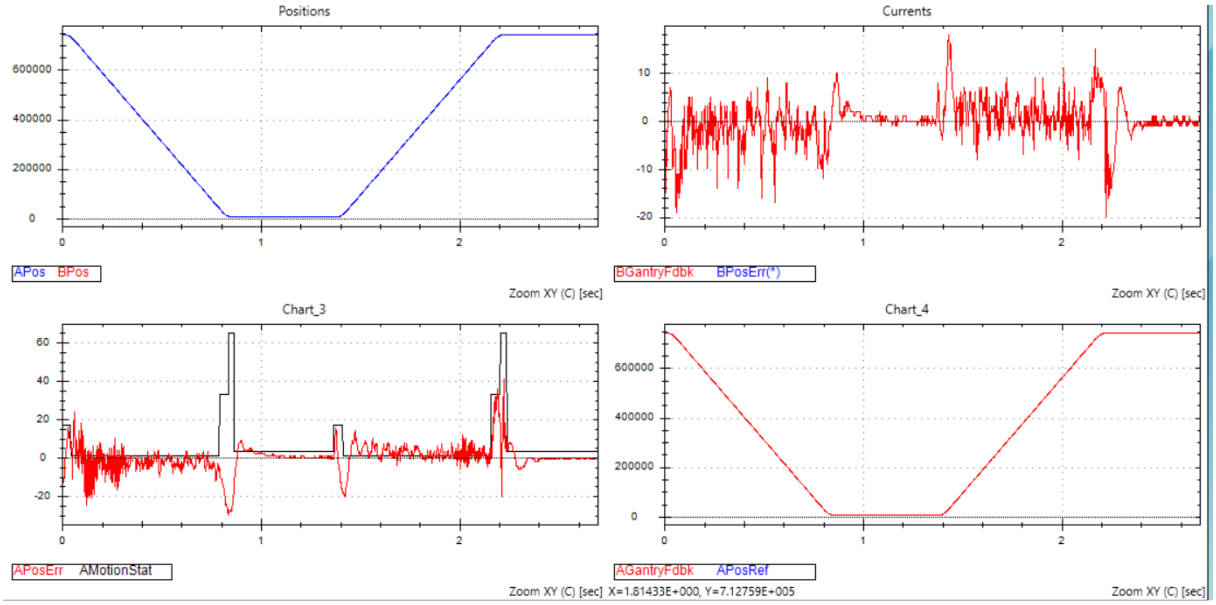

Figure 4. Gantry Yaw Tuning Data Recording.

2.4.2 Gantry yaw control

BGantryYawRef is used to do the gantry yaw control.

As shown below, the default BGantryYawRef value is 0, and can use Terminal to set value to BGantryYawRef.

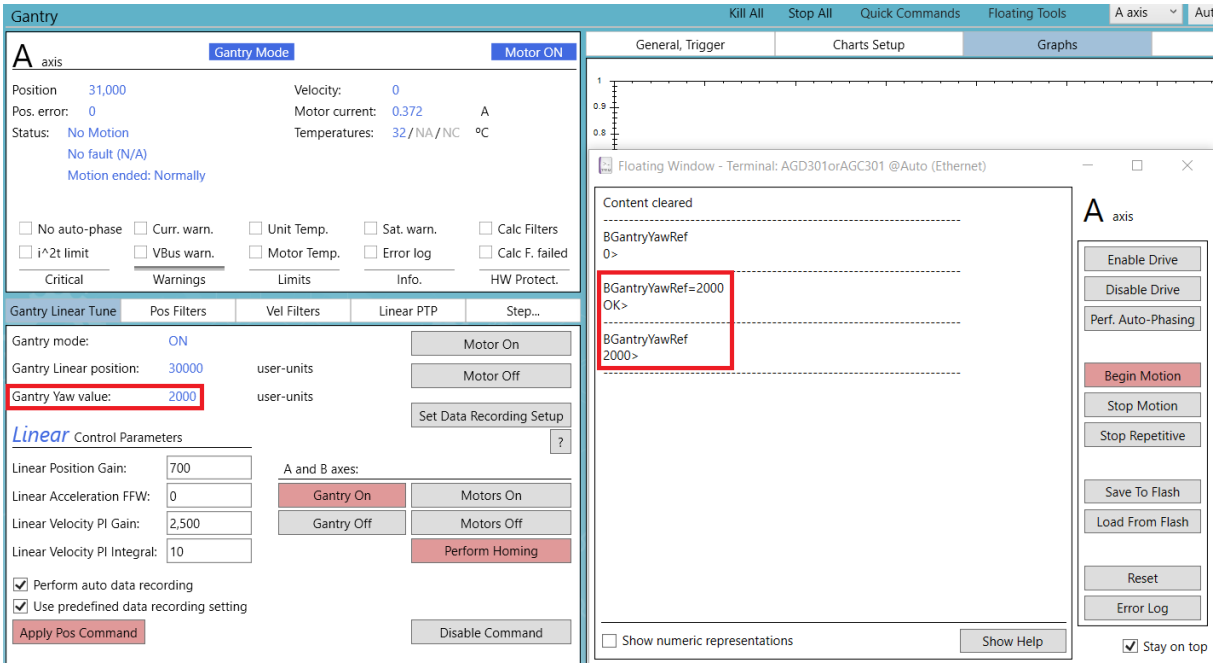

Figure 5. Gantry Yaw Value and Setting.

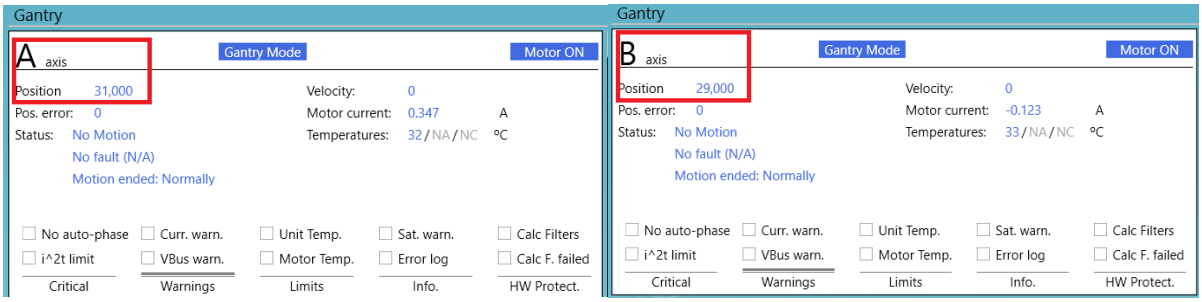

Figure 6. APos and BPos after Gantry Yaw Setting.

Note

Please notice that, once the BGantryYawRef is set, the A axis and B axis position will change immediately!

2.4.3 Do homing for flexible gantry

If you are using a flexible gantry, you can do homing by clicking 'Perform Homing' button under Gantry Panel.

Do homing before using gantry mode, Homing can be done with Gantry Mode disabled.

Select HOME 

in the TOOLS 

option, set Homing steps.

Select GNTR 

in the TUNE tap 

, click 'motors on' and 'Perform Homing':

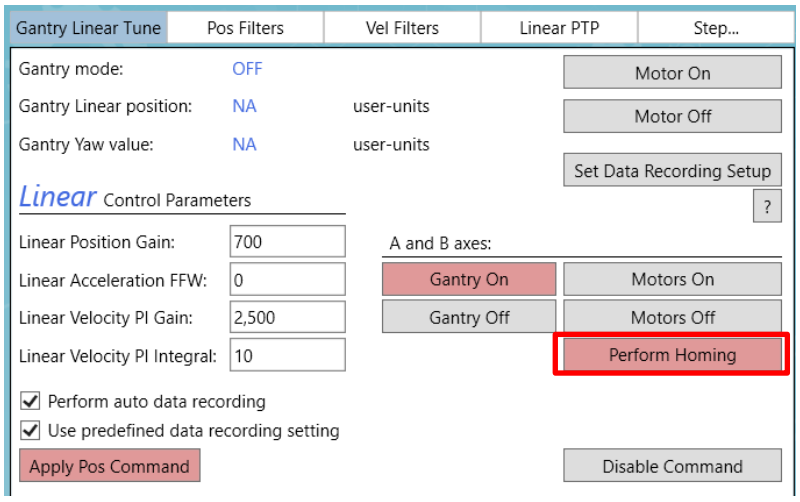

Figure 7. Perform Homing under Gantry Panel.

Axis A and B will start homing at same time follow the homing sequence you set under Homing Panel.

2.4.4 Do homing for rigid gantry

Please be careful if you are using a rigid gantry.

We do not suggest customer to use 'Perform Homing' and 'Gantry On' under Gantry Panel.

Instead, we suggest customer to write User Program in IDE+ to do the homing step and gantry on. You can use a AGendata variable to enable the 'perform homing' or 'gantry on'.

Agendata[601]: Instruction of Homing

Agendata[603]: Index difference between A and B (Measured with laser equipment)

Homing:

When Agendata[601] = 1, execute Homing sequence

1. Find index of A and B

2. A drags B and A homing

3. B makes a compensation move

2.4.5 Gantry error mapping

The Error Mapping function is applied on axis A and B individually.

If Error Mapping is not applied on Axis A, the A position is same with Gantry Linear Position.

When Error Mapping is applied on Axis A, as below, A position will change based on the mapping value setting. And Gantry Linear Position will follow the A Position Reference. B position will not change as there is no Error Mapping on B axis now.

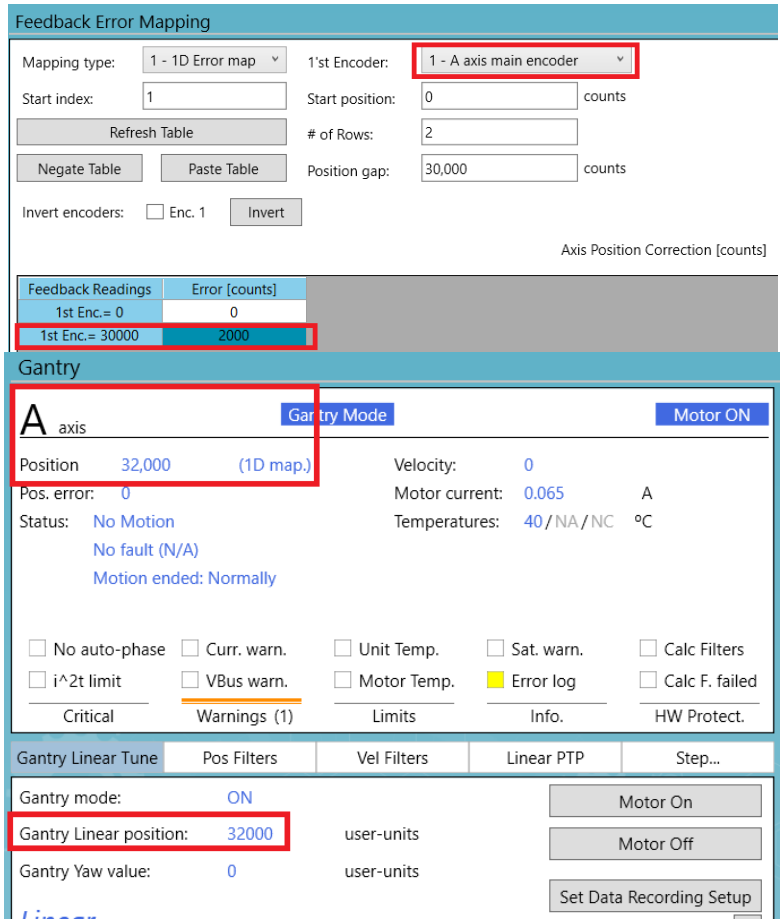

Figure 8. Error Mapping on Axis A.

If Error Mapping is applied on B axis, B position will change based on the mapping value setting. But Gantry Linear Position will now change, as it still follows the A Position Reference.

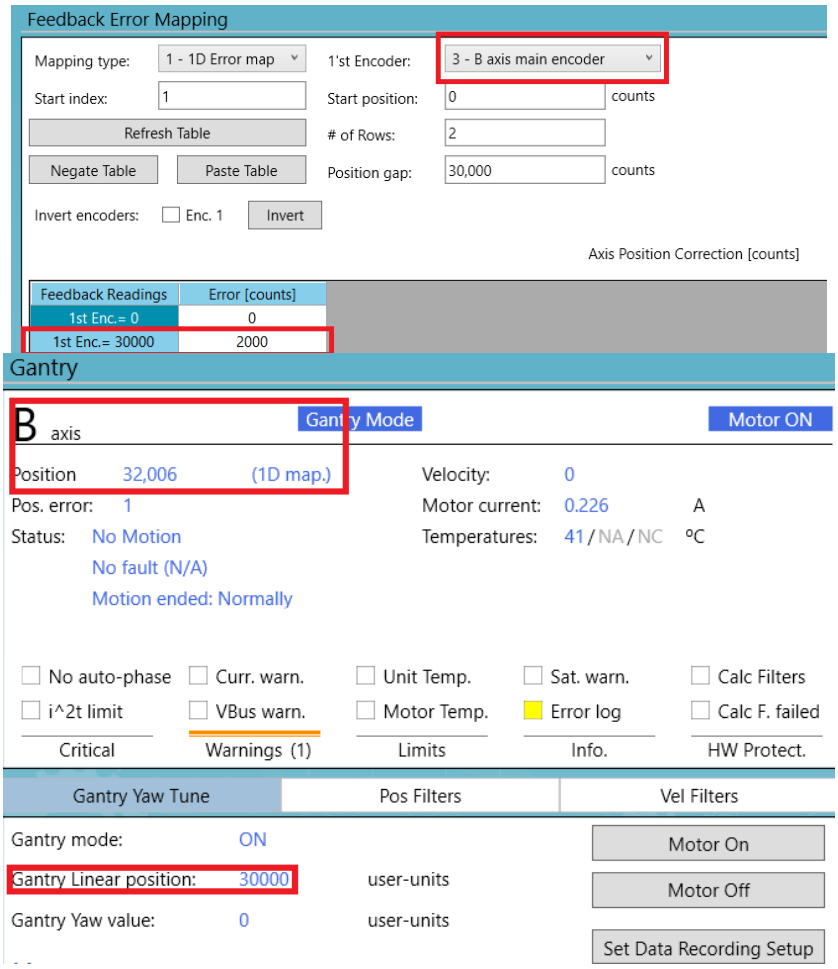

Figure 9. Error Mapping on Axis B.

3 Appendix

3.1 pup2 coding

```
#include Gantry Homing.puh2

#information    Set AGenData[600] to 1 for rigid gantry and 2 for flexible gantry
#information    Set AGenData[601] to the position value to set at index position of Axis A
#information    Set AGenData[602] to the relative distance required to move to A index (when at orthorgontal position, at B index position).
#information    Set AGenData[603] to 1 to begin homing
#information    Set AGenData[604] to 1 to do gantry on

#information    Read AGenData[620] for gantryHomingStat
#information        100 - Success
#information        0 - Homing triggered
#information        10 - Moving to RLS
#information        20 - Moving to FLS while searching for lock
#information        30 - Moving to Axis A Index position

#information        -1 - Gantry type not defined
#information        -2 - Could not find index (Axis A index lock not triggered)

main([10,30],[5,20],[800,1000])
        AMotorOn = 0        //Turn off both motors on event that the system is in gantry mode.
        BMotorOn = 0

        while(1)            //Poll for AGendata signal to begin homing or gantry on
                AWaitTime, 100
                if(gantryHomingOn == 1)     //RigidHoming is saved in Agendata[601]
                        if(gantryType == 1)
                                HomeRigidGantry()           //Execute homing sequence for rigid gantry
                        else if (gantryType == 2)
                                HomeFlexibleGantry()        //Execute homing sequence for flexible gantry
                        else
                                gantryHomingStat = -1
```

```
                    end
                end
                if(gantryEnable == 1)
                    EnableGantry()      //Move Axes B to ortholinear position and turn on gantry mode.
                end
            end
endofmain

task: doPTPMotion
  //shortcut
endoftask

function: HomeRigidGantry()
            gantryHomingOn = 0          //Reset trigger
            gantryHomingStat = 0        //Set status to "Homing triggered"

            AGantryOn = 0       //Turn off gantry mode in case it is on
            ALockSrc = 32
            BLockSrc = 31
            ALockEn = 1         //enable Axis A, B lock function to save index value
            BLockEn = 1

            AMotionMode = 0             //Set A Axis to jog mode

            AAccel = 50000      //Set slow and safe motion profile for homing
            ADecel = 50000
            AEmrgDec = 500000
            BAccel = 50000
            BDecel = 50000
            BEmrgDec = 500000

            //AStuckCurr = 1000         //Set lower motor stuck current, to protect against mechanical stress if anything goes wrong
            //AStuckVel = 10000
            //BStuckCurr = 1000
```

```
//BStuckVel = 10000

AMotorOn = 1        //Turn on A Axis
BMotorOn = 0        //Turn off B Axis

gantryHomingStat = 10        //Set status to "Moving to RLS"

ASpeed = -5000
ABegin
while (ALimitsStat != 1 && BLimitsStat != 1)
//while (AMotorCurr > -5000)            //If no limit switch
end
AStop
AWaitStatus[7], 0
AWaitTime, 1000

gantryHomingStat = 20        //Set status to "Moving to FLS"

ASpeed = 5000

ALockCntr = 0        //Set lock counter to 0, to see if index is triggered later
BLockCntr = 0

ABegin
while (ALimitsStat != 2 && BLimitsStat != 2)
//while (AMotorCurr < 5000)            //If no limit switch
end
AStop
AWaitStatus[7], 0
AWaitTime, 1000

if(ALockCntr != 1 || BLockCntr != 1)
        if(ALockCntr == 0 || BLockCntr == 0)            //Sometimes lock is triggered
twice, use this line to bypass
                gantryHomingStat = -2
                return
```

```
                end
        end

        axisALockValSS = ALockVal          //Take snapshot of lockval values
        axisBLockValSS = BLockVal
        ALockEn = 0         //Disable A, B lock function
        BLockEn = 0

        AMotionMode = 1
        BMotionMode = 1
        ASpeed = 5000
        BSpeed = 5000

        ARelTrgt = 0
        AAbsTrgt = axisALockValSS
        ABegin
        gantryHomingStat = 30       //Set status to "Moving to Axis A Index position"
        AWaitStatus[7], 0
        AWaitTime, 1000

        BMotorOn = 1       //Motor on Axis B
        ASetPosition, posAtIndexA
        BSetPosition, posAtIndexA - (BLockVal + indexDiff - BPosRef)
        gantryHomingStat = 100
endoffunc

function: HomeFlexibleGantry()
        gantryHomingOn = 0

        AGantryOn = 0      //Turn off gantry mode in case it is on
        ALockSrc = 32
        BLockSrc = 31
        ALockEn = 1        //enable Axis A, B lock function to save index value
        BLockEn = 1

        AMotionMode = 0 //Set A Axis to jog mode
```

```
BMotionMode = 0             //Set B Axis to jog mode

AAccel = 50000              //Set slow and safe motion profile for homing
ADecel = 50000
AEmrgDec = 500000
BAccel = 50000
BDecel = 50000
BEmrgDec = 500000

//AStuckCurr = 1000                 //Set lower motor stuck current, to protect against mechanical stress if anything goes wrong
//AStuckVel = 10000
//BStuckCurr = 1000
//BStuckVel = 10000

AMotorOn = 1        //Turn on A Axis
BMotorOn = 1        //Turn on B Axis

ASpeed = -5000
BSpeed = -5000
ABegin
BBegin
gantryHomingStat = 10       //Set status to "Moving to RLS"

while (ALimitsStat != 1 && BLimitsStat != 1)            //Check for RLS
//while (AMotorCurr > -3000 && BMotorCurr > -3000)      //If no limit switch
end
AStop
BStop
AWaitStatus[7], 0
BWaitStatus[7], 0
AWaitTime, 1000

ALockCntr = 0       //Set lock counter to 0, to see if index is triggered later
BLockCntr = 0
```

```
ASpeed = 5000
BSpeed = 5000
ABegin
BBegin
gantryHomingStat = 20        //Set status to "Moving to FLS"

while (ALimitsStat != 2 && BLimitsStat != 2)        //Check for FLS
//while (AMotorCurr < 3000 && BMotorCurr < 3000)    //If no limit switch
end
AStop
BStop
AWaitStatus[7], 0
BWaitStatus[7], 0
AWaitTime, 1000

if(ALockCntr != 1 || BLockCntr != 1)
        if(ALockCntr == 0 || BLockCntr == 0)        //Sometimes lock is triggered twice, use this line to bypass
                return
        end
end

axisALockValSS = ALockVal        //Take snapshot of lockval values
axisBLockValSS = BLockVal
ALockEn = 0                      //Disable A, B lock function
BLockEn = 0

AMotionMode = 1
BMotionMode = 1
ASpeed = 5000
BSpeed = 5000
ARelTrgt = 0
AAbsTrgt = axisALockValSS
BRelTrgt =0
BAbsTrgt = axisBLockValSS + indexDiff
ABegin
```

```
        BBegin
        gantryHomingStat = 30        //Set status to "Moving to Axis A Index position"

        AWaitStatus[7], 0
        AWaitTime, 1000

        ASetPosition, posAtIndexA
        BSetPosition, posAtIndexA
        gantryHomingStat = 100
endoffunc

function: EnableGantry()
        if (gantryHomingStat != 100)          //Do not allow gantry on if homing not successful
                return
        end

        gantryEnable = 0    //Disable gantry mode so that axes can be moved independently
        AGantryOn = 0
        AMotorOn = 1        //Ensure that motors are on
        BMotorOn = 1

        AMotionMode = 1             //Move to Axis A index position for gantry on
        ARelTrgt = 0
        AAbsTrgt = APosRef

        BMotionMode = 1
        BRelTrgt = 0
        BAbsTrgt = APosRef

        ABegin
        BBegin

        AWaitStatus[7], 0
        BWaitStatus[7], 0
        AWaitTime, 100
        AGantryOn = 1
```

endoffunc

3.2 puh2 coding

#definevar gantryType{600} w //Set 1 for rigid gantry and 2 for flexible gantry

#definevar posAtIndexA{601} w //The position value to set at index position of Axis A

#definevar indexDiff{602} w //Set this to the relative distance required to move to A index (when at orthorgontal position, at B index position).

#definevar gantryHomingOn{603} w //Set this to 1 to begin gantry homing

#definevar gantryEnable{604} w //Set this to 1 to move axis B such that the axis is ortholinear and then do gantry on.

#definevar gantryHomingStat{620} w //100 - Success

//0 - Homing triggered

//10 - Moving to RLS

//20 - Moving to FLS while searching for lock

//30 - Moving to Axis A Index position

//-1 - Gantry type not defined

//-2 - Could not find index (Axis A index lock not triggered)

#definevar axisALockValSS{630} w

#definevar axisBLockValSS{631} w