

gsi Lumonics

DISH Software Manual for the Intelligent Servo Driver



Component Products Group
39 Manning Road
Billerica, Massachusetts 01821

TABLE OF CONTENTS

1 INTRODUCTION 1

2 SOFTWARE INSTALLATION INFORMATION 2

3 GETTING STARTED 3

4 THE NAVIGATION BAR 4

5 BASE 5

6 MATRIX 9

7 TUNING 14

8 QUERY AND TRACE 23

1 INTRODUCTION

The Intelligent Servo Driver (ISD) is a two-axis analog-input/digital-processing programmable servo controller for large and small scan angle applications of the VM500/1000/1500 series of galvos. Its features include:

- 57.6 Kbaud RS-232 communications for tuning, status, and data recall
- 2nd order/type 0/“speed” and 3rd order/type 1/“accuracy” tuning
- Command Feedforward for 2nd order tunes
- On-line tune adjustment/replacement
- Multiple tune storage in flash
- Low-noise and high-speed tuning optimization available
- “Instant” tune switching via SYNC inputs
- Notch filtering to minimize galvo/mirror resonances
- Error compression limiter for large signal stability enhancement
- Wide angle K_T compensation
- Overposition automatic recovery
- Overtemperature and overcurrent protection without fuses

The Digital Servo Host Interface (DISH) is a LabVIEW® executable program designed to provide a full-featured interface to the Intelligent Servo Driver. Features of the DISH program include:

- Real-time tuning controls that emulate the operation of analog potentiometers
- File storage and recall of tunes and system settings
- Interactive notch filter tuner with bode plot display
- Trace readback of ISD internally buffered variables

Program requirements:

- Pentium 500 MHz or greater (Pentium III or greater preferred)
- 64 MB RAM or greater
- Windows 98, 2000 or NT 4.0 (Windows 95 is less stable and its use is discouraged)
- 57.6 Kbaud serial port with hardware flow control
- 20 MB free disk space for installation
- CDROM reader for installation
- Color monitor:
 - Minimum resolution: 800 X 600; Preferred resolution: 1024 X 768
 - Color Depth: True color, 24 or 32 bit color
 - Display font: Small fonts selected
- ISD Daughterboard (for communication with the ISD; may be removed after programming)

2 SOFTWARE INSTALLATION INFORMATION

The DISH is a complex program with stringent CPU power and memory requirements. The program image while running is 25 MB and during operation no part of the program should be swapped to the page file. We recommend allowing at least 32 MB over the memory consumed by the operating system and any other programs running concurrently with the DISH program. Typical systems will require at least 64 MB, and busy systems will require more.

Running the DISH under Windows 95 is possible but not recommended, since an interaction with certain Windows 95 video drivers may crash the system with an error message that refers to “GDI.EXE”. Solutions may involve driver updates or Microsoft service packs. Other problems with Windows 95 may be corrected by installing a recent version of Internet Explorer (5.X or later).

The display color depth should be at least 24 bits (“True Color”). If not, the program will assign CPU power to interpolation of colors, thereby reducing performance. Font type should be set to “small fonts” for best appearance.

The DISH installation program will attempt to install version 6.00.8169 of the Active-X MSCOMM32.OCX control in the system folder to interface with the Windows COMM port. (It has been shown that earlier versions of this control cause memory leaks and instability.) On certain access protected operating systems, such as Windows 2000 Professional, administrator privileges will be required.

The DISH program requires the National Instruments LabVIEW® 6.0.2 Runtime Engine, available in English, German, French, and Japanese. Information related to LabVIEW® Runtime Engine can be downloaded from <http://www.ni.com>.

Installation:

In the CDROM folder “LabVIEW runtime engine” open the preferred language subdirectory and then the 602rt* subdirectory. Double click the file “lvrteinstall.exe” which starts the installation of the LabVIEW Runtime Engine. If you are installing on an older version of Windows that does not have Windows Installer, the installation utility will first install Windows Installer and then ask you to reboot your computer. Installation of Windows Installer may require administrator privileges on access protected systems.

After reboot or if Windows Installer was present proceed with the installation of LabVIEW® Runtime Engine 6.0.2 (please use this version of LabVIEW® Runtime Engine only.)

When the installation of Runtime Engine is complete, navigate to the CDROM directory “dish program\DISH1.30”. If you are installing a Japanese version please use the subdirectory DISH1.30JPN, all others, should use DISH1.30ENG. Double click on the file “setup.exe” and proceed with the installation of the DISH program.

3 GETTING STARTED

For information about supply voltage levels and related information, please refer to the Intelligent Servo Driver Hardware Manual.

Connect the Intelligent Servo Driver with daughterboard to your computer with serial cable P/N 712-74214 and turn on the servo.

Select the DISH program from the Program Files/GSI Lumonics folder.

Start the DISH program by clicking the run arrow (). The configuration window shown in Figure 3.1 will then appear.

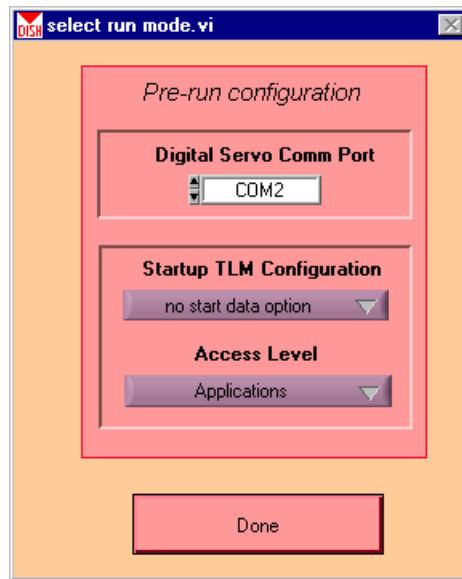


Figure 3.1 DISH configuration window.

Select the **Digital Servo Comm Port** (COM1-COM28) you will be using for communicating with the Intelligent Servo Driver, COM2 being the default. If a non-existent comm. Port is selected, the DISH program will not start.

Select a basic configuration from the **Startup TLM Configuration** list. If you select **no start data**, you will not be able to access the tune window until you define a minimum set of tunes in the Matrix window.

The DISH program supports multiple **Access Levels** with varying levels of complexity. This manual is written for the **Applications** level. It is not possible to change access level without exiting and restarting the DISH program.

Click **Done** and the Navigation bar and Base window will appear.

4 THE NAVIGATION BAR

The buttons on the Navigation bar shown in Figure 4.1 provide access to the various DISH functions.

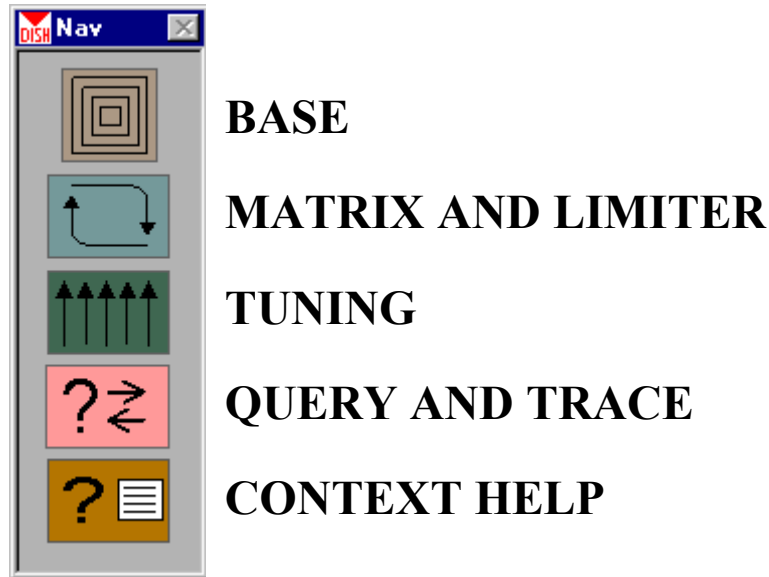


Figure 4.1 The DISH Navigation bar.

Base button

Opens the Base window which provides access to command voltage input range, probe options, last error text box, and various global system parameters.

Matrix button

Opens the Matrix window which provides access to the tune matrix (one or more tunes defined for each axis), offsets, and the error compression limiter. From here the entire state of the servo may be saved to disk or downloaded and saved to non-volatile flash memory. **In general, no tuning can be done until a matrix defining the tuning structure has been uploaded to the ISD.**

Tuning button

Opens the Tuning window which provides access to individual tunes within the ISD tune matrix structure and the enable/disable controls.

Query and Trace button

Opens a window which provides access to error status, servo ID, galvo/servo temperature, and probe trace buffer readback.

Context Help button

Opens a window providing help for each control in the DISH program. Place the mouse arrow over the control or indicator of interest and explanatory text will appear in the help window.

5 BASE

The Base window is shown in Figure 5.1 and its features described below. Click the *send* button associated with each parameter or parameter set to transmit the parameter data to the RAM memory of the Intelligent Servo Driver. **All parameter settings are lost when ISD power is removed unless the state of the system is saved to non-volatile flash memory from the File menu of the Matrix window.** At this time, it is not possible to read back the state of these parameters from the ISD into the DISH.

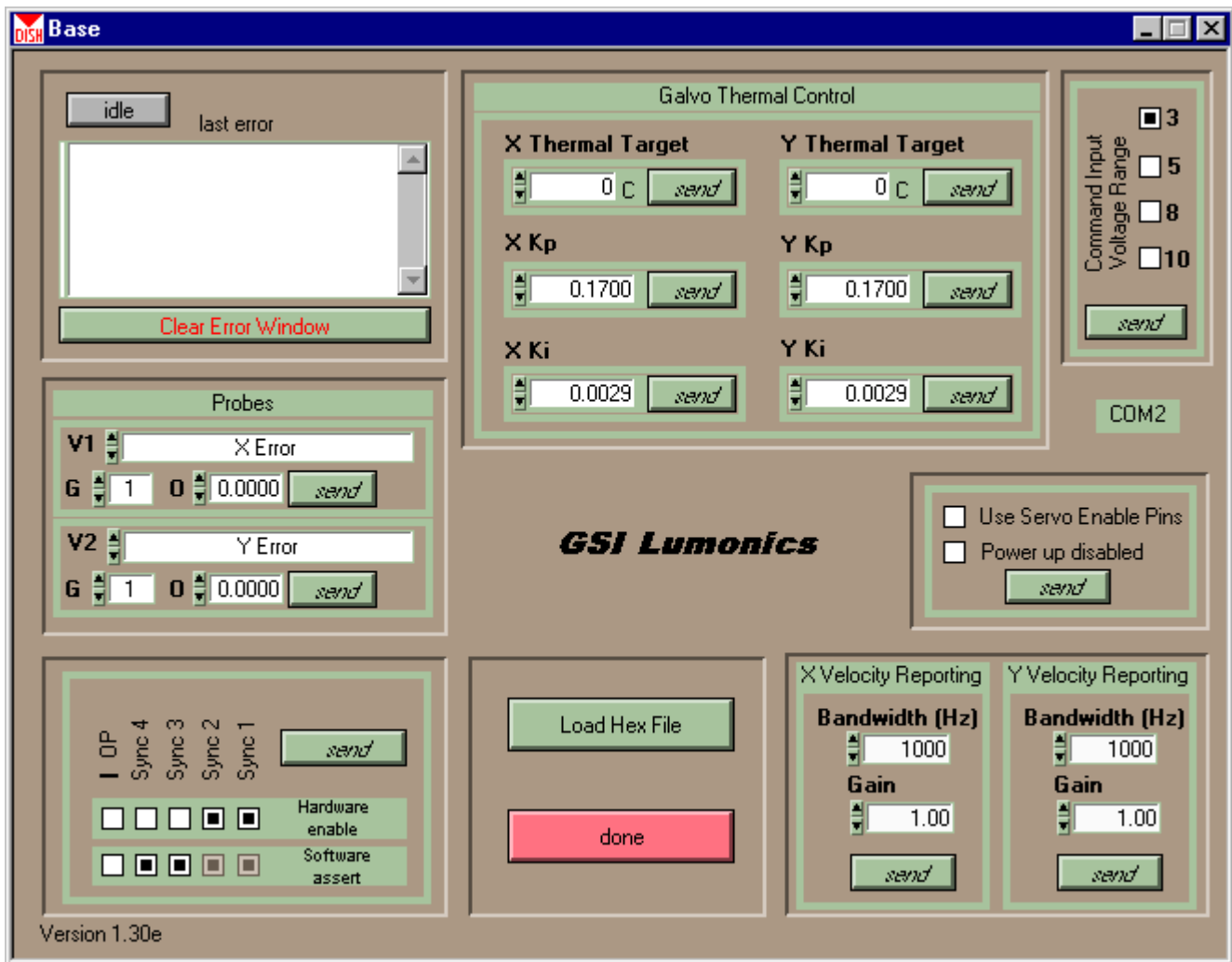


Figure 5.1 Base window.

idle

During transmission of data from the DISH to the Intelligent Servo Driver, the idle button – identical to the one on the Tuning window – will flash **green**. Suspended Tuning window transmissions will cause the idle button to flash **yellow**. The idle button will flash **red** if attempting to send bad data such as invalid tunes of any kind from the Matrix or Tuning windows or if communication with the ISD is interrupted.

last error

The last error window displays the most recent DISH error. It does not report any errors generated by the ISD (see Query section). Typical errors include invalid tunes generated from the Tuning window and serial port timeout if communication with the ISD is interrupted. Click **Clear Error Window** to clear the last error window.

Command Voltage Input Range

Select the desired command input voltage range from the available list (and click the **send** button). The default factory setting is $\pm 3V$. Improper setting of the voltage range will NOT damage the ISD. However, driving an ISD set for $\pm 3V$ with a $\pm 10V$ source may needlessly drive the galvo mirrors into overposition which could damage the mirrors if the stops are not properly set. The actual full scale command voltages associated with the nominal $\pm 3/5/8/10V$ are given in Table 5.1.

Nominal	Actual
$\pm 3 V$	$\pm 3.2516 V$
$\pm 5 V$	$\pm 5.1125 V$
$\pm 8 V$	$\pm 8.3436 V$
$\pm 10 V$	$\pm 10.2045 V$

Table 5.1 Command Input Voltage Ranges.

Use Servo Enable Pins and Power Up Disabled

By default, the ISD powers up with each galvo enabled if 1) the daughterboard is present or 2) the corresponding Servo Enable input is low. **Power Up Disabled** prevents this, and the ISD will wait for **Enable Axis** commands (see Tuning section) or high-low (\square) transitions on the Servo Enable inputs. This option is meaningless until the system is saved to non-volatile flash memory (see Matrix section).

By default, galvos are enabled and disabled by the Servo Enable pins only when a daughterboard is not connected to the ISD. Select **Use Servo Enable Pins** to inhibit the **Enable Axis** command while Servo Enable is high.

See the Intelligent Servo Driver Hardware Manual for more information about Servo Enable.

Galvo Thermal Control

The VM500/1000/1500 series of scanners have integrated heaters for thermal regulation (above ambient temperature) by the ISD daughterboard. **Thermal Target** sets the target temperature for each axis in degrees Centigrade. **Kp** and **Ki** are the error-proportional and integrated error terms of the thermal servo. It is not recommended to change **Kp** and **Ki**. Thermal control is not possible without the ISD daughterboard.

Probes and Velocity Reporting

Through the Test Interface connector of the ISD daughterboard, **Probes** perform a function analogous to attaching oscilloscope probes to the internal workings of an analog servo. Each probe, **V1** and **V2**, can monitor an internal variable of either axis with **Gain** and **Offset** from the following list:

Position	
Command	
Low Resolution Error	(NOT for tuning)
5x Error	(Recommended for tuning, see below)
Integrated Error	
Damping, Low Pass	(from $\Delta\theta/\theta t$)
Damping, High Pass	(from model)
Command FeedForward	
DAC out	(ISD output)
Delta Position	
Delta Command	
Velocity	(6 x field _{rad} krad/s/V or 104 x field rad/s/V at Velocity Reporting Gain=1 & G=1)
Coil Current	(3.3A/V at G=1)
Coil Temperature	(Case temp + I ² R model)
Case Temperature	(25mV/°C)

Probe output voltage is $G \times (V+O) \times 3.1125$ volts. Because $-1 < G \times (V+O) < 1$, probe output voltage is limited to approximately $\pm 3.1V$. The analog Scanner Position readback signals (3.072 volt scale) available on the Control Signal Interface and daughterboard Test Interface connectors will exceed full scale if the scanner is in an overposition condition.

See the Intelligent Servo Driver Hardware Manual for information about the Test Interface connector.

5x Error (error times 4.99) is the variable which the PID servo (see Tuning section) attempts to drive to zero and should be used for tuning. **Low Resolution Error** is not intended for tuning.

Velocity is specified according to **Velocity Reporting Gain** and **Bandwidth** for each axis. For **Gain=1** & probe **G=1**, $\omega/V = \text{rad/s/V} = 6000 \times \text{field}_{\text{rad}} = 104 \times \text{field}_{\text{deg}}$. (Tuning window **field** slider is defined in degrees.)

Reporting measured values inevitably involves some delay as the signal is sampled, processed, and converted back to an analog signal and filtered. In addition, the interleaved nature of sampling and converting in the ISD makes these latencies non-uniform. Table 5.2 gives the approximate delays from measurement to reporting.

Signal	Probe 1	Probe 2
X Command & Position	21μs	14μs
X 5x Error & Coil Current	11μs	18μs
Y Command & Position	14μs	21μs
Y 5x Error & Coil Current	18μs	11μs

Table 5.2 Probe delays.

OverPosition/sync control

The normal operation of the ISD permits tunes to be selected from the tune matrix on the basis of four TTL/CMOS-compatible “sync” inputs. In addition, special overposition (OP) tunes may be designated in the matrix to allow for graceful startup and for the recovery of galvos which have been driven beyond full scale. The DISH program permits simulation of sync input combinations and overposition conditions.

To control one or more OP/sync bits, uncheck the associated **Hardware enables**, set the **Software asserts** to the desired state, and click the **send** button. Hardware enables are exclusive; software asserts are ignored for all hardware enabled OP/sync bits. In Figure 5.1, the tune will be selected on the basis of **011HH**, where **HH** is the state of sync inputs 1 and 0. If the particular state – **01100**, **01101**, **01110**, or **01111** – does not correspond to a defined tune no change of tune will occur (see Matrix section). Note that removing the OP Hardware enable will suppress overposition detection and recovery on **both** axes. Likewise, asserting overposition will affect **both** axes, setting command scaling to zero, and driving both scanners to zero position. The operational state of sync bits is shown in the **servo status** portion of the Tuning window, as are flags for overposition (see Tuning section).

The OP/sync state cannot be saved to flash; overposition detection and all sync hardware enables will be restored after power cycling the ISD. Sync inputs have on-board pullup resistors, so they will default to 1 without connection.

See the Intelligent Servo Driver Hardware Manual and the Matrix section of this manual for more information about the application of OP/sync inputs.

Load hex file

For ISD software maintenance and special loading.

Done

Exit the DISH program. This will have no effect on the ISD.

6 MATRIX

Unless the Intelligent Servo Driver has had a matrix stored in flash, no tuning can begin until a tune matrix has been sent to the ISD. The following explains how to generate and send a tune matrix.

At this time, it is not possible to recall the state of the ISD to the Matrix or Tuning windows. Saving any and all tuning information to file is strongly recommended.

To change the value of a numeric slider:

- 1) edit the value, **or**
- 2) drag the indicator (◀) with the mouse, **or**
- 3) click up/down arrows (▲/▼) to increment/decrement the value by 1% of the displayed range, **or**
- 4) increment/decrement a digit within the value by placing the cursor over the digit and pressing the up/down arrows (↑/↓) on the keyboard.

Slider range is not fixed and may be adjusted by editing the top and bottom range values.

Figure 6.1 shows the Matrix window with a typical tuning structure. Although independent of each other, the X & Y matrices are interleaved in the Matrix window to save space.

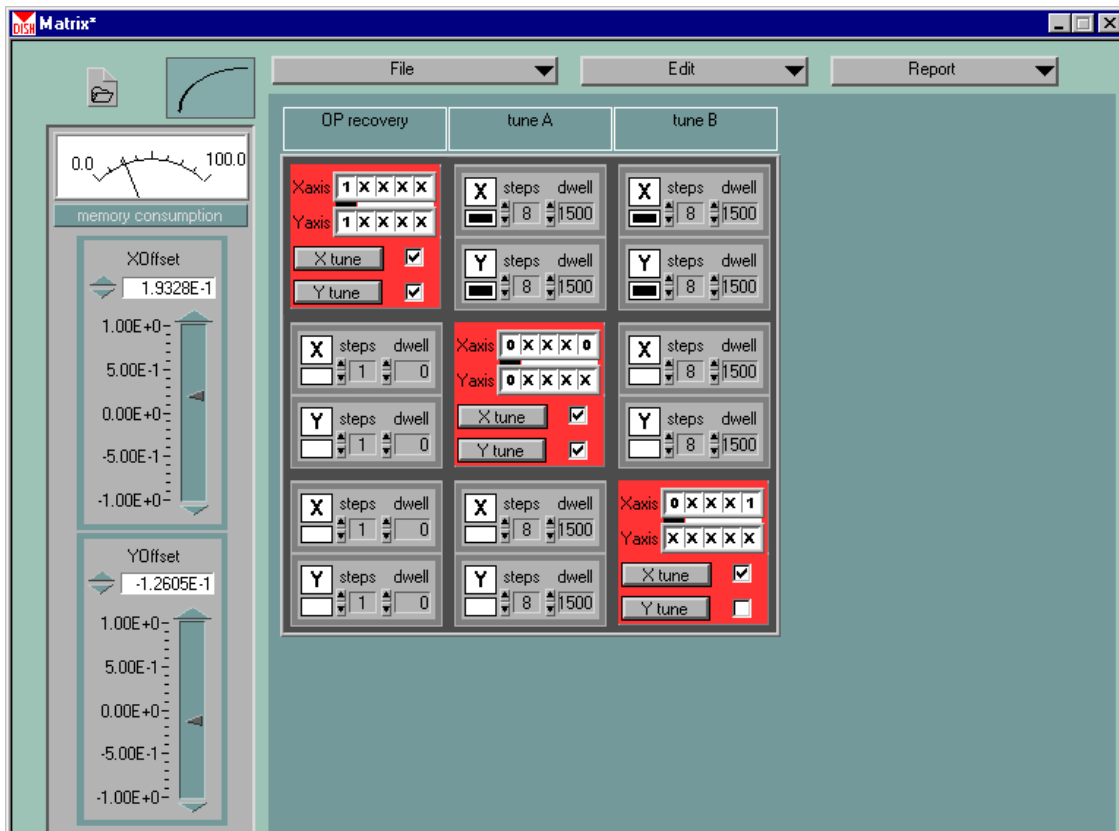


Figure 6.1 Matrix window showing a typical configuration.

For each axis, a tune matrix is a set of tunes and their 5-bit selection rules for using the 5-bit “address” formed by the axis **OverPosition Flag** (OP, see Tuning) and the four sync inputs, shown in Figure 6.2.

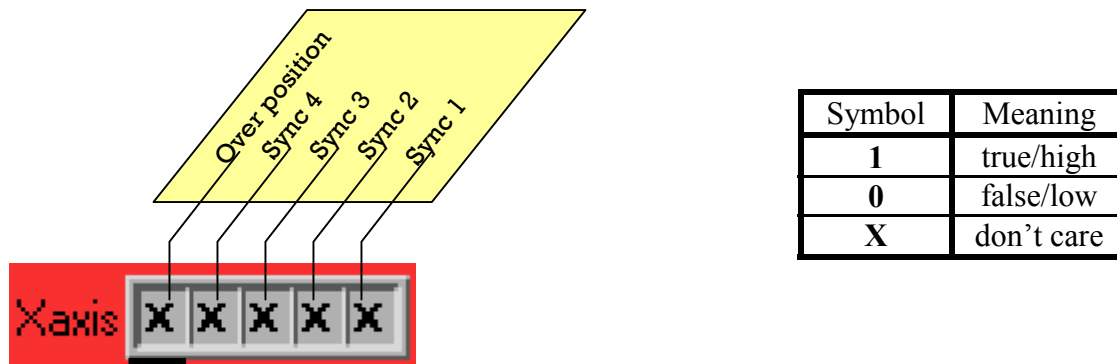


Figure 6.2 5-bit word construction and symbol meaning for tune selection.

Selection rules must be mutually exclusive; two tunes for the-X axis with selection rules 0XXXX and XXXXX would not be allowed because, for instance, 01111 (not overposition, all sync inputs high) would fit both rules. Incomplete rule sets are permissible, e.g. selection rules XXX00, XXX01, and XXX10 for three Y-axis tunes would not include 01111 and the ISD would not change tunes in response to this address.

Sync input pins are internally debounced and their state not internally latched for 120ns after a transition on any pin. The operational state of sync bits is shown in the *servo status* portion of the Tuning window, as are flags for overposition (see Tuning section), and may be manipulated by the OP/sync control in the Base window. See the Intelligent Servo Driver Hardware Manual for further information about the sync input pins.

In Figure 6.1, three tunes have been defined for the X axis and two for the Y axis, as seen in the three **Red** Primary Tune Blocks along the diagonal, where checked tune boxes () indicate defined tunes. For the X axis an overposition state (**1**) will cause the ISD to switch to the overposition tune, named “OP recovery” in this case, regardless of sync pins (**XXXX**). Otherwise, the tune is selected by sync1: low for “tune A”, high for “tune B”. The Y axis will respond similarly to an overposition, but otherwise has only one additional tune defined, “tune A”.

Click the **X tune/Y tune** buttons to load tunes from disk into the X or Y axis of the Primary Tuning Block. Tunes - “X: OP recovery”, “Y: OP recovery”, “Y: tune A”, etc. – are edited from the Tuning window described below. Attempting to load a tune into the Y axis which was originally saved as an X axis tune will require “Cross axis import” confirmation.

Figure 6.3 shows another example in which the “low noise” and “high speed” tunes may be selected independently for the X and Y axes by sync1 and sync2.

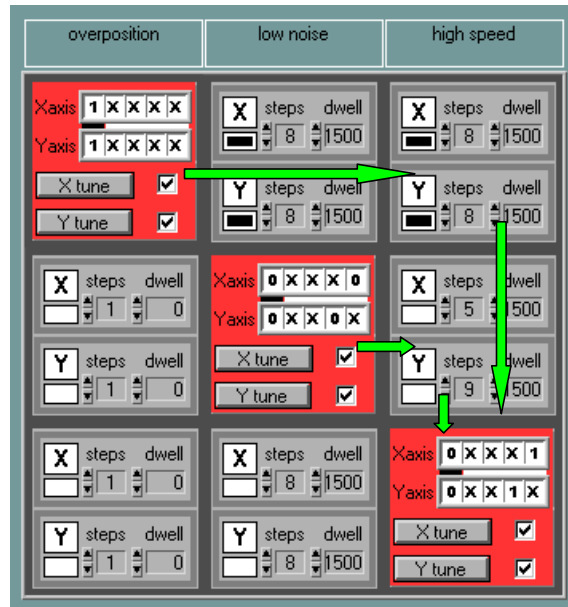


Figure 6.3 Tune matrix with independent X/Y tune control.


When two tunes have markedly different characteristics, an abrupt change from one to another may cause instability, so that transitioning through a number of interpolated intermediate states is required. The off-diagonal Ladder Blocks in the tune matrix indicate how “tune ladders” are to be built between tunes to ensure smooth operation. *steps* is the number of intermediate tunes used to move from one tune to another. *dwell* is the number of 13.44µs servo cycles to spend on each step. These numbers may be entered directly. As the arrows in the Matrix button on the Navigation bar and the green arrows in Figure 6.3 indicate, the matrix is traversed clockwise when moving from one tune to another. Thus, moving from “low noise” to “high speed” involves the Tune Ladder Block at the intersection of row 2 and column 3; the X axis will employ 5 steps (the Y axis will employ 9) and X & Y will each employ 8 steps when recovering from an overposition state to the “high speed” tune. Long dwells do not consume memory, while additional steps do. However, the *memory consumption* of the example in Figure 6.3 was below 50%.

In some rare cases, it may not be possible to form a ladder. Very rarely, an intermediate tune will be unstable. It is also not possible to interpolate between 2nd order and 3rd order tunes. In these cases, the first step of a ladder may be specified explicitly and the latter interpolated from that First Step tune. Click on the relevant X or Y in a Ladder Block to load a First Step tune from file. The darkened block (■) beneath the letter indicates that an explicit First Step tune has been defined. Click on the darkened block to cancel an explicit First Step tune and the block will clear (□). It is not possible to view or adjust First Step tunes or any other tune within a ladder, although limited information about First Step tunes is available in the *Report on Tunes* in the *Report* menu described below.

Overposition

Once the ISD begins to traverse a ladder, sync inputs are ignored until the process is complete, except in the case of an overposition, when more drastic means must be employed. Overposition causes an immediate jump – even from mid-ladder – to the overposition tune (**1** in the first bit) specified by the sync inputs. Therefore, *step*=1 and *dwell*=0 for tune ladders transitioning into overpositions. Furthermore, command scaling is reduced to zero in overposition tunes, driving the galvo back towards zero position. When the overposition state is cleared, the ISD will begin to traverse the appropriate ladder according to the state of the sync inputs at that time and command scaling will ramp up to 1 again. Note that it is possible to have more than one overposition tune for a given axis (e.g. **1XXX0** and **1XXX1**) and that the ISD will move between overposition tunes if the sync inputs change state while the galvo is an overposition state.

Overposition is also used at startup, when the galvo may be far from the origin (zero). When first enabled, the overposition bit is set for one cycle and then cleared, causing the ISD to traverse the appropriate ladder if an overposition tune has been defined. This ensures a stable startup process and is perhaps the most common use of the overposition tune.

Often, 2nd order overposition tunes will be normal tunes with *Dlp Scale* and *Dhp Scale* set to some large value ~25. For 3rd order systems, overposition tunes with *Integrated Error* = 0 are absolutely necessary to allow the system to stabilize at powerup. Because this will require much more than the single cycle of the powerup/enable process, the first step tune is explicitly filled with the overposition tune for all ladders out of overposition as indicated by the darkened First Step indicators () in Figure 6.1 and Figure 6.3 above. This will not adversely affect 2nd order tunes.

File menu

Load System from File

Restores all parameters from disk.

Save System to File

Saves all parameters – matrix, offsets, limiter, and all Base window parameters – to disk (as a .TLM file). The asterisk (*) after “Matrix” in the title bar of Figure 6.1 indicates that changes have been made since this matrix was last saved to disk.

Upload System to Servo

Halts the servo if it is running, transmits the matrix, offsets, limiter, and all Base window parameters to ISD RAM (but not to non-volatile flash memory), and restarts the servo.

Save System to Servo Flash

Writes the entire servo system from ISD RAM (not from the computer!) to flash with the exception of probe settings and OP/sync input enables and asserts; overposition detection and all sync hardware enables will be restored after power cycling the ISD. The system stored in flash will be restored when power is cycled.

Edit menu

Red Primary Tune Blocks are added with the *edit* menu *add tune* command as shown in Figure 6.4. While it is NOT possible to delete blocks, click on a checked *X tune/Y tune* box () to clear the check () and “undefine” the tune.

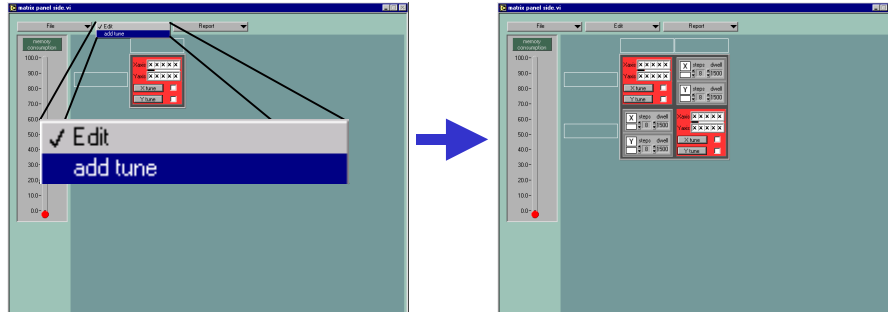


Figure 6.4 Adding tune blocks.

Report menu

Trigger Conflict Window

Reports any selection rule violation, i.e. rules are overlapping/not mutually exclusive.

Report on Tune Ladders

Reports on the validity of tunes and ladders concluding with brief statistics about the matrix.

Report on Tunes

Select a tune from the list of tunes and explicitly defined first step tunes for a listing of a limited number of parameters including the comment box. This is primarily useful for reviewing the contents of an explicitly defined first step tune, since they cannot be selected in the Tuning window.

Limiters tool button ()

Opens the limiter tool window to adjust the ISD's error compression limiter. See the Tuning section of this manual for a complete description.

memory consumption

Indicates how much ISD memory will be required for the matrix.

XOffset/YOffset

These sliders provide a global position offset for each axis and can move the origin $\sim\pm 60\%$ of the field. Any adjustment is transmitted to the ISD immediately.

7 TUNING

With the exception of the **Limiter**, all of the features described below are accessible from the main Tuning window.

Figure 7.1 shows the Applications level Tuning window. The tune displayed is selected from the menu located above the **comments** box. Except for **Second Order/Third Order** selection, **Galvo Parameters**, and **Servo Scaling Parameters**, any adjustment to a tuning window parameter - sliders, model selection, check boxes, etc.- will cause all parameters of the tune to be transmitted to the Intelligent Servo Driver immediately. Changing the **Second Order/Third Order** selection will automatically set the Tuning window to **Off-line**, suspending transmission. (Click **On-Line/Off-Line** to resume). **Galvo Parameters** and **Servo Scaling Parameters** changes are transmitted by pressing the **F1** key or changing any other tune parameter. (Complete numeric entries by pressing the **Enter** key before pressing **F1**.)

If it is not possible to generate a valid tune from the tune parameters the **idle** button will flash **red**, no tune information will be sent, and the last error text box in the Base window will contain information about the problem and how to rectify it. If communication with the ISD is interrupted, the **servo status** portion of the Tuning window will darken as shown in Figure 7.1.

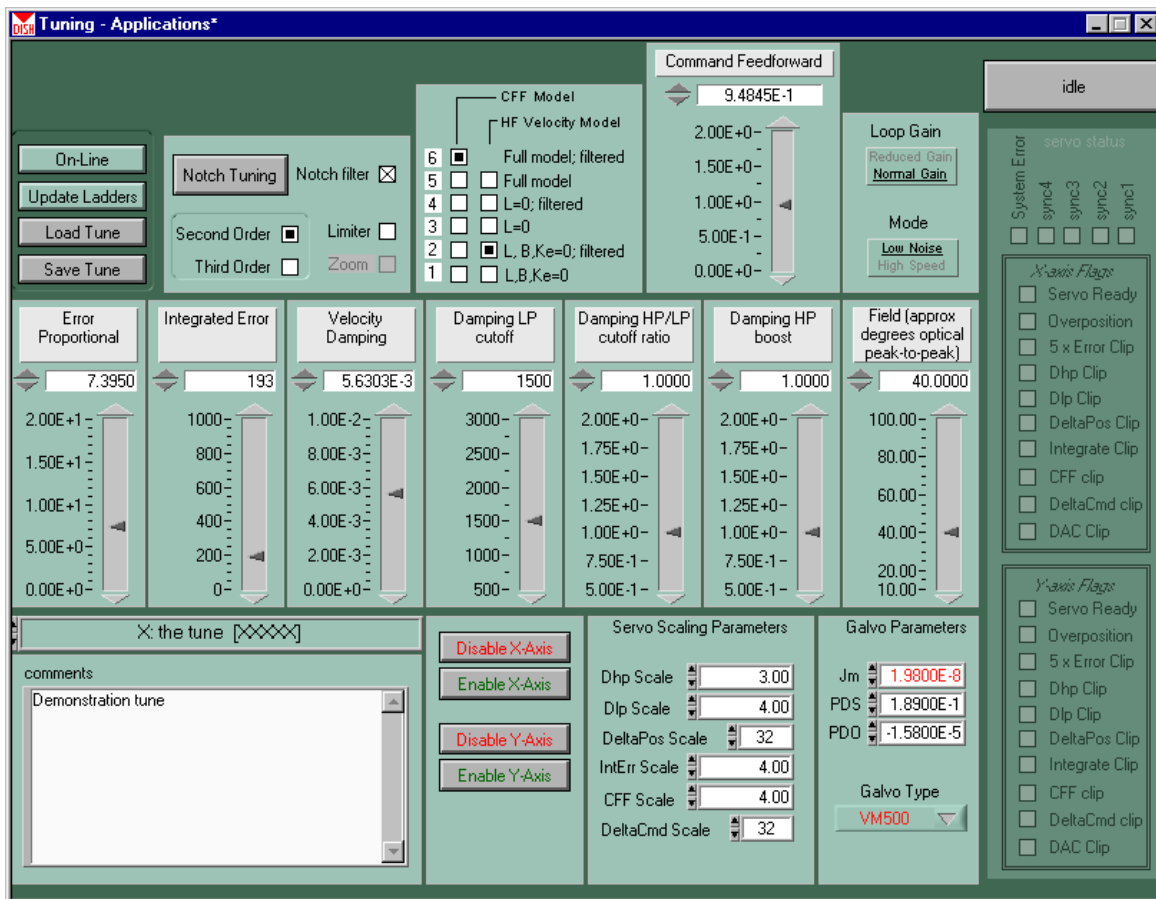


Figure 7.1 Applications level Tuning window.

ISD *Error Proportional* and *Integrated Error* correspond to the usual P and I terms. However, the ISD employs derivative-of-position *Velocity Damping* rather than conventional PID derivative-of-error “error damping”. In addition, the velocity term is a blend of lowpass (LP) filtered measured velocity ($\Delta\theta/\Delta t$) and a highpass (HP) filtered velocity estimate based on output voltage and a model of the galvo (defined jointly by *HF Velocity Model*, *Galvo Type*, and additional *Galvo Parameters*). Taking the derivative of measured position is an inherently noisy process, and the degree of LP filtering required does not provide adequate bandwidth for aggressive (i.e. fast) tuning. Estimating velocity as a function of voltage is basically an integrating process and the model is therefore much less prone to noise, but it has limited accuracy for periods longer than a few milliseconds. Taken together they provide a wideband, low noise velocity signal. *Damping LP cutoff* sets measured velocity LP filter frequency in Hertz. *Damping HP/LP cutoff ratio* adjusts the frequency of the estimated velocity HP filter relative to the LP filter although, as a rule, these frequencies should be identical (ratio = 1.0). Adjusting the gain level of high frequency damping relative to low frequency damping via *Damping HP boost* is more common. Figure 7.4 illustrates the effect of these adjustments. To first order, the selection of crossover frequency from measurement to model is arbitrary and dictated primarily by experiment for a given tune. Experience suggests that increases in cutoff frequency require corresponding increases in HP boost and that matched tunes generally have similar crossover frequencies for each axis.

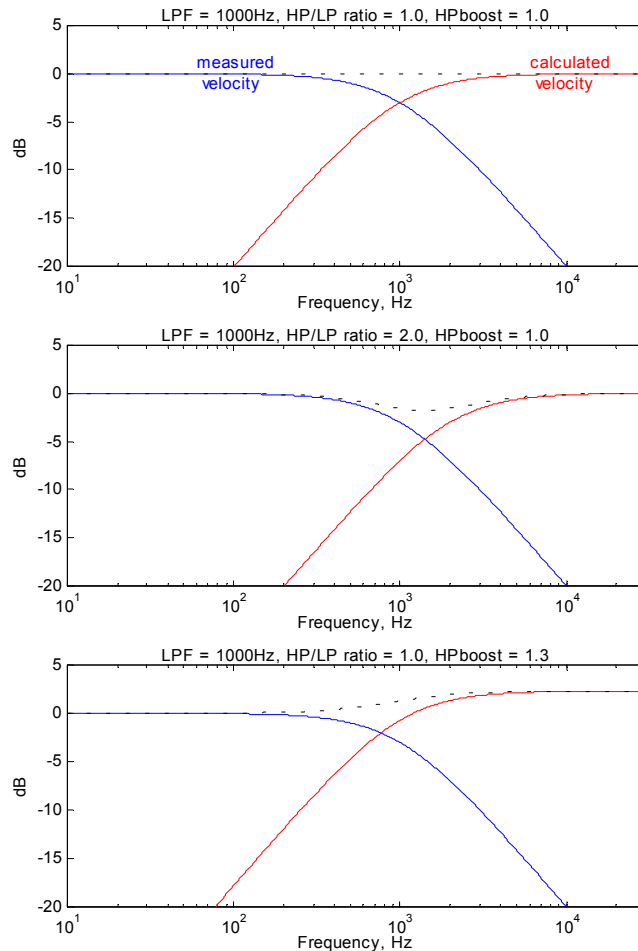


Figure 7.4 Effect of ratio and boost on velocity calculation.

In addition to **Second Order**/Type 0 error proportional “speed” tuning, the ISD also supports **Third Order**/Type 1 position proportional “accuracy” tuning as indicated by switch **S1** in Figure 7.3. In this case, the **Error Proportional** slider is replaced by a **Position Proportional** slider.

In addition to the PID servo, optional command feedforward (CFF) applies the position command signal to an inverse model of the galvo, G^{-1} (defined by **CFF Model**, **Galvo Type**, and additional **Galvo Parameters**), and adds the result directly to the servo output. To the extent that this model accurately reflects actual galvo response behavior, $\text{Command} \times G^{-1} \Rightarrow G \approx \text{Position}$ and, generally, when used **Command Feedforward** ≈ 1.0 . The PID then acts to correct command feedforward error, generally requiring lower gains than without command feedforward. Command feedforward may thus help reduce noise, especially for larger galvos (VM1500). Command feedforward is not applicable to 3rd order tunes and the slider and associated controls will darken when **Third Order** is selected.

To reduce the excitation of resonant modes in the galvo/mirror structure, a second order (bi-quad) **Notch filter** may be applied to the output of the servo algorithm. Click the **Notch Tuning** button to access the Notch Filter Tuner window shown in Figure 7.5 below. The values shown in the Notch widow correspond to the tune selected in the Tuning window, and changes to a notch filter parameter will immediately cause all parameters of the tune to be transmitted to the ISD. The (approximate) filter response is displayed on the graphs. Note that **F ratio** (= F pole / F zero) ≈ 1.0 .

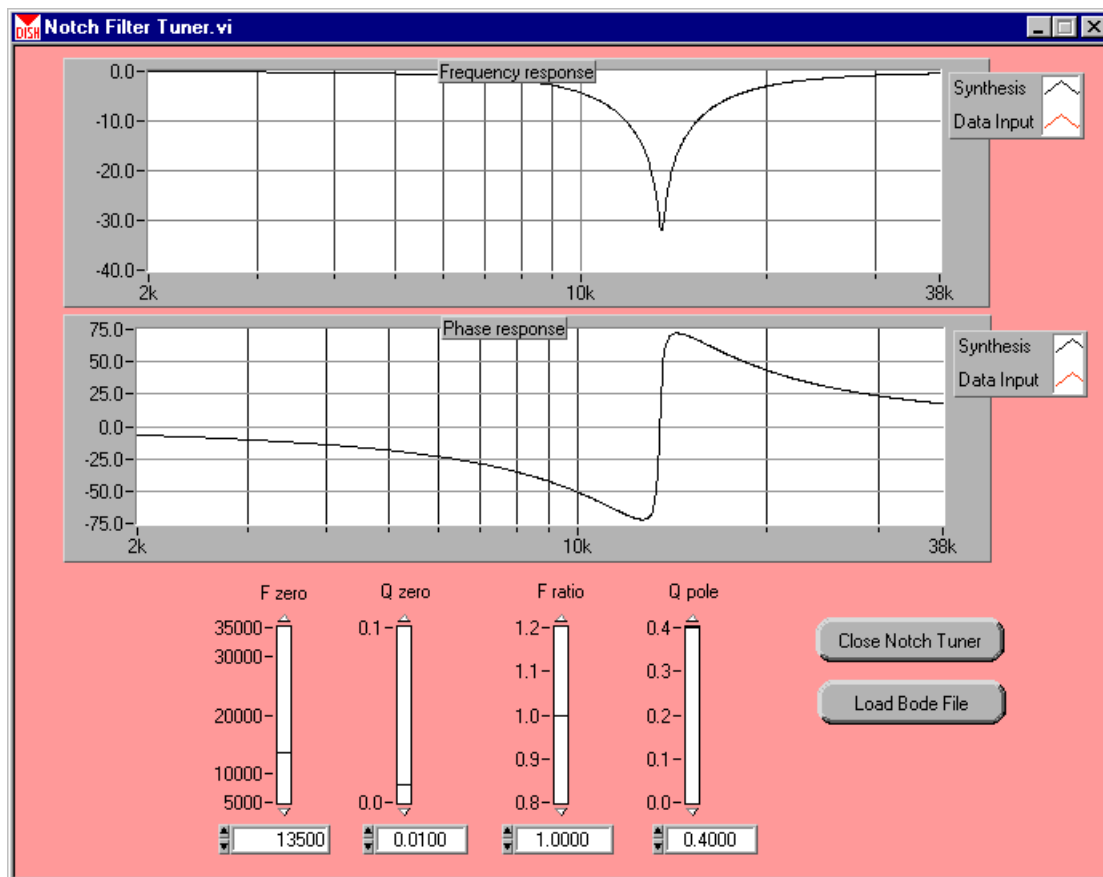


Figure 7.5 Notch tuning window

Wide angle compensation is available to boost output at large angles. When designing a servo system for a typical commutated rotating DC motor, it is reasonable to assume that the relationship between coil current and generated torque is constant regardless of rotor position. However, this is not the case for GSI Lumonics' limited-angle non-commutating galvos, where torque per Ampere falls slightly at large angles. By compensating for torque constant variations, system response is more nearly uniform at all angles of operation.

Although the conventional PID servo of Figure 7.2 is drawn with a “voltage source” output stage, a DC motor/galvo is fundamentally a current-torque converter and it is not uncommon to employ a “current source” as the output stage to functionally eliminate the electrical L/R time constant and thus greatly simplify tuning. However, this assumes large – theoretically infinite – voltage rails. Limited rail voltage leads to voltage clipping of the output, a non-linear behavior which may affect system response. In the ISD, two different settings of **Loop Gain** are designed to reduce – not “eliminate” – the electrical time constant. In the normal case, the output amplifier more closely approximates a current source, moving the time constant to $L/(R+15)$. Reduced loop gain moves to time constant to $L/(R+5)$, which will generally permit reduced noise but slower tunes. The effect of loop gain is included in all galvo models.

Models and their selection

The simplest model of the galvo is simply a resistive torque generator having a specified resistance (R), torque constant ($K_T = \tau/i$), and inertia (J). In control systems parlance, the transfer function to velocity from voltage is $\omega(s)/v(s) = K_T/(JR_s)$. Accounting for viscous damping (B), back-EMF ($K_e = K_T$), and coil inductance (L) yields


$$\frac{\omega(s)}{v(s)} = \frac{K_T}{JLs^2 + (JR+LB)s + (RB+K_eK_T)}$$

This more sophisticated model provides a better representation of the actual galvo response and theoretically allows for tighter, faster tuning but also involves a larger number of calculations, each of which adds a small amount of noise in the form of roundoff error. As a result, selecting an appropriate model involves a tradeoff between speed and noise. Also note that because the galvos designed for use with the Intelligent Servo Driver have typical inductance and resistance values of 220 μ H and 3 Ω , respectively, the default high loop gain results in a time constant of $220 \times 10^{-6} / (3+15) = 12\mu$ s which is even faster than the servo cycle time of 13.44 μ s. In almost all cases, a filtered version of the model is available with a pole near the Nyquist frequency to further reduce noise. For velocity estimation, this filter has a single pole whose default value is 35KHz; for command feedforward, the filter has two poles which default to 3.5KHz and 35KHz. Experience suggests that the best Velocity Model is generally #2 (simple model + filter), while #6 (full model + filter) is best for CFF.

Waveform structuring and error compression limiter

The basic rule of optimal servo design and use is “do not ask for the impossible”. This means restricting servo command waveforms to those which will not exceed the capabilities of the tuned servo/galvo combination. It is not possible for any galvo of non-zero mass to move instantaneously from one position to another. Similarly, the existence of galvo coil inductance and finite voltage rails prohibit instantaneous acceleration. The fastest systems are those whose commands are structured for smooth acceleration and deceleration for all but the smallest changes in position. Failure to observe

these limitations results in 1) underdamped and possibly unstable servo response, requiring 2) additional damping to stabilize the system resulting in 3) a sluggish response.

The ISD's error compressing **Limit**er may be included in a tune to enhance large-signal stability in those systems where waveform structuring is not feasible. Click the limiter tool button () on the Matrix window to open the window shown in Figure 7.6. Error compression limiting is not "slew rate limiting", where a limit on galvo velocity constrains overall servo performance; a properly scaled error compression limiter only takes effect when the command exceeds the capabilities of the tuned galvo/servo combination. Thus, a small step error signal – defined as one which does not significantly saturate the output driver – passes unaltered through the limiter to the PID algorithm, while for larger steps compression becomes progressively more severe. For the limiter shown in Figure 7.6, steps less than 30% of full scale (15% of full field) are not compressed, while the error immediately resulting from a 100% full scale step would be compressed from 100% to 45%. Errors greater than 100% full scale are clamped to 100% full scale (50% field) prior to the limiter. The values shown were chosen for visual clarity only; a more realistic example would be a small step of 0.4° in a 40° field (20° full scale), requiring an **offset** slightly larger than $.4/20 = 0.02$. In practice, **exponent** would also be somewhat smaller (~ 0.02) to provide more aggressive compression. The limiter data is not transmitted each time the offset and exponent parameters are altered; click **Upload to Digital Servo** to transmit the limiter.

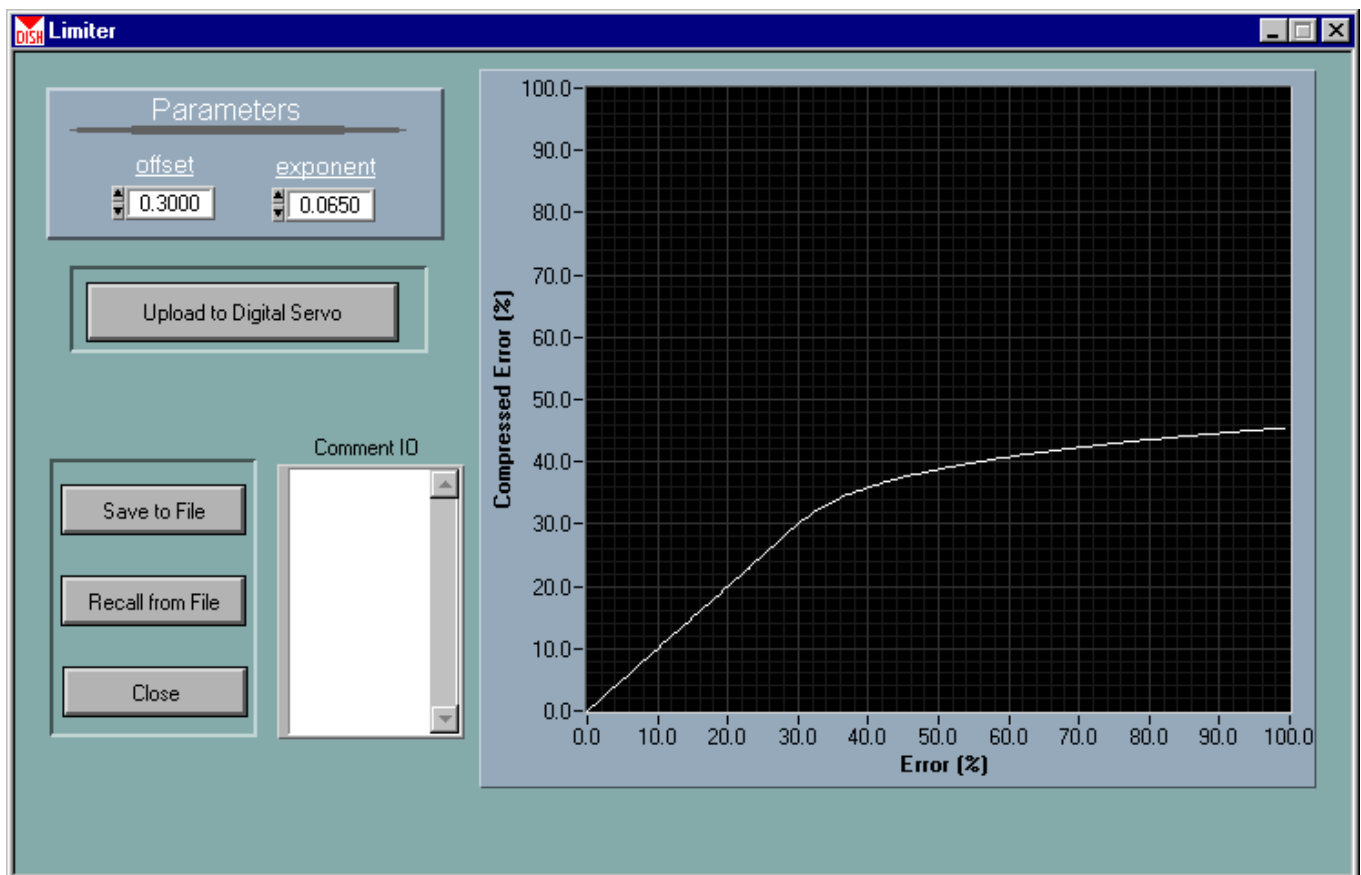


Figure 7.6 Error compression limiter tool window.

Error compression should by no means be considered a tuning parameter, as it is obviously non-linear. Rather, it is a safety net for signals which exceed small-signal tuning limits. For this reason, the limiter is not defined on a per tune or even a per axis basis; a single limiter is defined for the entire servo (and therefore appropriately accessed from the Matrix window). Similarly, the limiter should never be “on” while tuning to a small signal, but should be turned on after tuning is complete and should always have a linear offset region slightly larger than the small-signal limits.

Galvo Type and Galvo Parameters

Select the proper **Galvo Type** (VM500, VM1000, VM1500) to set the appropriate values of resistance, inductance, torque constant, and galvo inertia. (A tune with a galvo type of **Other** indicates that the tune was generated with a galvo parameter set which does not match any of the known galvo types.) In addition, because proper modeling depends on knowing the total inertia $J_T = J_g + J_m$ is it important to enter the approximate mirror inertia J_m . A partial list of mirror inertias (including standard mounting hardware) for the VM500/1000/1500 are given in Table 7.1 in kg-m². Contact GSI Lumonics Customer Support for information on mirrors not listed here.

Mirror Set (optical peak-to-peak)	X inertia	Y inertia
VM500 4mm 60°	5.11x10 ⁻¹⁰	8.81x10 ⁻¹⁰
VM500 5mm 50°	1.56x10 ⁻⁹	2.42x10 ⁻⁹
VM500 6mm 40°	2.11x10 ⁻⁹	3.18x10 ⁻⁹
VM1000 6mm 50°	1.20x10 ⁻⁸	1.26x10 ⁻⁸
VM1000 8mm 40°	1.37x10 ⁻⁸	1.98x10 ⁻⁸
VM1000 9mm 30°	1.63x10 ⁻⁸	1.83x10 ⁻⁸
VM1500 9.5mm 44° x 48°	3.71x10 ⁻⁸	6.46x10 ⁻⁸
VM1500 10mm 50°	5.53x10 ⁻⁸	7.11x10 ⁻⁸
VM1500 12mm 50°	8.60x10 ⁻⁸	1.38x10 ⁻⁷
VM1500 15mm 60°	1.48x10 ⁻⁷	1.89x10 ⁻⁷

Table 7.1 VM500/1000/1500 mirror inertias in kg-m².

The **field** slider should yield a peak-to-peak excursion within 10% of the displayed value. The accuracy of this slider may be improved if the **PDS** and **PDO** values for a particular galvo are known. These values should be adjusted before final tuning, since altering PDS or PDO after a galvo has been tuned will require readjustment of the **Error Proportional/Position Proportional, Integrated Error, and Velocity Damping** terms. Do not alter PDS and PDO unless these values for a particular galvo are known.

Servo Scaling Parameters

The operation of the digital Intelligent Servo Driver involves the numeric representation of physical states such as voltage, position, velocity, etc. Because servo variables are integers limited to a range of ±32768, scaling is required and involves noise/resolution/dynamic range tradeoffs. High resolution will yield better noise performance but may result in saturation (“clipping”) of the variable in question. Low resolution will extend the variable’s range and reduce the likelihood of saturation but the increased granularity of measurement may not provide sufficient precision, thus increasing noise levels. Table 7.2 lists the **Servo Scaling Parameters**, default value, saturation flag of the associated variable, and the

corrective action to be taken if saturation occurs. *X/Y-axis Flags* in the *servo status* portion of the Tuning window flash red (■) to indicate saturation. Note *DeltaPos Scale* and *DeltaCmd Scale* are limited to powers-of-2 (1,2,4,8,...,256) and are also the only multiplicative scaling factors (increasing makes saturation more likely), the remainder being reductive (decreasing makes saturation more likely).

Scaling Parameter	Default	Saturation Flag	Corrective action
<i>Dhp Scale</i>	3.0	<i>Dhp Clip</i>	increase value
<i>Dlp Scale</i>	4.0	<i>Dlp Clip</i>	increase value
<i>DeltaPos Scale</i>	32	<i>DeltaPos Clip</i>	decrease value
<i>IntErr Scale</i>	4.0	<i>Integrate Clip</i>	increase value
<i>CFF Scale</i>	4.0	<i>CFF Clip</i>	increase value
<i>DeltaCmd Scale</i>	32	<i>DeltaCmd Clip</i>	decrease value

Table 7.2 Intelligent Servo Driver scaling parameters.

In general, variables should operate near, but not at, full scale. Figure 7.7, for example, illustrates properly and improperly scaled Low Pass Damping. Note that clipping of the damping term in the graph on the right has resulted in an underdamped small step response with significant overshoot. Any attempt to correct this sort of problem without re-scaling to address the underlying saturation is doomed to failure.

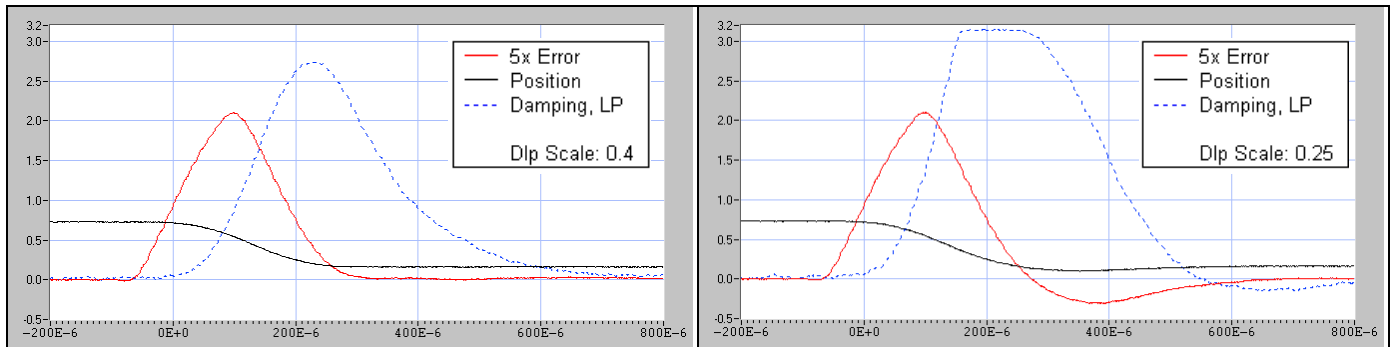


Figure 7.7 10x 5xError, 10x Position, and Low Pass Damping.

In Figure 7.8, *Integrated Error* clipping in a third order tune results made it impossible for the galvo to track the sine-wave command to full scale position. Increasing *IntErr Scale* solved this problem. For third order tunes *Integrated Error* may easily exceed 100,000 and require *IntErr Scale* values of 100 or more.

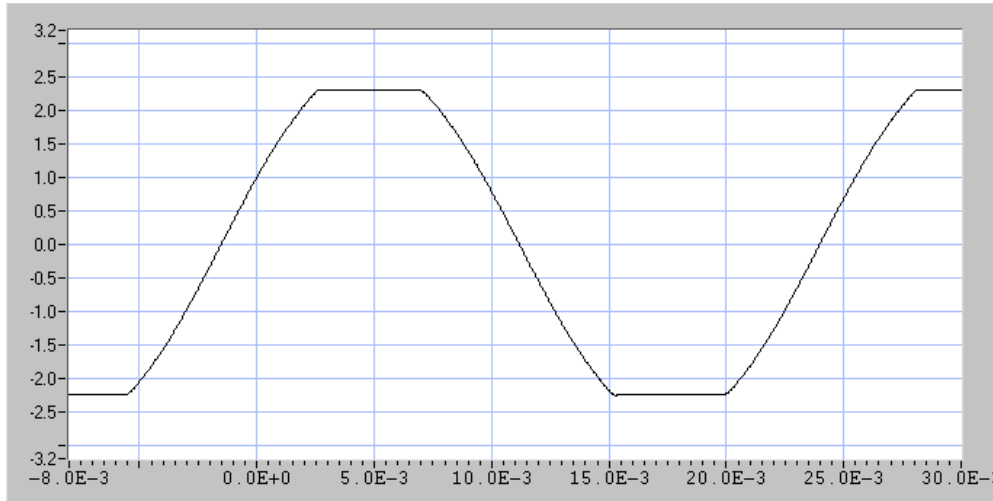


Figure 7.8 Position clipping in a third order tune.

The default values shown in Table 7.2 will suffice for most applications. If, after the system is properly tuned – no clipping! – further improvement in noise performance is sought, first adjust the limiter and enable it in each tune and send typical application command signals to the ISD. Then for each scaling variable in turn, adjust until the clipping indicator lights occasionally and then take slight corrective action so that the variable never saturates. For verification, probe each variable (see Base section).

The ISD tuning coefficients generated from tuning parameters are subject to the same constraints as ISD variables. If one or more variables exceed the valid range, no tune information will be sent, the *idle* button will flash red, and the *last error* box in the Base window will contain information about the problem and how to rectify the problem.

8 QUERY AND TRACE

The query and trace readback window is a set of four tools for retrieving information from the Intelligent Servo Driver.

Error Status

If a problem should be detected by the ISD (overcurrent, overtemperature, etc.), the System Error box on the Tuning window will turn red (■) (as will a real LED if placed across pins 7 and 8 of the Sync Input connector). Select this tab or click in this window if already selected to read the error report. Requesting error status when the box is not red gives the result shown in Figure 8.1. If the error is not fatal, the request will clear the box (□).

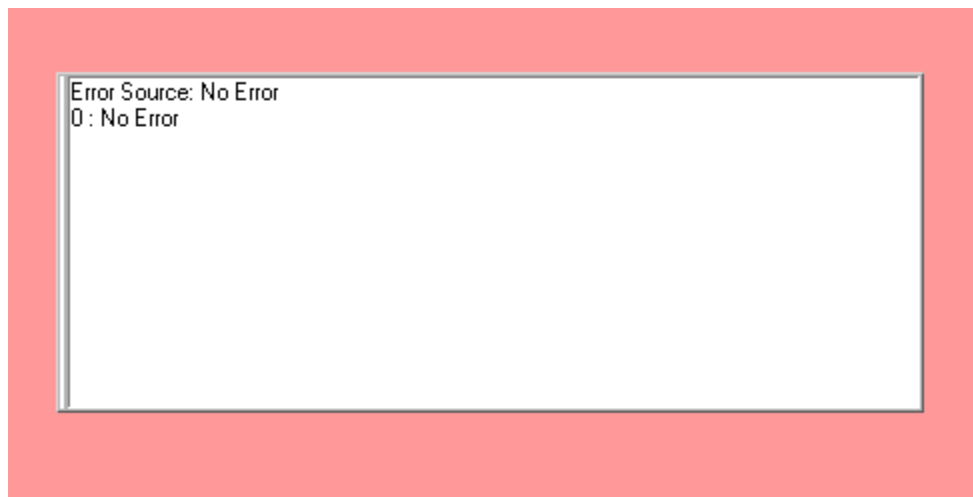


Figure 8.1 Error Status report.

Servo ID

Select this tab or click in this window for a report of the current ISD software revisions. See Figure 8.2.

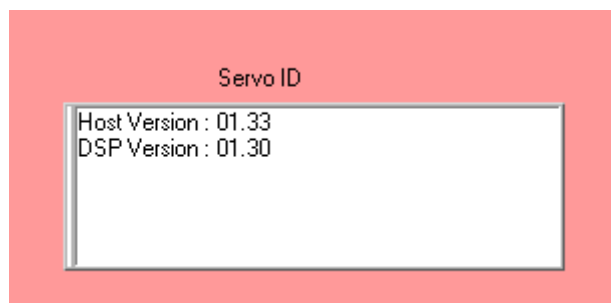


Figure 8.2 Servo ID report.

Temperature

Select this tab or click in this window for X galvo, Y galvo, and ISD temperature in degrees Centigrade. Auxiliary temperature is a fourth temperature input which is currently unused. See Figure 8.3.

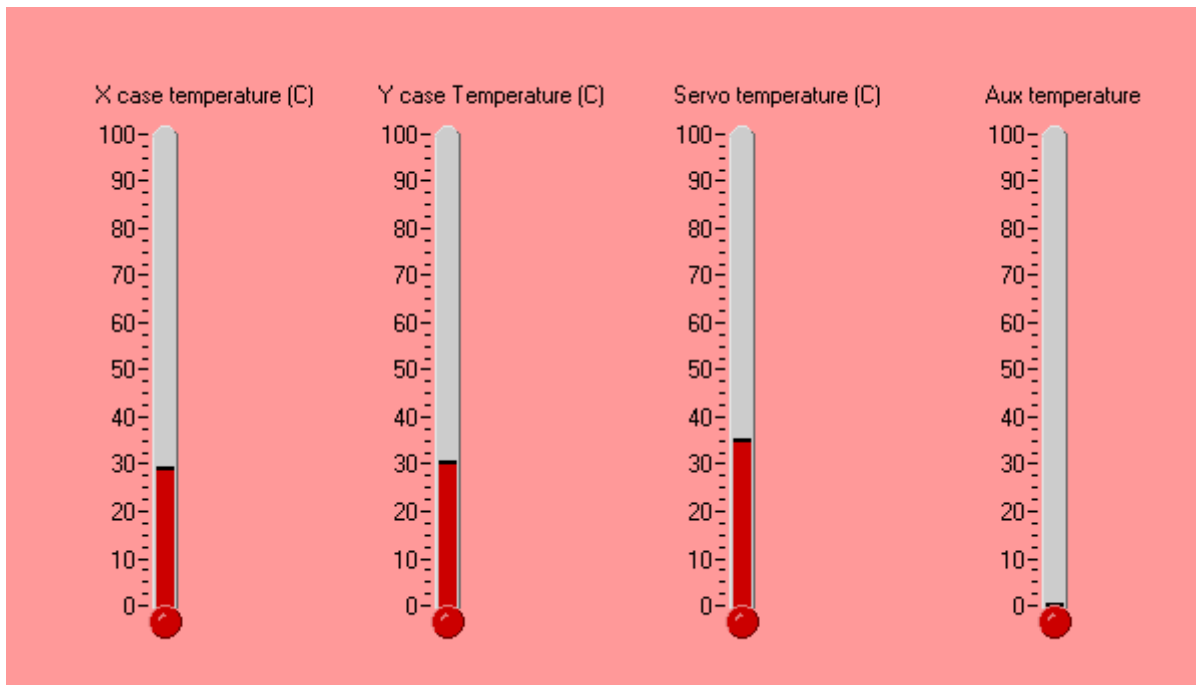


Figure 8.3 Temperature report.

Probe Trace Buffer

This tool, shown in Figure 8.4, allows the user to employ the ISD's internal probe buffers and the computer display as an oscilloscope when access to the Test Interface connector or an oscilloscope is not available. Signal sources are defined by the **Probe(s)** on the Base window independent of **Probe G** and **O** settings, providing the full 16 bits of the internal data stream for individual **gain** and **offset** manipulations within the tool. **Single Trace** supports a single 3500 word buffer which fills from probe **V1** at a fixed rate of one sample per 13.44 μ s for a maximum period of 47ms. **Dual Trace** divides the buffer in two and fills each from probes **V1** and **V2** at the same 13.44 μ s rate. Triggering is external at pin 5 of the Sync Input connector or may be asynchronously forced from the window. Up to eight captured and/or reference waveforms may be arranged on the screen and stored to or recalled from disk individually or as a set.

Config Trace Buffer

Opens the configuration window Figure 8.5. Select **Single/Dual Trace**, **Buffer Size** and **Trigger Location** within the buffer in microseconds, and **Rising/Falling/Dual Edge**. Click **OK** to arm the trigger. **Buffer Size** and **Buffer Location** automatically round to the nearest multiple of 13.44 μ s.

Force Trigger

For use when external triggering is not available.

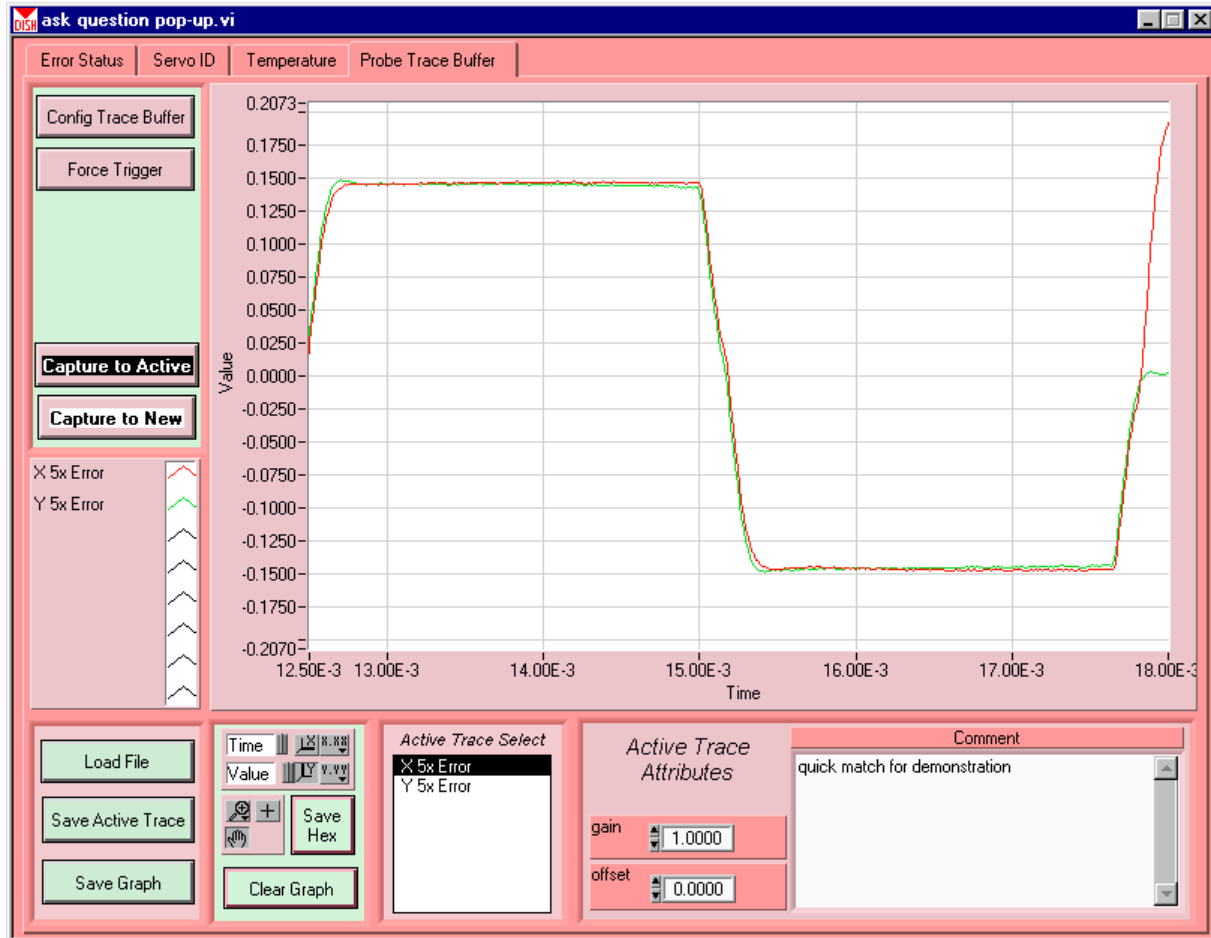


Figure 8.4 Probe Trace Buffer readback tool.

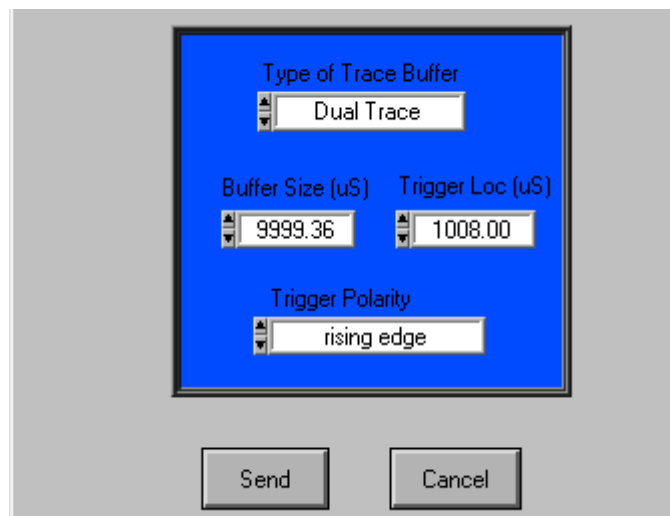


Figure 8.5 Config Trace Buffer window.

Capture to Active and Capture to New

Reads the buffer(s) from the servo, either overwriting existing traces beginning with the highlighted **Active Trace Select** signal or appending to the end of the list. If no trigger has occurred, there will be no response. Otherwise, the window in Figure 8.6 should appear with the selected probe(s) in the left-hand window(s) and the trace label(s) to be displayed on the right. To change a label, click >>> or type in a new label and click **OK**. After a capture, the trigger is re-armed.

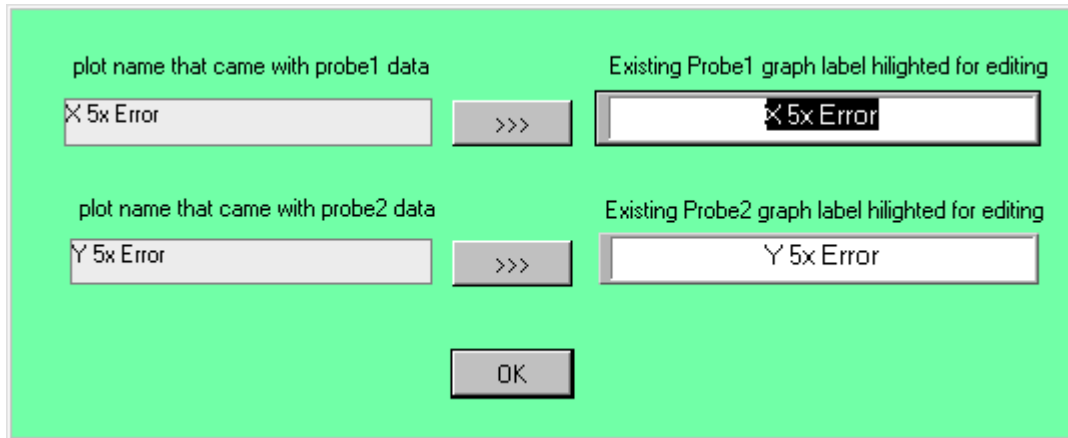


Figure 8.6 Probe Trace Buffer readback label window.

Active Trace Select

Choose a trace from this window to adjust its **gain** and **offset**, add a **Comment**, or **Save Active Trace** to file.

Save Graph

Save the entire set of traces as a graph.