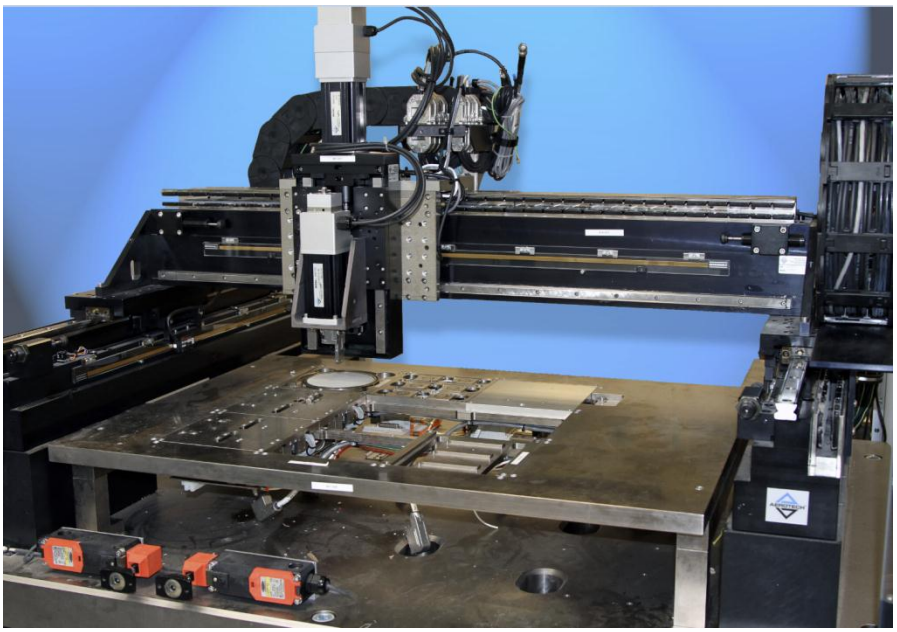
Elmo Application Studio (EAS) Gantry System Tuning Application

www.elmomc.com

Special Tuning Application – Gantry System

1.1.1 Overview



Gantry system

The Gantry system demonstrates control of two axes of which the Master performs two separate synchronized control operations; the Master axis locates and resolves the center point of the bridge and resolves the differential position between small anomalous movements of the bridge. To obtain optimum performance, all the PWMs must be synchronized.

The Gantry is configured such that the pivot controller is defined as the Master relative to the bridge end controller which is defined as the Slave. The movement of the Slave is therefore dependent on the Master. The X_1 Master therefore controls:

- $X_{center} = (X_1+X_2) / 2$
- $\theta = (X_1-X_2)$
- X_1 Master in Position mode send current command to Slave

Whereas the X_2 Slave controller implements the Forcer/Driver X_2 Current Loop, and sends the X_2 position information to X_1 Master.

In addition, this algorithm supports a planar motor where the differential adjustment is performed to the Y axis, together with the commutation angular correction.

X_1/X_2 have their own commutation feedbacks, and the Y_1/Y_2 commutation angle is set by the Y_1 sensor, and corrected by the θ phased shift.

Make sure that the hardware cabling is installed as per the Gantry Cabling Application Note.

1.1.2 Method Overview

After connecting the hardware according to the Gantry Cabling Application Note, perform the following steps to define and tune a gantry with Elmo Gold drives:

1. Setup the software.
2. Set up the communication between PC and the servo amplifiers.
3. Configure the **Slave** to include Commutation.
4. Copy the parameters from the slave to the master axis. This ensures that the basic operational parameters are identical.
5. The advanced configuration of the master and slave is continued with the position setting of the gantry controllers. These settings differ between the master and slave.
6. Tune the Yaw controller.
7. Perform error mapping and test the Gantry motion.
8. Raise the power limits to operational levels, tune for higher power levels and retest the Gantry motion.
9. The gantry is now tuned and ready for normal operation.

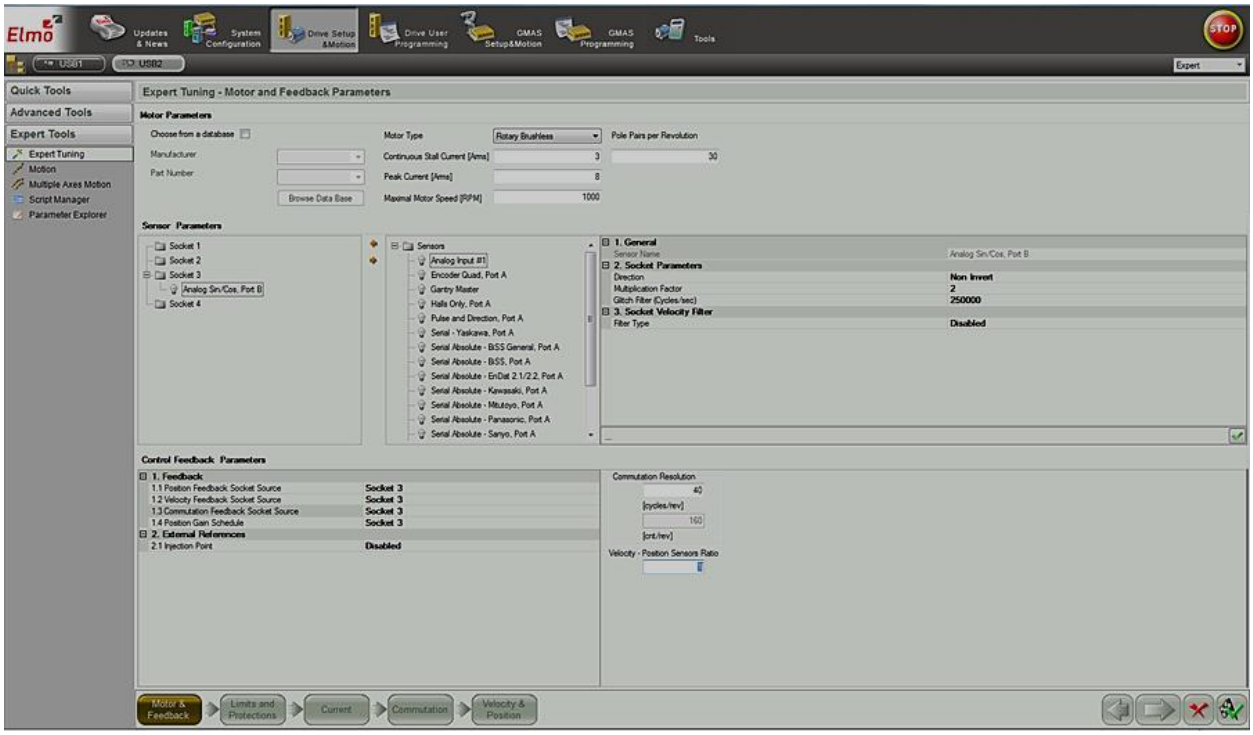
1.1.3 Setup and Slave Tuning

To setup and tune the Gantry Slave:

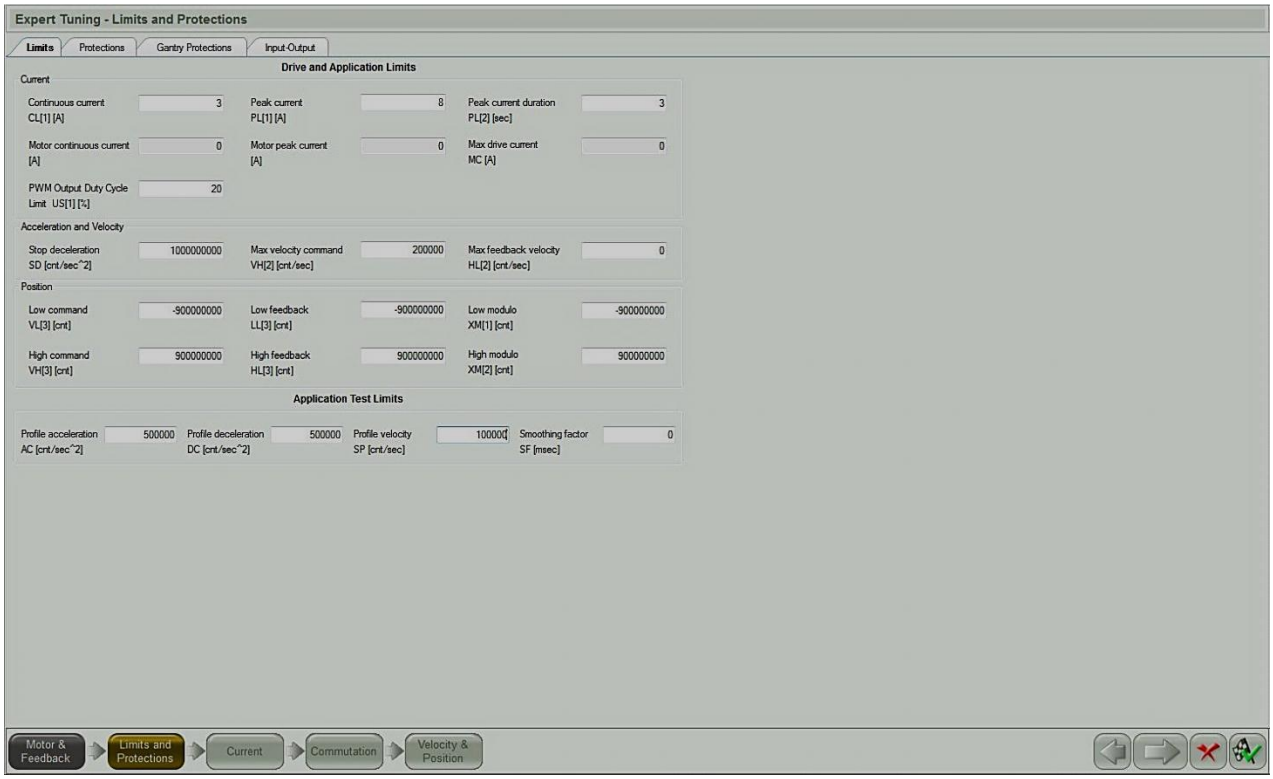
1. Verify that the EAS version is 1.2.0.2 or higher. Download the drive firmware version file NGDrive01.01.04.8317June2012B02G.gabs or higher (check for the latest released version), and upgrade the drives' firmware.
   Verify that the drives are GCON Rev C. This is obligatory, and can be verified by typing the command WS[8], the value returned should be 2.
2. Connect the Gantry Master and Slave to the PC via USB, Ethernet or G-MAS connection. For the purpose of this procedure, the connection is assumed to be USB.
3. In the System Configuration window, **connect** the Master and Slave controllers.
4. From the top row menu, select the Tools icon, and open the terminal.
5. Verify that all drives in the gantry have similar values to the following:
   a. Maximum current (MC)
   b. Sampling time TS. Note that for GTRO/DRUM products TS≥60
   c. **XP[1]**
   d. **XP[2]**
6. Make sure the Slave **USB** (or Ethernet Slave) is selected.
7. From the **Drive Setup and Motion** window, run the Expert Tuning wizard starting at **Motor & Feedback**.

Note: **All captures shown below are for illustration purposes only, unless specifically stated otherwise.**

8. Setup the Expert Tuning – Motor Feedback Parameters, as shown in the screen below, with the following conditions:
   - Set the Motor Parameters with the Linear Brushless motor type, and settings as per motor specifications.
   - In the Sensor Parameters, use Socket 3 with the Analog Sin/Cos, Port B sensor. Where the sensor is a Digital Quad in port A or B, set accordingly.
   - Set the feedback parameters as per specifications.
   - Similarly, adjust the Control Feedback Parameters with the Position, Velocity, and Commutation sources set to Socket 3.
   - Set the Commutation Resolution according to the encoder specifications.



9. Click **Limits and Protection** in the Tuning wizard.
10. Set the Limits tab according to the application specifications. For the initial tuning, set the application current limits to 50% of the operating CL/PL.



11. In the PWM duty cycle field box in the diagram above, enter the value of **US[1]** (PWM duty cycle) to 20%. Lowering the PWM duty cycle to a minimum for the tuning

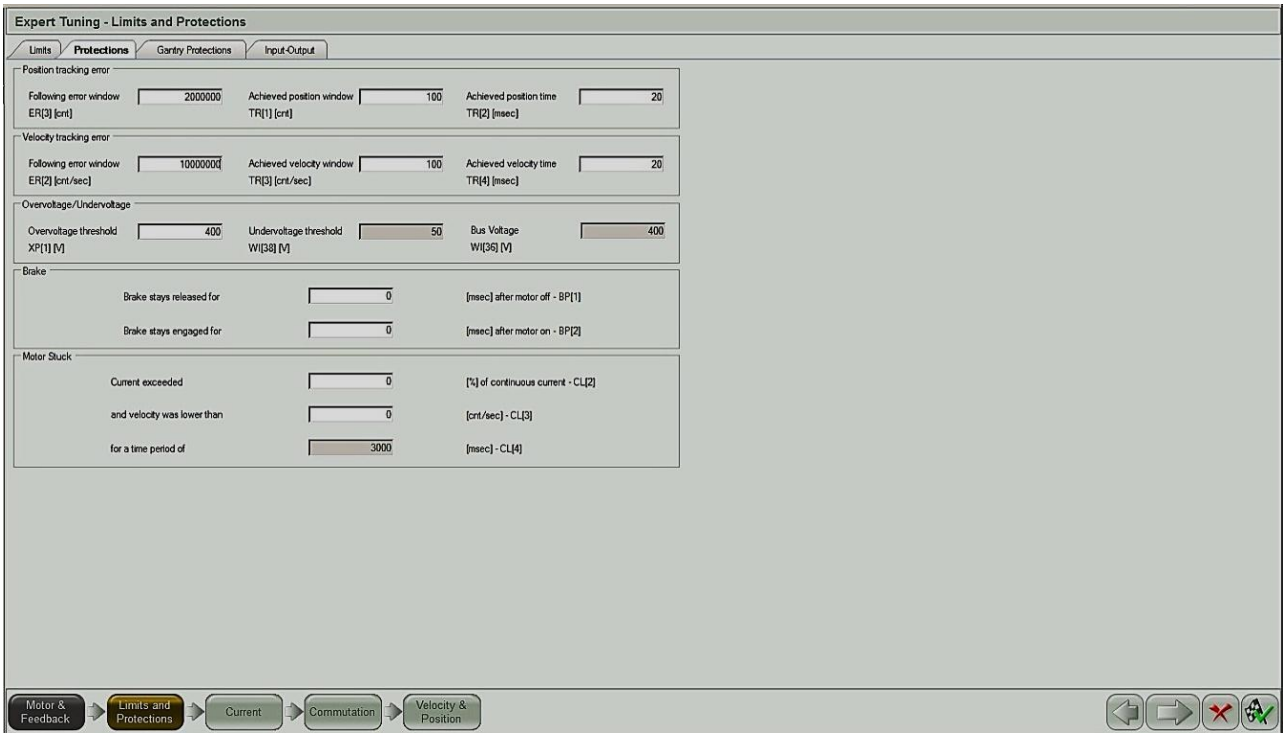
procedure prevents uncontrolled motion at high power by limiting the motor speed & BEMF.

12. Set the Protections tab, as shown in the example below:

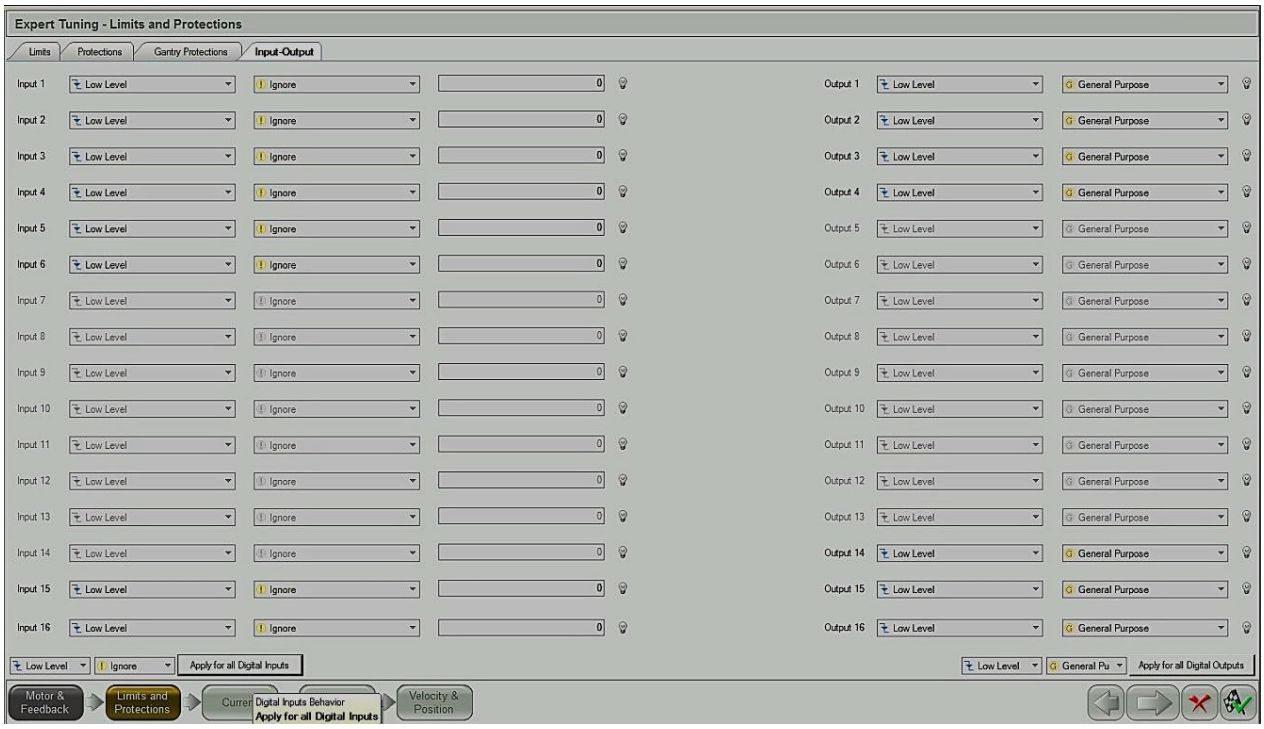
    a. Limit **ER[2]** to 20% of maximal velocity

    b. Limit **ER[3]** to several cm at most

13. Define the "motor stuck" event. This is an important step.

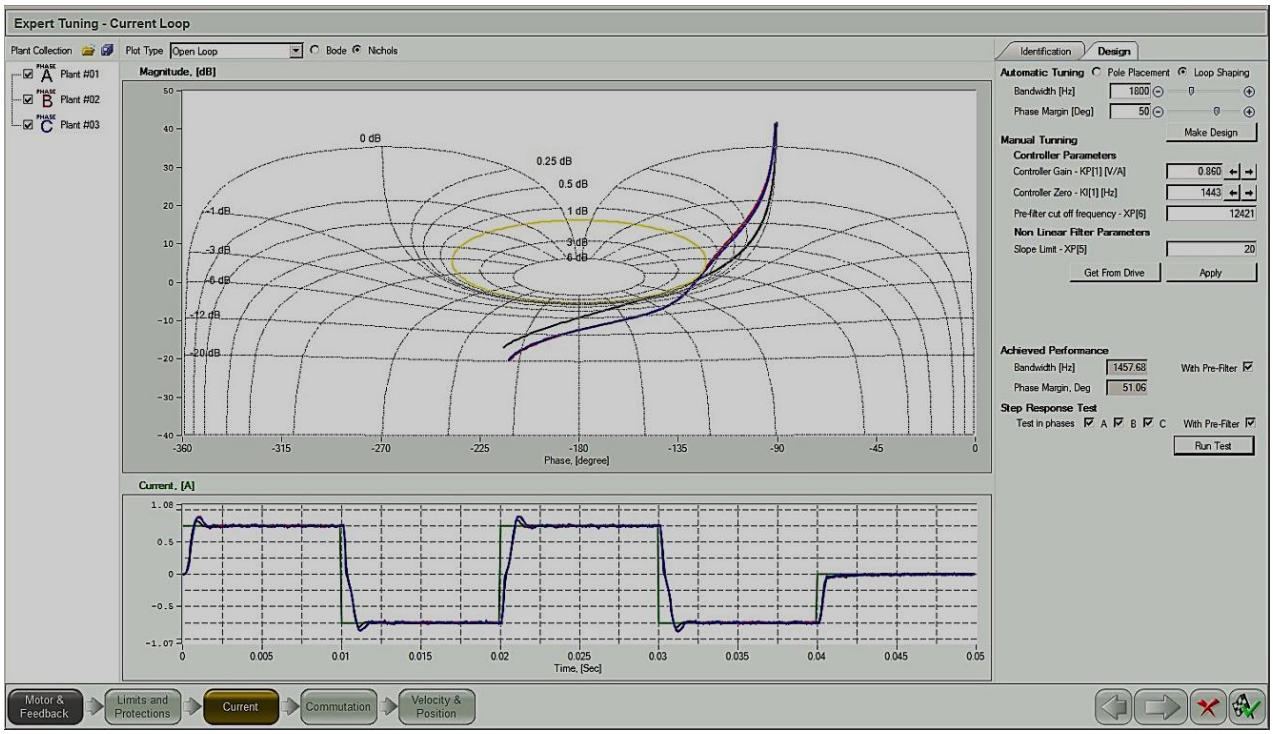


14. Set the Input Output according to the application. If there are none, set the Inputs to **Ignore** as shown in the example below.



15. Click **Current** in the Tuning wizard.
16. In the Current window, Identification tab, click **Run Experiment**. The Bode graph is displayed. If necessary, adjust the Maximal or Minimal Current Level and repeat the experiment.

17. Click the design tab and click **Make Design**. The Nichols graph is drawn and the Plot Type appears as **Open Loop**.



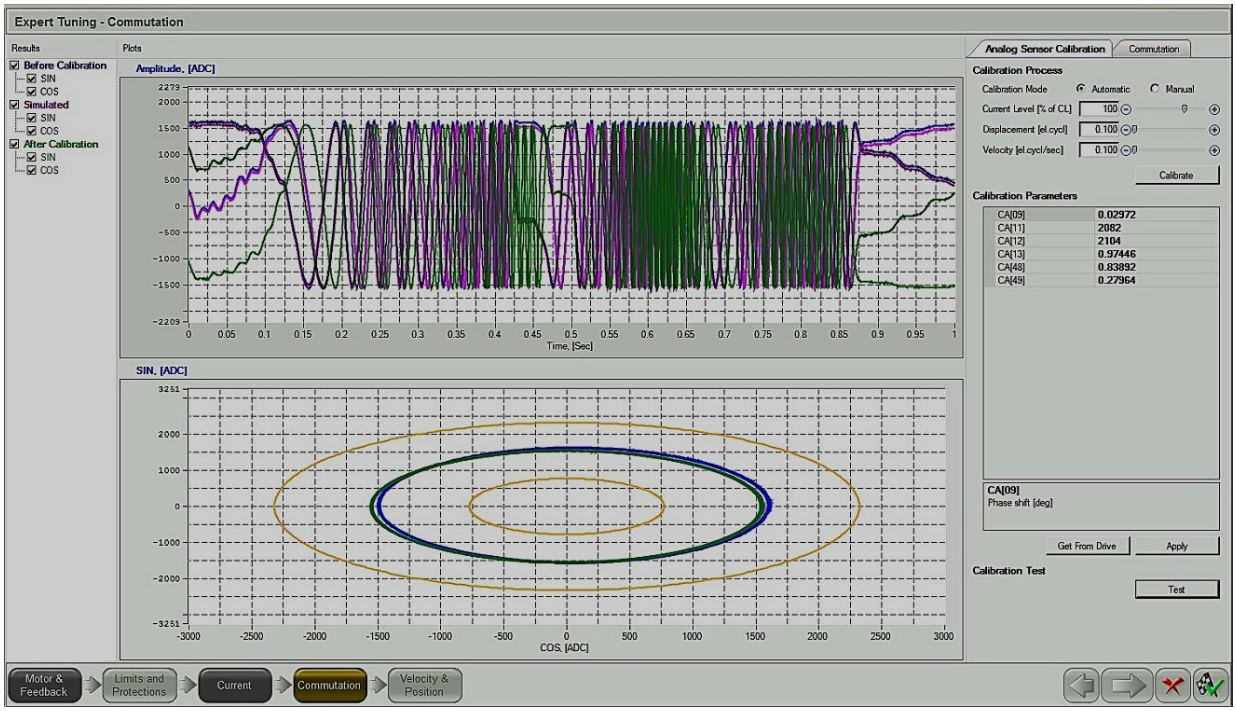
18. When satisfied with the Nichols graphs, click **Run Test** to check the performance. The lower graphs shown above are displayed.
19. Click **Commutation** in the Tuning wizard.
20. For an **Analog Sensor,** perform **Calibration**.

    Since the gantry feedback sensor are usually exceptionally sensitive, run the experiment with minimum displacement and velocity. The Calibration Mode is set to Automatic as default:

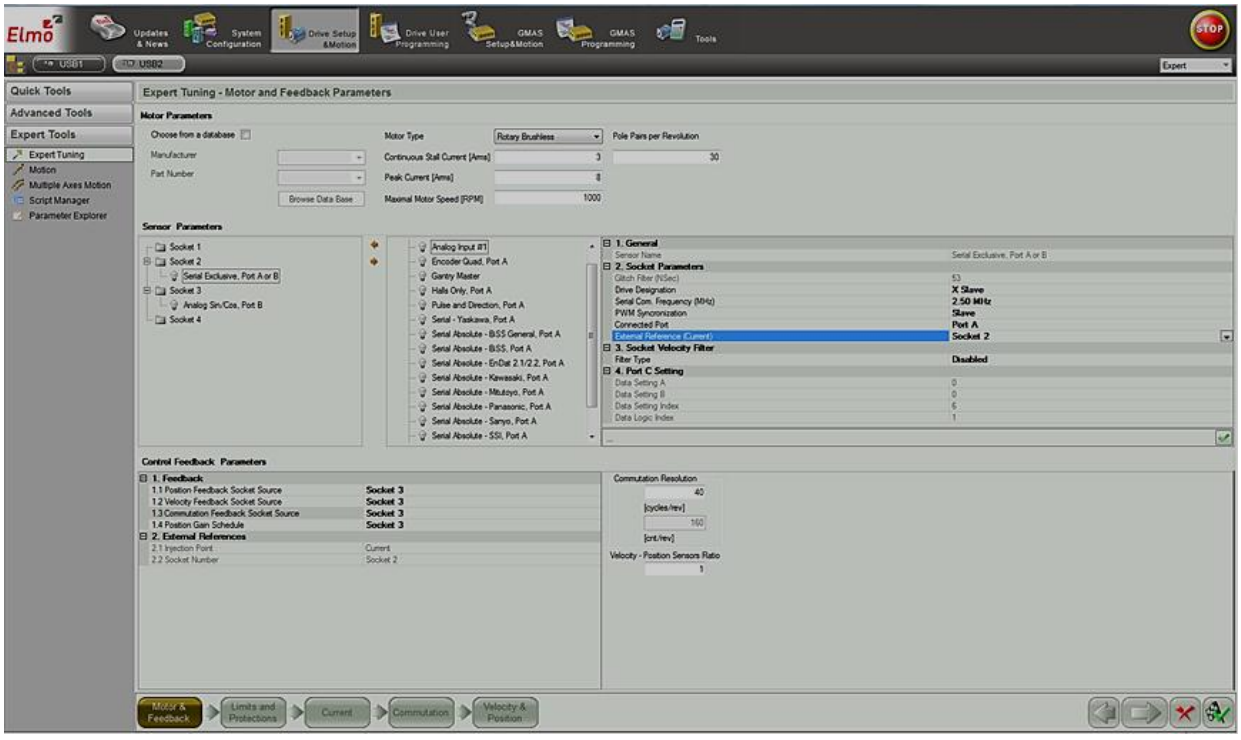
    a. In the Analog Sensor Calibration tab, click **Calibrate**. Two graphs are produced, of which the lower SIN [ADC] graph should appear similar to the shape of a circle, or ellipse.

    Both graphs display the sine/cosine before and after calibration. The post-calibration lower graph should appear as a circle or ellipse.

    b. At Calibration Test, click **Test**. The following result is displayed which should follow the pattern of the original graphs as close as possible.



21. Click the Commutation tab. Select the Commutation method and click **Run Experiment**.
22. When the Gantry bridge moves, notice the direction of movement. Define this direction as positive or negative.
    Pay attention to the direction of the master and slave in relation to the system as this must be uniformly and consistently defined.



23. Save the drive status by clicking  or at the terminal, use the SV command.
24. Return to System Configuration screen, download and save the parameters.
    These parameters will be later uploaded to the master drive.
25. Click **Motor & Feedback** to set up the **Serial Exclusive** at **Socket 2** for the **Slave:**

    c. Attach the Serial Exclusive sensor to **Socket 2**.

    d. Setup the Serial Exclusive sensor as shown in the screen capture below. Make sure that the Drive Designation is set to **X Slave**, and the PWM Synchronization set to **Slave**.

    e. Set the **External Reference Current** to **Socket 2**.

    f. The **Control Feedback Parameters** should all be set to **Socket 3**.

    The external references should remain grayed out.

1. General	
1.01 Sensor Name	Serial Exclusive, Port A or B
2. Socket Parameters	
2.01 Glitch Filter (NSec)	53
2.02 Drive Designation	**X Slave**
2.03 Serial Com. Frequency (MHz)	**5.00 MHz**
2.04 PWM Syncronization	**Slave**
2.05 Connected Port	**Port A**
2.06 External Reference (Current)	**Socket 2**
2.07 Index Data Settings A	0
2.08 Index Data Settings B	0
2.09 Index Data Function	6
2.10 Index Data Logic	1

Socket Position: 0

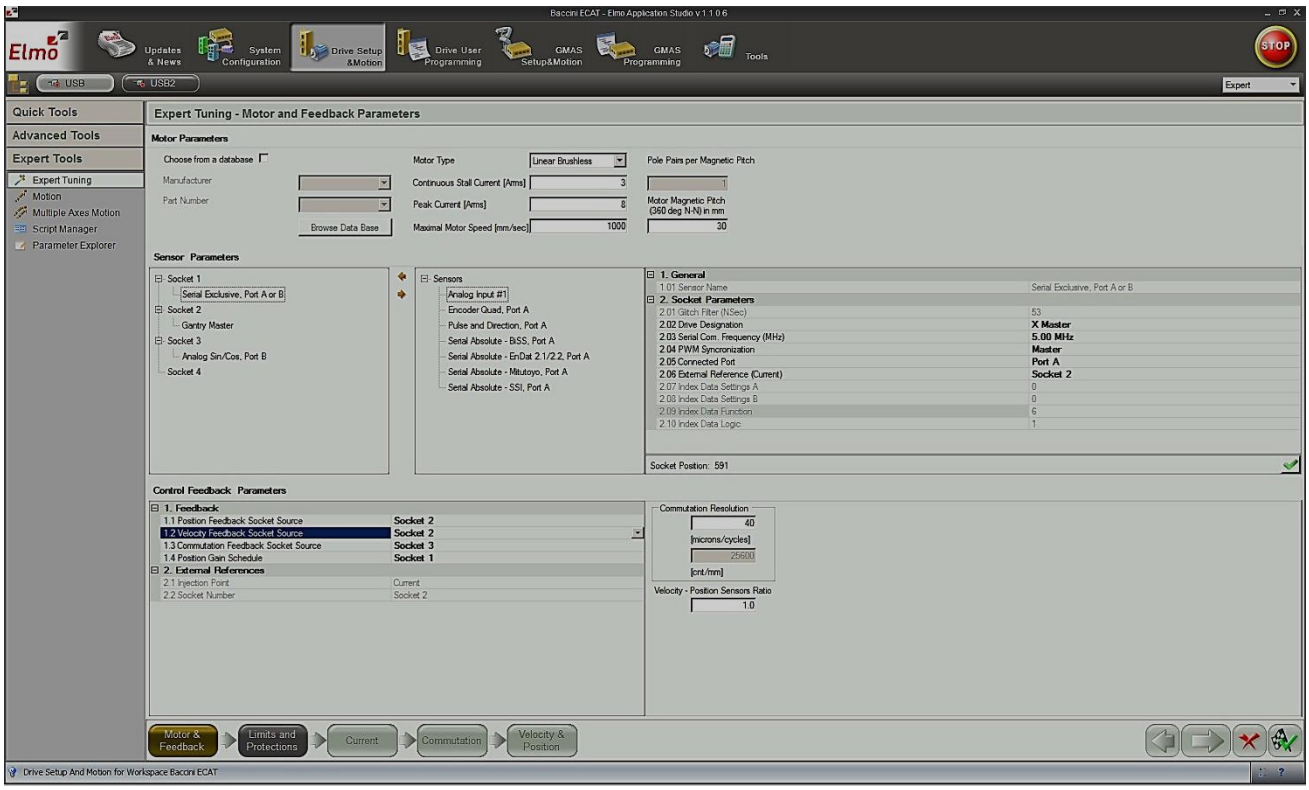
The Slave Socket parameters for the Serial Exclusive are defined according to the table:

Drive Designation	X Slave
Serial Communication Frequency	5.00 MHz (default value)
PWM Synchronization	Slave
Connected Port	Port A (Port B here is used for a feedback sensor)
External Reference (Current)	Socket 2. The current originates from the Master via this socket.

1.1.4 Master Setup and Tuning

To setup and tune the Gantry Master:

1. Click **System Configuration** to select the Master drive.
2. Upload the Slave's parameters saved previously, into the Master drive.
3. Select the Master **USB** (or Ethernet Master).
4. In the Tuning wizard, click **Motor & Feedback**.
5. Check the following:
   a. General motor parameters
   b. Limits & Protection
   c. Input/Output
   d. Verify that the settings for the Master are the same values as described in section 1.1.3 steps 5 - 10 above for the Slave.
   e. If a **digital quad encoder** is used, type the command **GS[3]**=[maximum speed]in the Command Terminal.
   Add a FIR filter to cancel the 1/T velocity calculation.
6. Click **Current** in the Tuning wizard.
7. In the **Identification** tab, click **Run Experiment** to test the motor current response.
8. Click the **Design** tab, and select **Run Test** to check step response. *However, do not redesign the current loop.* If in either tab a significant difference exists between the Master and the Slave, investigate. The Master and Slave should be identical.
9. Switch to **Commutation** in the Tuning wizard, and proceed with commutation for the Master drive as per section 1.1.3 steps 20 - 22 above using the same commutation method as for the Slave.



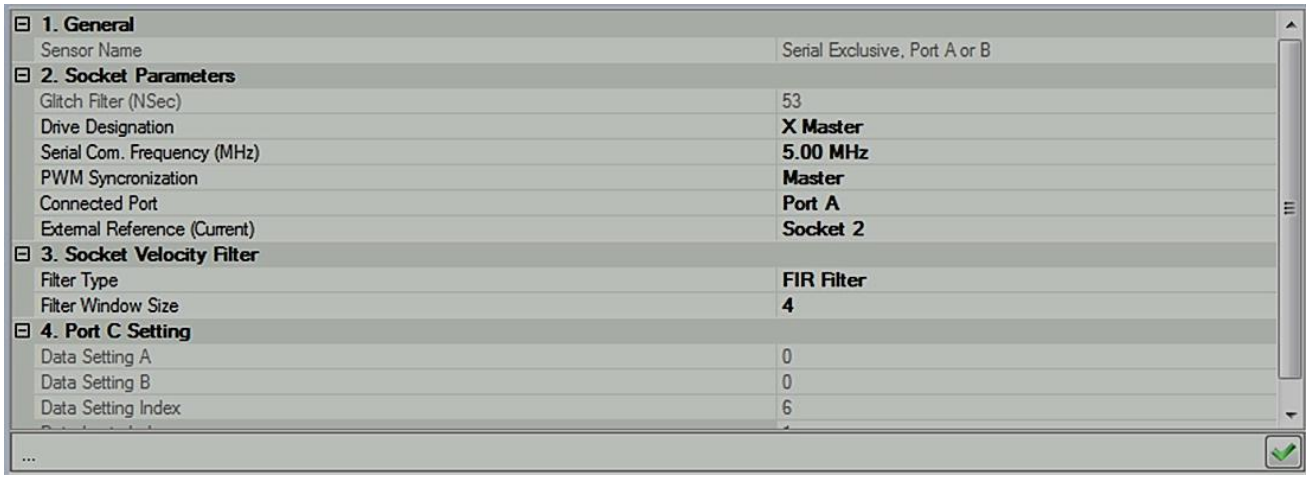
10. Click **Motor & Feedback** to set up the **Serial Exclusive** at **Socket 1**, and the **Gantry Master** at **Socket 2** for the **Master** at **USB (or Ethernet)**:

    a. Make sure that the order of the sensors at the Sockets is exactly as shown in the screen capture below.

    b. Setup the Serial Exclusive sensor as shown in the screen capture below. Make sure that the Drive Designation is set to **X Master**, and the PWM Synchronization set to **Master**.

    c. Set the External Reference Current to **Socket 2**.

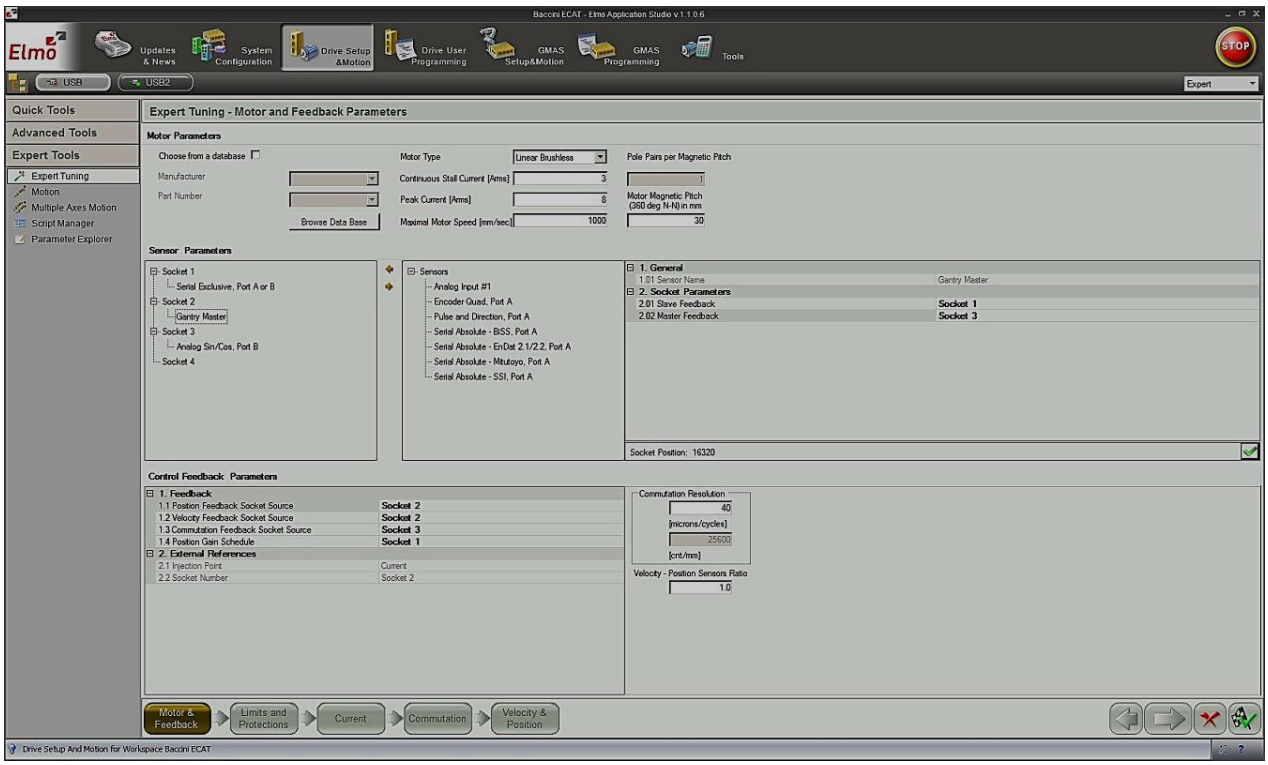
    d. The Control Feedback Parameters should be set as shown in the screen capture below.



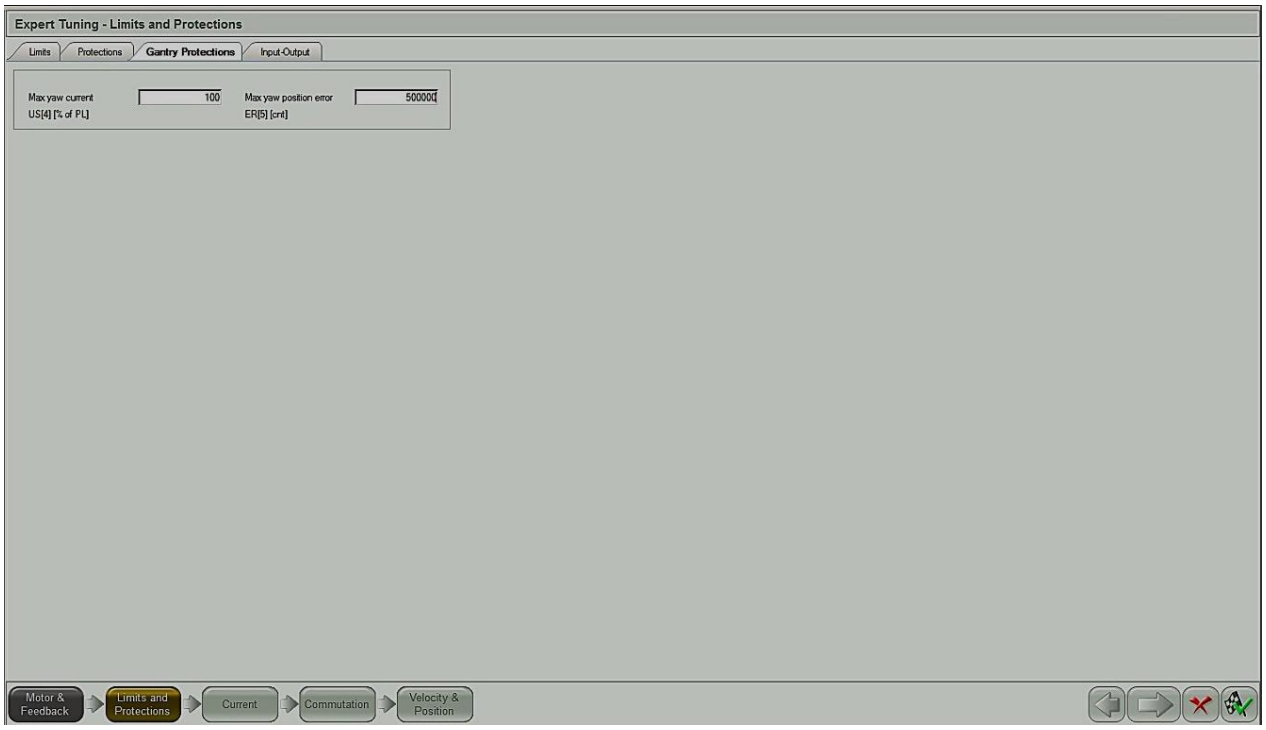
The Master Socket parameters for the Serial Exclusive are defined according to the table:

Drive Designation	X Master
Serial Communication Frequency	5.00 MHz (default value)
PWM Synchronization	Master
Connected Port	Port A (Port B here is used for a feedback sensor)
External Reference (Current)	Socket 2. The current is calculated from the Gantry socket.
Socket Velocity Filter	FIR filter

11. At the Sensor Parameters, highlight the **Gantry Master** connected to **Socket 2**.
12. Set the Socket Parameters as shown in the screen capture below. Make sure that:
    a. The Slave Feedback is set to **Socket 1**.
    b. The Master Feedback is set to **Socket 3**.
    c. The Control Feedback Parameters are set as shown in the capture, with the external references preset to **Socket 2**.
    d. Set the Position and Velocity Feedbacks to **Socket 2**.
    e. Maintain the commutation feedback socket source selected to **Socket 3**.
13. Set a FIR filter at **Master Socket 1**. Make sure that the FIR filter window size is identical to that, at **Master Socket 3**.
14. Check that the motors are disabled.
15. Manually, move the gantry and check the encoder readings for all three sockets. *Check that the Slave and Master are defined in the same direction.*



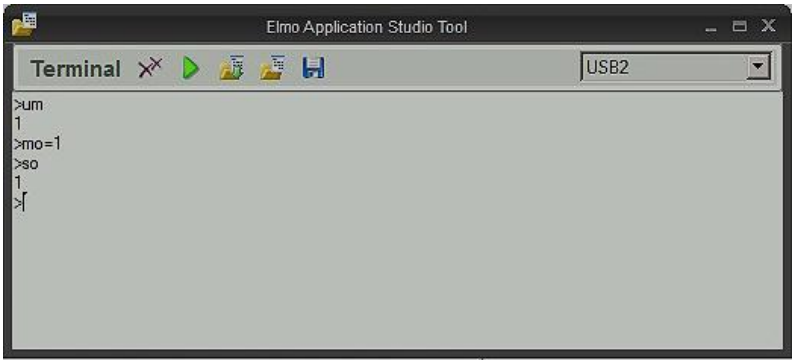
16. Click **Limits and Protections** in the wizard. The **Gantry Protections** tab should now be accessible.
17. Set the **Gantry Protections** according to the application requirements, similar to that shown in the screen capture below.



ER[5] – This Command parameter sets the maximum value for differential position. Limit the value of **ER[5]** to several mm. When the absolute value of the differential position is above this parameter, all the Gantry drives are disabled.

18. At the top of the EAS menu click 

. The EAS Tool Terminal opens.
19. Enter the commands shown in the screen capture to operate the **Slave** drive (**USB2**). Note that if the Slave drive is not enabled, the Master drive cannot be enabled. This is a safety feature, because once the gantry controls have been connected, the Master depends on inputs from the Slave, and without them, motion is not controlled.



20. Close the Terminal.

Once the appropriate software switches are enabled, there is a link between the drives. The yaw control loop is active and the velocity/position loops will be tuned for the plant.

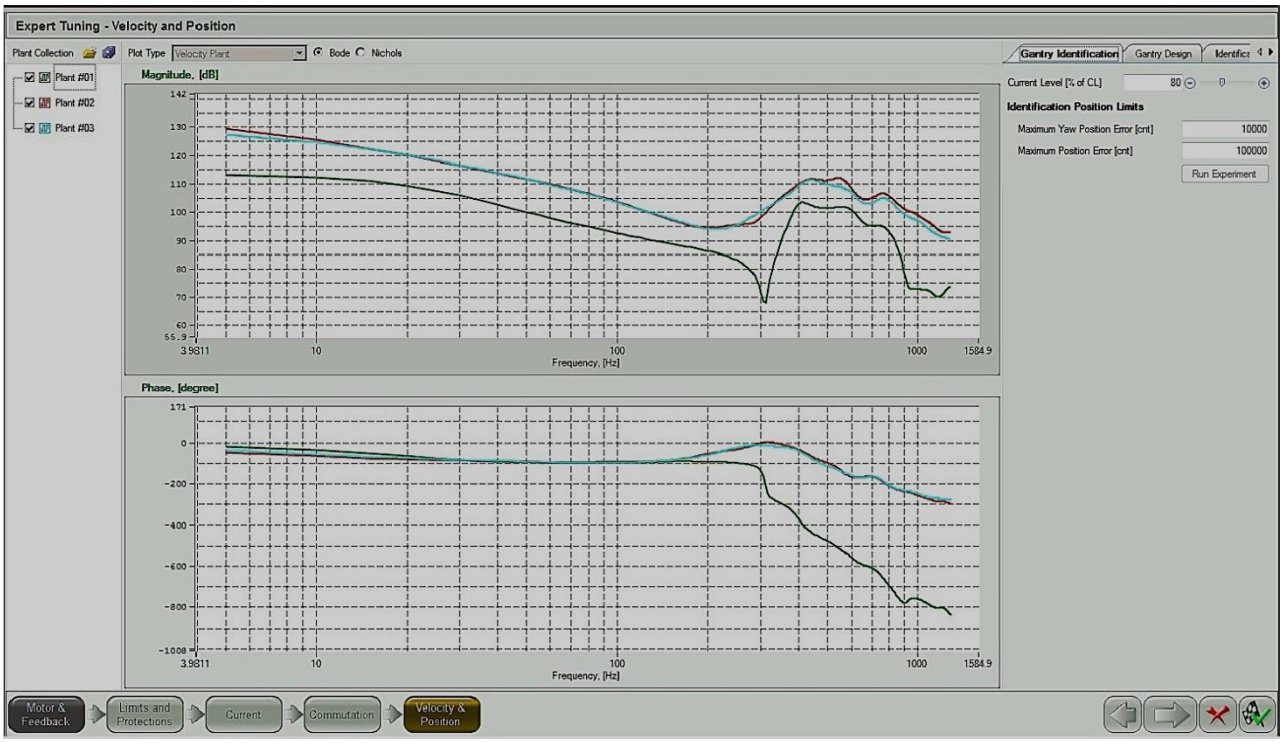
1.1.5 Configuring the Gantry Controller

To configure the Gantry Controller:

1. In the Tuning wizard, click **Velocity & Position**.

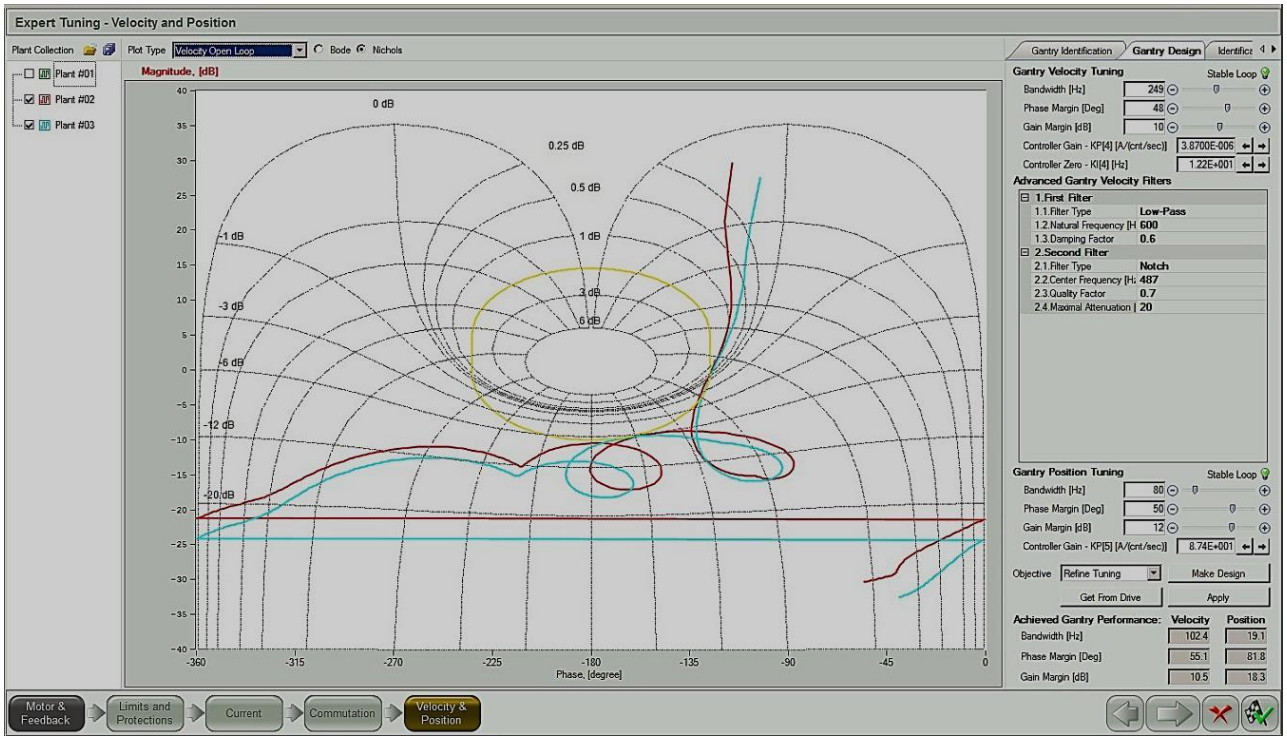
Note: **EAS is automatically configured to set FP[1]=0, FP[3]=0, and PX =0 when starting the identification. This is necessary, because a result of a position difference will immediately cause the yaw control to try to correct it, which may produce uncontrolled motion.**
While the Slave drive is enabled in Current mode, all Velocity & Position tests are performed from the Master viewpoint.

2. In the **Gantry Identification** tab, verify that the safety limits (maximum yaw position error and maximum position error) are acceptable. The appropriate proper values for both should be several cm. If uncontrolled motion occurs, the gantry drives will shut off when these limits are reached.
3. When the safety limits have been set, click **Run Experiment**. The Magnitude and Phase graphs are drawn. Select other values of the Current Level to produce converging graphs. When satisfied proceed to the next step.



4. Click the **Gantry Design** tab. Select the following:
   a. Plot type should be set to **Velocity Open Loop**.
   b. Select which Plants are suitable for the design in the Gantry Design window.
   c. Select from the Objective design options, and choose Advanced Gantry Velocity Filters, depending on the Objective design options selected. For

further details, refer to the section **Error! Reference source not found. Error! Reference source not found. Error! Bookmark not defined.**.



5. Click **Make Design**. The above Nichols graph or similar is produced for the various Plants selected. In addition, the Achieved Gantry Performance statistics are shown in the lower right column of the window.

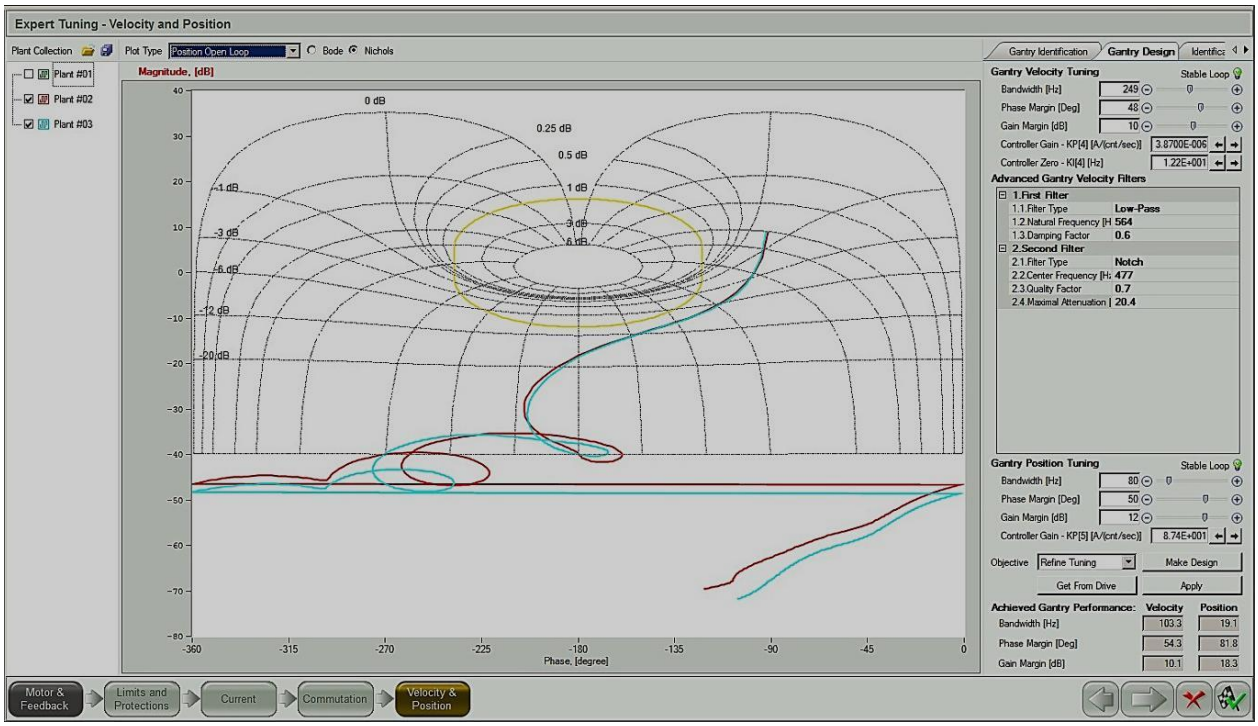
6. Select the following:

   a. Plot type should be set to **Position Open Loop**.

   b. Set the Gantry Velocity Tuning as shown in the screen capture below, the same settings as for the Velocity Open Loop.

   c. Set the Gantry Position Tuning as shown in the screen capture below, the same settings as for the Velocity Open Loop.

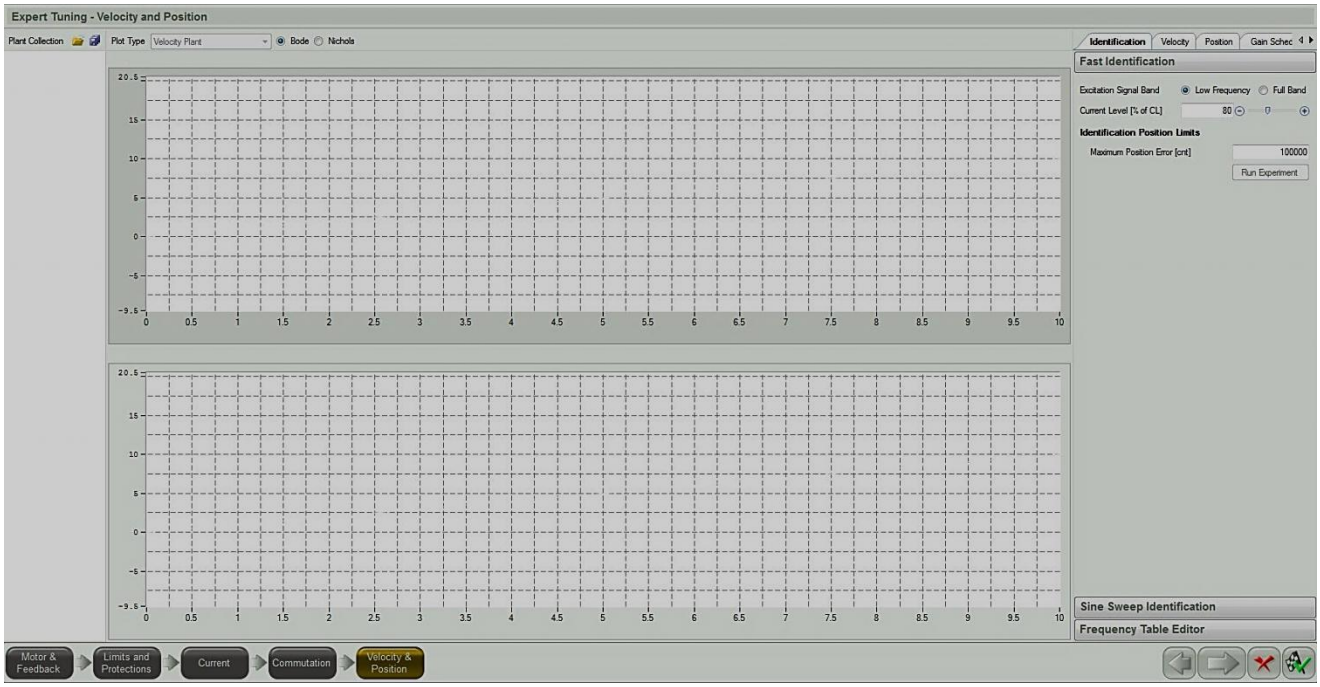
   d. Set the Advanced Gantry Velocity Filters as shown in the screen capture below.



7. Click **Make Design**. The above Nichols graph or similar is produced for the various Plants selected. In addition, the Achieved Gantry Performance statistics are shown in the lower right column of the window.

Note: **Make sure that the Slave is switched to ON.**

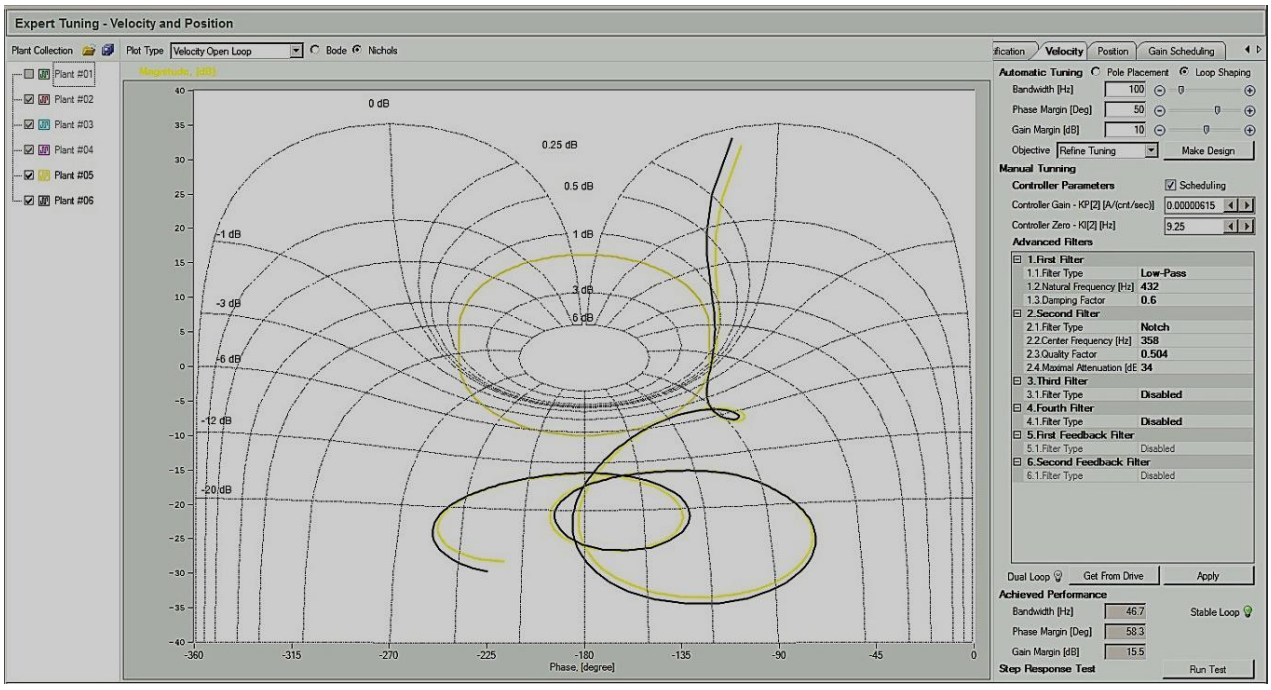
8. In the **Identification** tab, check that the safety limit (maximum position error) is acceptable. The appropriate proper values are several cm. If uncontrolled motion occurs, the gantry drives will shut off if the gantry motor moves farther than this limit allows.
9. Click **Run Experiment**. *The Sine sweep cannot be used with the Gantry.*



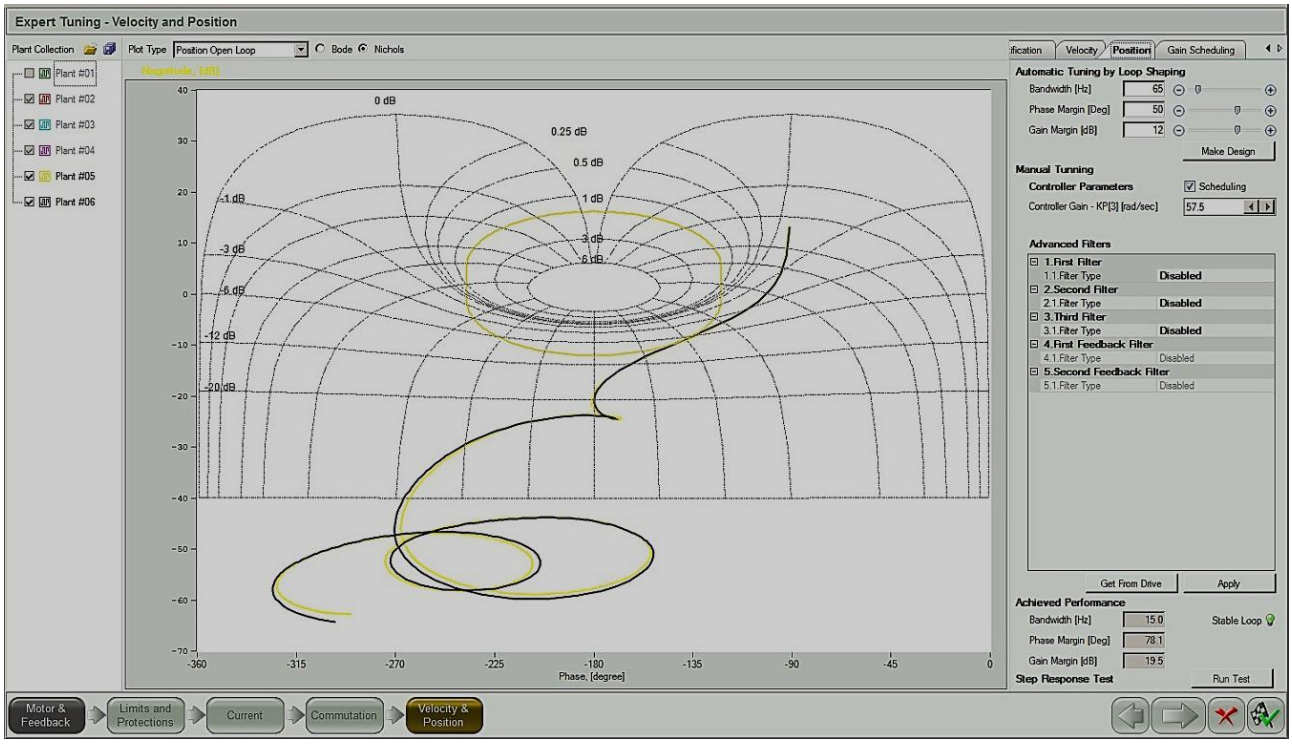
10. Click **Run Experiment**. *The Sine sweep cannot be used with the Gantry.*
11. Click the **Velocity** tab and set the following:
    a. Automatic Tuning parameters required by the system.
    b. Advanced Filter parameters as necessary depending on the Objective design options selected.

12. Click **Make Design**. The Bode graph is drawn and the Achieved Performance statistics are displayed as shown above.

    Notice that a new set of Plant #0X graphs are displayed in the Plant Collection. The Gantry Plants are grayed and cannot be changed.



13. Select **Position Open Loop** for the Plot Type.
14. Click **Make Design** to produce the Nichols graph for the two Plants selected by the Position Open Loop.



15. The Nichols graph is drawn as shown above. If satisfied with the results, save the configuration.

1.1.6 Error Mapping (Correction)

Most of Gantry systems are rigid, and if the sensor and mechanics are not perfectly aligned, the gantry (Yaw) control will continuously cause bridge misalignment, countered by excessive current in the system to force rearrangement of the bridge. To prevent this situation occurring, Elmo recommends using error mapping to correct the misalignment between the Master and the Slave. The following procedure explains how to perform the error correction in the gantry system.

To utilize Error Mapping on the Gantry it is necessary to perform the following:

1. Cancel the operation of the Yaw Control using the command **US[4]=0**.
2. Set **FP[1] = 0**; **FP[3] = 0**; **PX = 0**
3. Set **UM=5** to enable master position mode.
4. Perform Homing:
    a. Set command **OV[54]=3**. This will redirect the index signal so that all three sockets are homed with the Master's actual feedback.
    b. Reset positions by entering the commands **FP[1]=0**; **FP[3]=0**; and **PX=0**.
    c. Perform DS402 homing on the gantry Master using index or index+limits.

d. If a G-MAS is connected to the drives, use the G-MAS to perform Homing.

5. Perform Error Mapping:

   a. Run the error mapping procedure (refer to section 1.1.6.2 Code for Error Mapping Procedure). It may be adapted and used on any gantry. Please note that it is programmed for a gantry with an index signal **and** limit sensors.

   b. The errors are mapped in relation to **Socket 3**. The value for each index should be the gantry error (stored in **WS[35]**)

6. Perform Motion testing:

   a. Reestablish yaw control using the command **US[4]=100**.

   b. To test and tune the yaw controller use the command **TW[14]=<step size>**.
   This will provide the yaw controller with a step command, which can be monitored on the Recorder and corrected.

   c. Set the step to zero again by issuing the command **TW[14]=0**.

7. After the above steps have been completed, the following steps can be performed to tune the gantry yaw control:

   a. Home the system.

   b. Enable error mapping by running the error mapping procedure.

   c. Run the system at low speed over the entire span and:

i. Check the Currents.
If there are high currents in one direction and low currents in the other direction, or if currents are generally high, examine the error mapping procedure. The procedure is not functioning correctly.

ii. Make sure that the yaw control is small (~0) in both directions, forward and back.

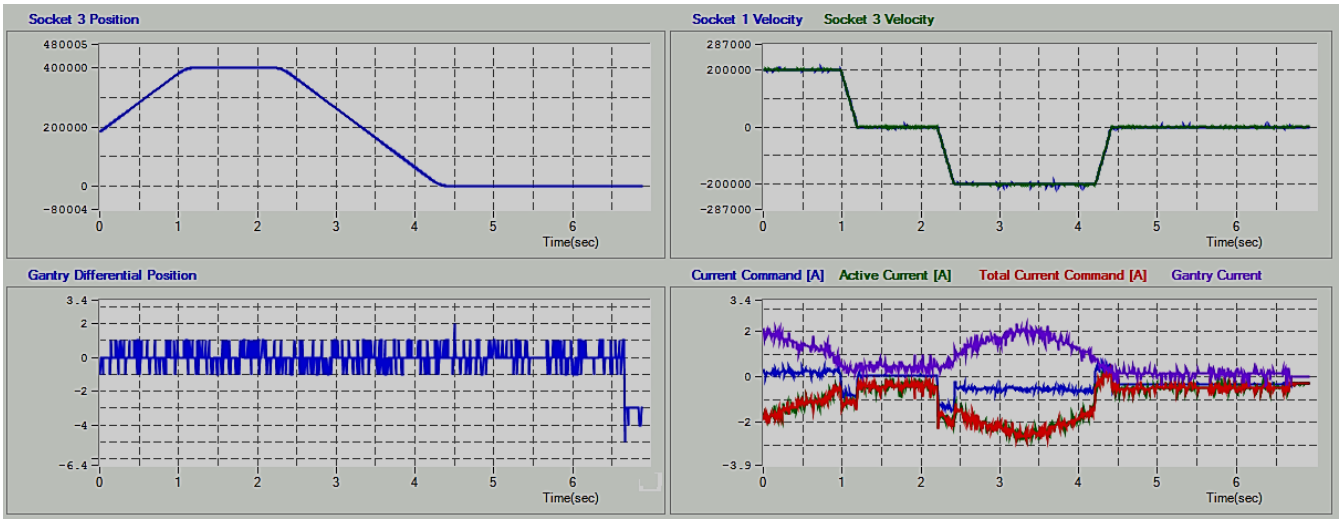
   d. When the system is running smoothly, the current limits and PWM may be returned to their operational values (but never exceed maximum).

   e. When both homing **and** error mapping have been proofed, both loops can be tuned for optimal performance with the operational current levels, voltage and gains.

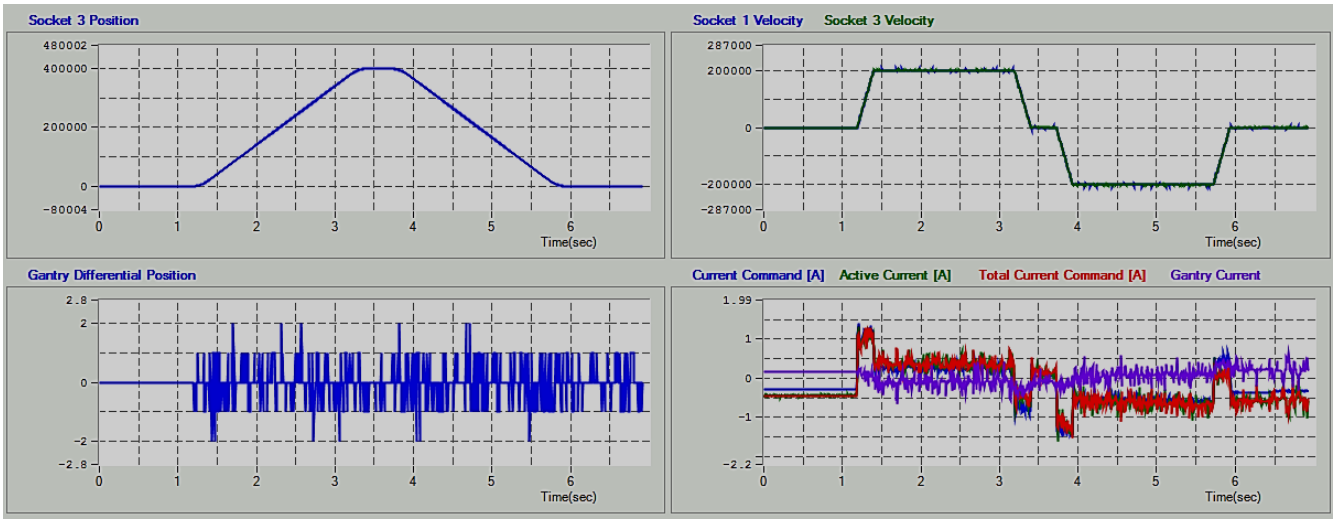
For further details refer to the Chapter Error Correction in the Gold Drive Administration Manual.

1.1.6.1 Error Mapping Results Example

1. To compare performance before and after error mapping, use the recorder to record the relevant parameters as seen in the figure below:



2. In the figure above it is evident, by the high currents during simple PTP motion, that the gantry yaw controller is (needlessly) applying great force to keep the gantry moving straight. Although the differential position error is small, the controller is over-straining to keep it in check.
3. Although in the example presented the differential errors are small before and after error mapping. After successfully applying the error correction procedure, the currents for correcting the differential errors are greatly diminished. The mechanical imperfections are corrected by the control with reduced effort.



1.1.6.2 Code for Error Mapping Procedure

```
##start
#define SPEED                   100000
#define HOMING_SPEED            40000
#define GRID_FACTOR             14
#define GRID                    16384

main()
function main()

       us[4]=0              // Disable yaw control before homing
       pc[1]=0              // Disable error mapping before starting procedure

       home()               // Function for homing gantry
       map()                // Function to map the gantry mechanical errors

       pc[1]=1              // Command to arm error mapping
       us[4]=100            // Start yaw control

       // Run test
       wait(1000)
       pa=vh[3] ; bg ;
       until(ms!=2)
       pa=0 ; bg ;

return

// ---------------------------------------------------------------
//  Function map()
// - Map the gantry differential error on a defined grid.
// - For each grid point, the magnitude of the gantry differential
//   error equals (X2 position) - (X1 position)
// - The function measures the differential error twice,
//   first time while running in the positive direction, and
//   second time in the negative direction.
// - The final error mapping value is the average value of the
//   two measurements.
// ---------------------------------------------------------------
function map()
       int i

                    // Power on (if not powered)
                    // NOTE: Slave axis must be on to power the master axis.
       if (so!=1)
              mo=1
       end if
       until (so==1)

  // Set up PC[] array
       pc[3]=2                          // Command for using ET[n] table
       pc[2]=3                                                        //
Command for error mapping on socket 3
       pc[6]=GRID_FACTOR                // Sets the grid to 2^GRID_FACTOR
       pc[4]=1                          // Low index of correction table
       pc[5]=(vh[3]/GRID)+1             // High index of correction table
       pc[7]=0                          // Starting position on grid

       // Loop1: Go over each grid point, and measure the differential error
       sp=SPEED
       gp[1]=0                  // The first error mapping point must be position 0
       for i=2:1:pc[5]          // The loop begins from the second point
              pa=(i-1)*GRID     // Command to go to grid point [i-1]
              bg
              until ((ms!=2) || (sr&0x10000000)) // Wait for end of motion (or limit)
              wait(700)         // Wait 700mS to stabilize
```

```
              gp[i]=ws[35]          // Set error mapping value for point [i]
                                    //     (ws[35] stores the (X2pos)-(X1pos) )
       end

// Go to software limit
       pa=vh[3]
       bg
       until (ms!=2)

       // Loop2: go over each grid point in the reverse direction, and measure
the
       // differential error. For each point, calculate the average value between
       // loop1 reading and this reading
       for i=pc[5]:-1:2
              pa=(i-1)*GRID         // Go to grid point [i]
              bg
              until ((ms!=2) || (sr&0x10000000))// Wait for end of motion (or limit)
              wait(700)
              gp[i]=(gp[i]+ws[35])/2     // Error mapping value[i] = the average
between
                                    //     this and the previous reading
       end
       gp[(pc[5])]=0                // Set the last (virtual) point to 0

return

// -------------------------------------------------------------
//  Function home()
// - Open software limits
// - Home on RLS using HF[] commands
// - Home on Index using HF[] commands
// - Capture FLS and set software limits
// -------------------------------------------------------------
function home()

       // Disable the drive
       mo=0

       // Open the software limits to max
       vh[3]=hl[3]
       vl[3]=ll[3]

       // Reset socket positions
       fp[1]=0;fp[3]=0

       // Configure HF[] array for RLS capturing
       hf[10]=3                     // Set socket 3
       hf[2]=0                      // Absolute value setting
       hf[5]=3                      // When the event occurs, set PX=HM[2]
       hf[3]=7                      // Event on RLS
       hf[4]=0                      // Arm the homing function

       // Power on (if not powered).
       // NOTE: Slave axis must be on to power the master axis.
       if (so!=1)
              mo=1
       end if
       until (so==1)

       // Check if RLS reached. If so, move relative 10,000 counts (away from
RLS)
       while (ip&0x80)
       // Move towards RLS and reset position
       sp=HOMING_SPEED
       pa=vl[3]
       bg
       until (hf[1]==0)             // Wait until the limit has been captured
```

```
        // Wait 200mS
        wait(200)

        // Configure HF[] array for index capturing
        hf[3]=3                     // Event on Index
        hf[1]=1                     // Arm the homing function
        pa=vh[3]                    // Move towards software limit
        bg
        until (hf[1]==0)            // Wait until Index has been captured

        // Wait 200mS
        wait(200)

        // Configure HF[] array for FLS capturing
        hf[3]=5                     // Event on FLS
        hf[5]=2                     // When the event occurs, do not change PX
        hf[1]=1                     // Arm the homing function
        pa=vh[3]                    // Move towards software limit
        bg
        until (hf[1]==0)            // Wait until index has been captured

        // Disable servo
        mo=0

        // Set software limits relative to home position
        vh[3]=hf[7]-1000
        vl[3]=0

return
```