Parker Hannifin Corporation
Electromechanical Automation Division

Electromechanical North America Division
Rohnert Park, CA 94928
Phone (800) 358-9070

Torque mode tuning procedure for the ACR9000 controllers

For an amplifier or drive in torque/force mode, all of the following parameters must be used to get good results. By the time you are finished, assuming the motor and amplifier is matched properly for the load, the tuning should yield maximum following error of less than +/- 5 encoder counts for the entire range of motor RPM.

Gain Parameters.

1.	Proportional Gain	PGAIN	(Volts per encoder pulse of following error).
2.	Integral Gain	IGAIN	(Volts per second per pulse of following error)
3.	Integral Delay	IDELAY	(Seconds after axis stops being commanded to move)
4.	Integral Limit	ILIMIT	(Volts)
5.	Derivative Gain	DGAIN	(Volts per pulses per second)
6.	Derivative Width	DWIDTH	(Seconds) Zero = Default is servo period.
7.	Feed-forward Velocity	FFVEL	(Volts per pulse per second)
8.	Feed-forward Accel	FFACC	(Volts per pulses per second per second)
9.	Torque Limit	TLM	(Volts)
10.	Feedback Velocity*	FBVEL	(Volts per pulses per second)

* Used for dual loop control only.

These gain parameters are controlled and set via the "Servo Tuner" screens in ACR-View. **Make sure you press the ENTER key each time you change any of the Gain values to actually update the controller. Failure to do so will prevent the new value from being sent to the controller.**

If a particular motor/amplifier has never been tuned before, remove the load from the motor so that if the motor runs away during tuning, it is not going to damage anything. If this is not possible, start with the low Torque Limit (TLM) value so minimal damage will occur if the motor runs away due to incorrect hardware setup. In any case, the machine must have HARDWARE OVERTRAVEL limit switches that will prevent any damage if any of the axes do take off towards any end of travel.

Amongst the ten gain parameters, the DWIDTH is one parameter that can be left at zero for most cases. This parameter determines how often the rate of change of following error is sampled. Where there are ample encoder pulses available per rev of the motor (at least 4000 after 4x multiplication) and the motor is not primarily used at very slow speeds, the DWIDTH parameter can be left at zero. This means that the rate of change of following error will be sampled every servo period (normally 500 microseconds). In cases where the operational speed is very slow, the DWIDTH can be changed to 2x 3x, etc multiplier of the servo period in seconds. The FBVEL parameter can also be left at ZERO for all cases that are not using DUAL FEEDBACK encoder (one encoder on the back of the motor for velocity feedback, second encoder on the load for position feedback).

Initial Considerations

a. To capture four channels of information at the highest resolution (servo period) we need to dimension at least 175,000 bytes for PROG0. This is done via the Configuration Wizard in ACR-View. Be sure to "Finish" the wizard and download the entire project after changing any memory allocation.
b. This document assumes the axis is attached to Master0 in the Configuration Wizard.
c. For safety, tune the servo system unloaded. Once the servo is stable and responsive, then add the load and tune the servo again.
d. Make sure motor shaft is free to move about. If the motor is mounted on the machine load, position load axis in the middle of travel.
e. Tune one axis at a time. This means enable only the motor you are tuning.
f. The feedback encoder must yield at least 4000 counts per motor revolution for a suitable operation at slow speed.

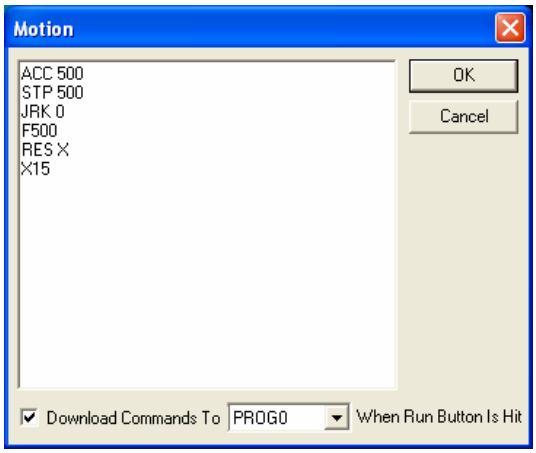
g. Higher resolution encoders will result in lower gain settings. This document was generated using a motor with 4000 pulses per rev. As a general rule of thumb, the tuning values for higher resolution encoders should be close to the values shown in this document times the ratio of 4000 divided by your encoder resolution. This will not be exact, but should be fairly close.
h. **Do not close down the scope until finished.** The scope settings are NOT saved. It will be necessary to reconfigure all scope setting each time the scopes are opened.

Warning!-When tuning a servo motor, remove all loads from the motor to prevent personal injury or mechanical destruction. Once tuning provides a stable and responsive servo motor, you can attach the load and start the tuning process again.

Tuning procedure for axis0 or "X"

1. Set TORQUE LIMIT to a level that ensures that the amplifier current stays within the "continuous current" region of the motor and amplifier. This will prevent overheating of motor and amplifier in case of tuning instability. The Torque Limit setting is based upon the amplifier's command voltage input range and the percentage of that range as it applies to the continuous regions of both motor and amplifier. For our example, we will assume that our drive is matched to our motor with respect to peak current limit. If our motor's continuous current range is 33% of its peak current range we would want to set our Torque Limit to approximately 3.3. Using the tuning gain window, set the Torque Limit to "3.3", or we could issue the "TLM X3.3" or "AXIS0 TLM 3.3" command in the terminal emulator.
2. If you have performed the initial tuning within the Configuration Wizard in ACR-View, then you will already have good starting gain settings for PGAIN, DGAIN, and possibly IGAIN/ILIMIT. If you have good starting values then skip to step 3. If you do not have good values, ZERO out all the gain parameters.
3. Issue a RES X to ensure that the axis position and therefore the DAC output is zero. Now enable (energize) the drive by issuing "DRIVE ON X" in the terminal . Keep one hand on the EMERGENCY STOP pushbutton if motor runs away for whatever reason.
4. If you started the process with valid PGAIN, DGAIN, and IGAIN/ILIMIT settings, skip to step 9. If you set all of the values to zero in step 1, the motor shaft will be "soft" and fairly free to move.
5. Next enter a very low value into the Derivative Gain (DGAIN.) Low value for this parameter usually means "0.000001" assuming encoder is giving about 4000 pulses per motor revolution.
6. At this stage, the shaft of a no load motor will become a little stiff and resist, up to a small degree, any force applied to either direction. Since we have no Proportional Gain (PGAIN) the shaft will not come back to null position but the DGAIN will add some resistance to forces applied to either direction by hand. If the machine load is bolted to the motor, this will not be very evident. Increase DGAIN until the motor shaft loses most of its spongy feeling and resists displacement in both direction even though it will never correct as it is running open loop
7. Next enter a starting value of about "0.001" to PGAIN (for encoder resolutions of 65,536 or higher, start with "0.0001"). The shaft of the motor will now attempt to return to the position it was displaced from.
8. Keep increasing PGAIN in very small increments until the motor begins to buzz and back off until the buzzing stops. For high-resolution encoders of 65,536 counts per rev or more, the value for PGAIN will be quite small. A test performed with a Compax3 amplifier with a resolver-based MPM motor produced a no-load PGAIN of "0.0008".
9. Now at this stage it should be possible to make closed loop indexes and watch following error. **Please make certain that it is safe to perform the moves before doing so.** We will set up an index initially for maybe 10 revs at 500 RPM to start off. Keep acceleration (ACC) and stop/deceleration (STP) so that the system will come up to speed in the desired time for the application. As an example, if finally one wants to run at 4000 rpm and wants to accelerate 0 to 3000 RPM (50 RPS) in 0.1 seconds then use formula "*a = v/t*" and come up with 50/0.1 = 500. This means that "ACC500 STP500" should do the work well.

10. In the Servo Tuner Oscilloscope screen in ACR-View select "MOTION". Enter the following code into the text box:



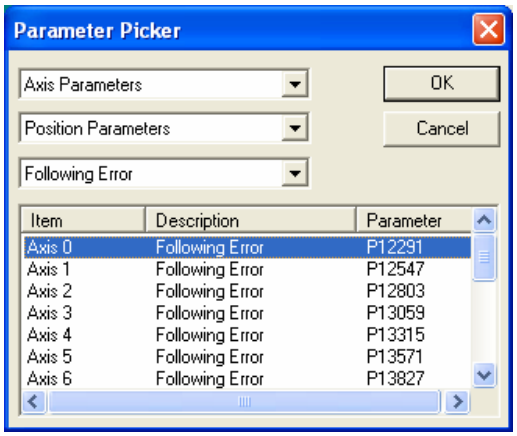
    ACC 500
    STP 500
    JRK 0
    F500
    RES X
    X10

    Note: You could also change code to make moves that are more indicative of your desired motion.

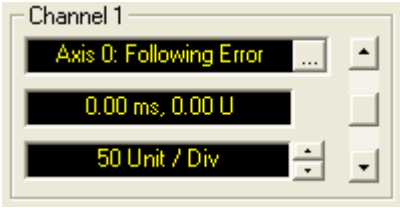
    Ensure that "Download Commands to PROG0 When Run Button Is Hit" is checked. This will send the code to the controller when the Oscilloscope is set to RUN or SINGLE.

11. Select Parameter for Channel 1:

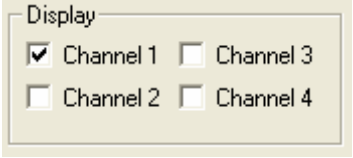
    a. Press "…" to open "Parameter Picker"
    b. Select "Axis Parameters" from top drop-down
    c. Select "Position Parameters" from middle drop-down
    d. Select "Following Error" from bottom drop-down
    e. Finally select the desired axis from list.



12. Set Channel 1 Scaling to 50 units/division.



13. Disable other channels by deselecting the appropriate "Display" checkbox.

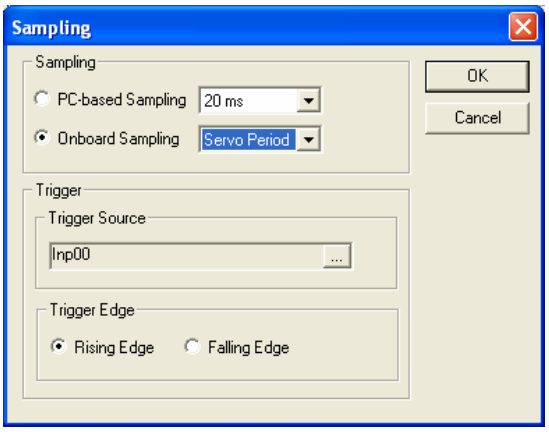


14. Select "SAMPLING…" and configure the scope for Controller-based sampling:

    a. Select "Onboard Sampling"

    b. Select "Servo Period"

    c. Click on "…" under "Trigger Source"



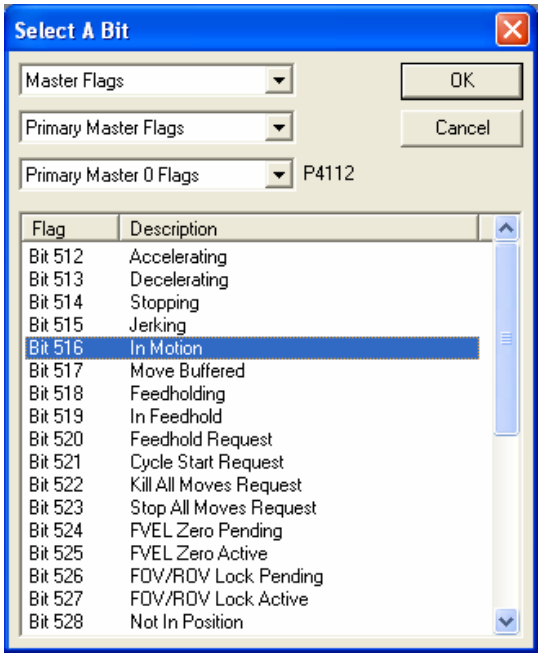
    d. In "Select A Bit" Window:

    Select "Master Flags"

    … then "Primary Master Flags"

    … followed by "Primary Master0 Flags"

    … finally "Bit 516 – In Motion"



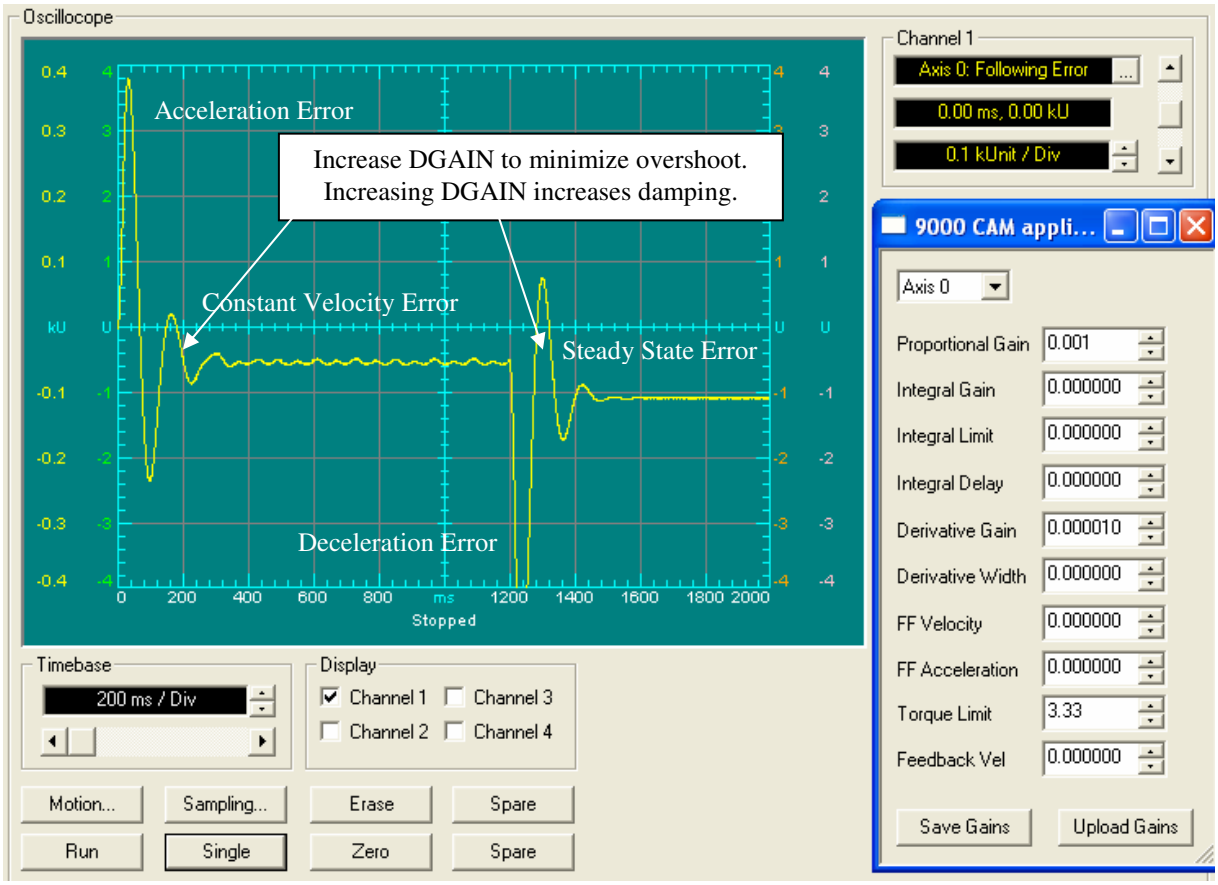
15. Set "Timebase" to 200ms. We wish to capture the entire index during the ACCEL, AT SPEED and DECEL portion. There are a total of 10 vertical divisions on the scope. As a estimate, assuming ZERO acceleration the index should take approximately

    10 revs * 60 seconds / 500 RPM = 1.2 seconds.

    This means that if we choose the Timebase scale of about 200 ms/div, we will see the entire profile.

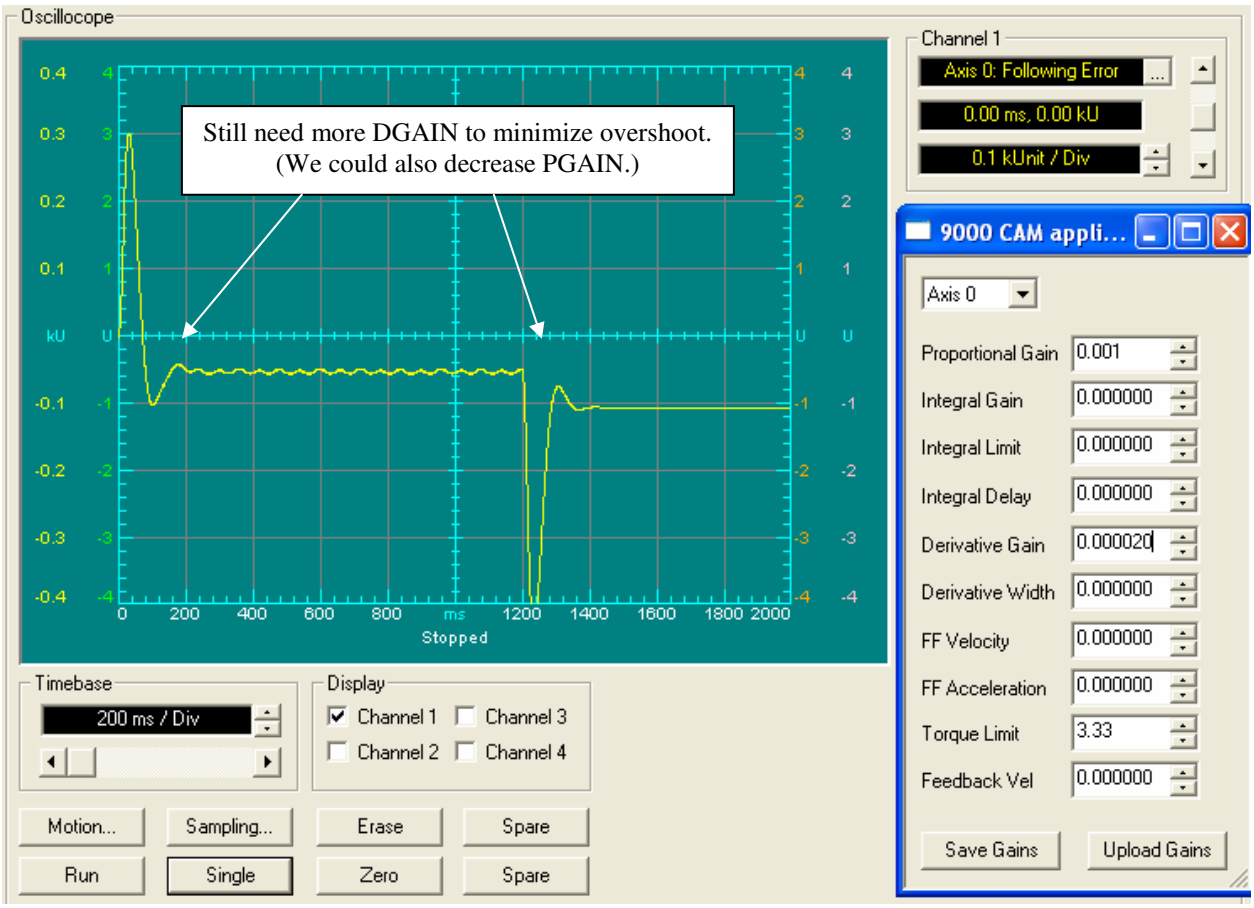
16. Now press "SINGLE". This will result in the motor moving 10 revs in the forward direction and stop. If no waveforms are shown, open the terminal emulator and look for error messages such as "Out of memory", "Unknown command", and "Out of range". "Out of memory" and "Out of range" indicate that not enough memory was set aside for PROG0 in the Configuration Wizard. Update the Configuration Wizard, Finish, then download configuration, again. "Unknown command" indicates the axes were not attached to Master0.

17. Tweak the Channel 1 setup parameters to get a good look at the profile and execute the profile again (see below). At this stage the following error is going to be probably off scale. Now we must manipulate the PGAIN, the two DGAIN parameters, the three IGAIN parameters and the two Feedforward parameters to bring the gain under control. At this stage the profile might look like the following. The result is that the profile will not end up using the entire screen.



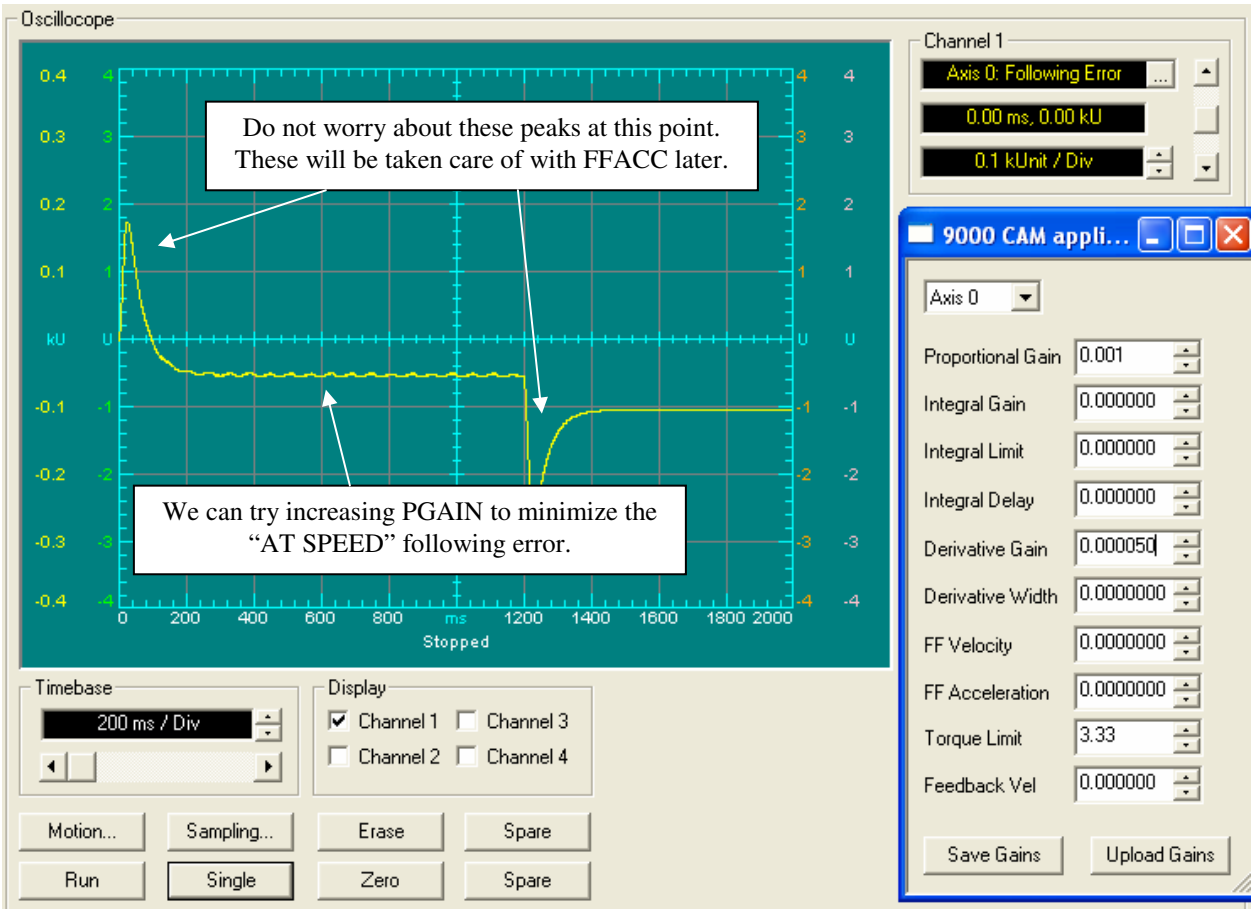
Note that there is a lot of ringing during the ACCEL portion of the profile and the DECEL portion at the end end. Note that the Horizontal scale is 200 msec/div and the vertical scale is 100 pulses per division.

Also there is a steady error during the "at speed" portion of the profile. Additionally there is steady state error after the profile is over and done with. All of these will be taken care of before we are done. Intuitively, note that PGAIN effects the entire portion of the profile. DGAIN effect portions where the following error is CHANGING. This is verified quickly by looking at the ringing which seems very evident during the ACCEL and DECEL portions.
The ringing is due to not enough DGAIN in relation to the PGAIN. Note that the DGAIN and PGAIN kind of work opposite to each other in the sense that PGAIN adds and DGAIN subtracts to from the signal. This next printout is when DGAIN is increased to 0.000020. Everything else remains the same.



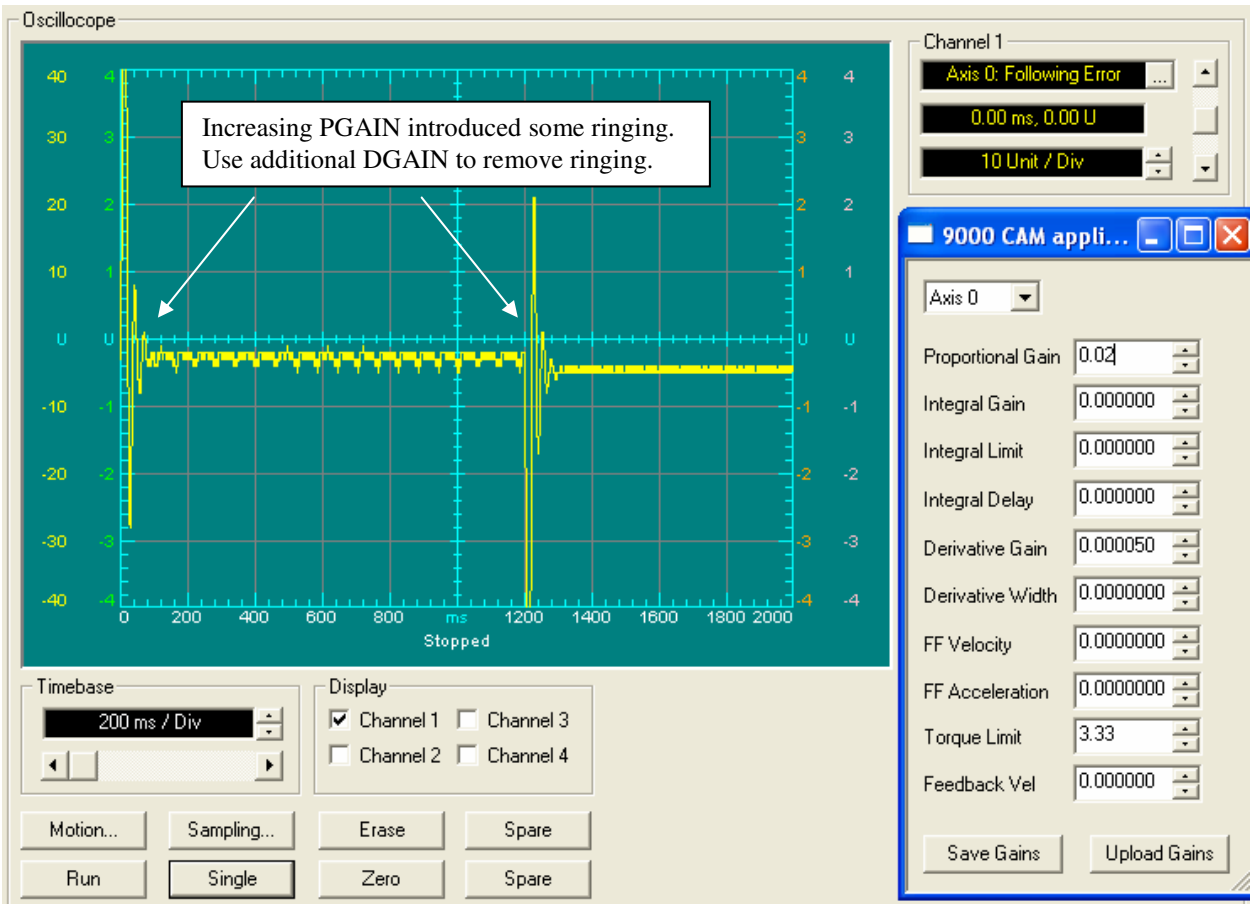
Note that most of the ringing is gone. Note also that the ringing is less but the flat following error during the at speed portion of the profile has not changed

We must increase the DGAIN further to get rid of the last of the ringing. The next profile illustrates the point. If DGAIN is set too high, the axis will begin to emit an audible (vibration) noise. If it starts making the audible noise, then we will need to lower DGAIN, then lower the PGAIN to minimize the overshoot.



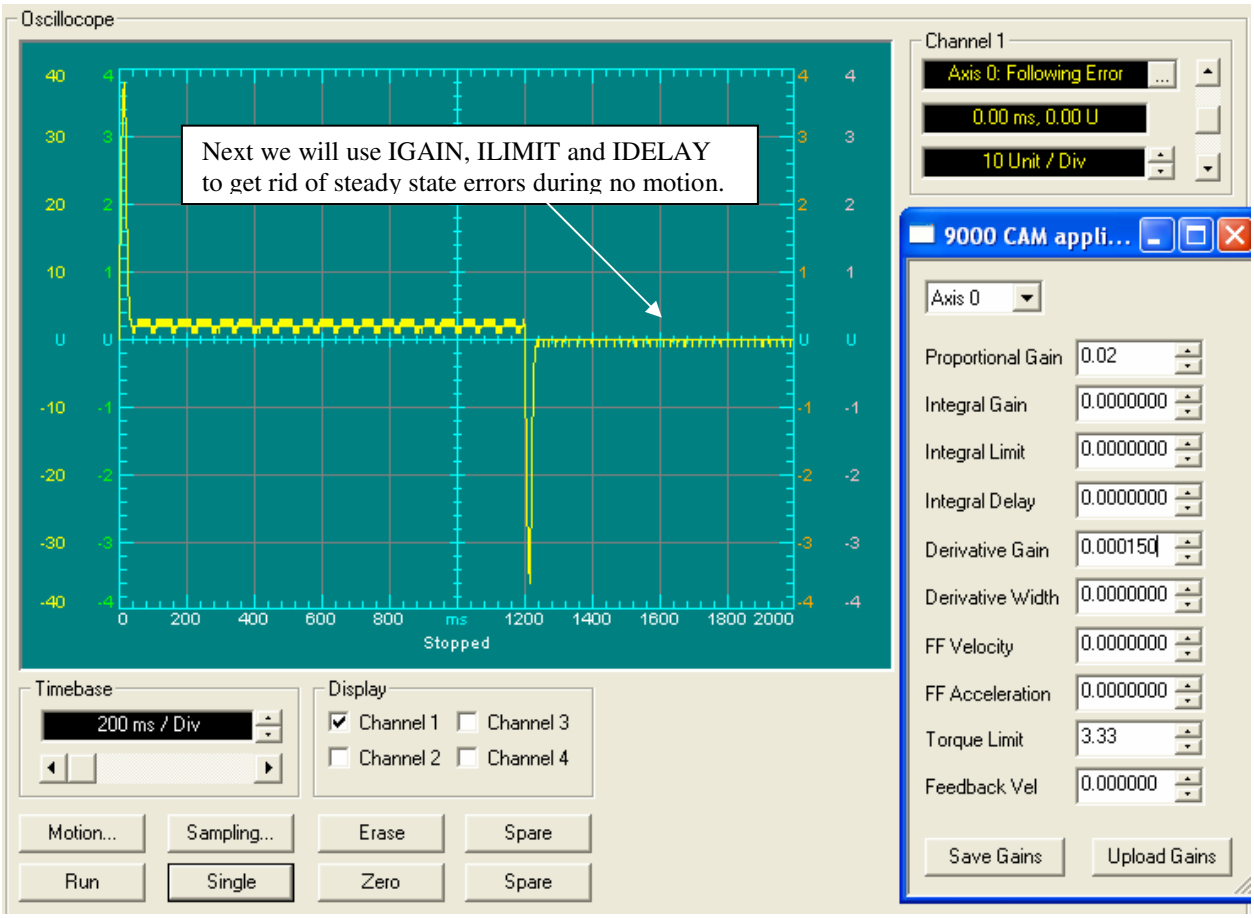
Note that now the ringing is gone. Now we can try to bring the AT SPEED following error down by increasing the PGAIN.

Just like the DGAIN, If you get the PGAIN setting too high, the motor will start to buzz and may go unstable if set excessively high. Do not try to chase out all of the following error with PGAIN and DGAIN. The spikes in the acceleration and deceleration phases will be removed by the adding of FFACC in a later step.

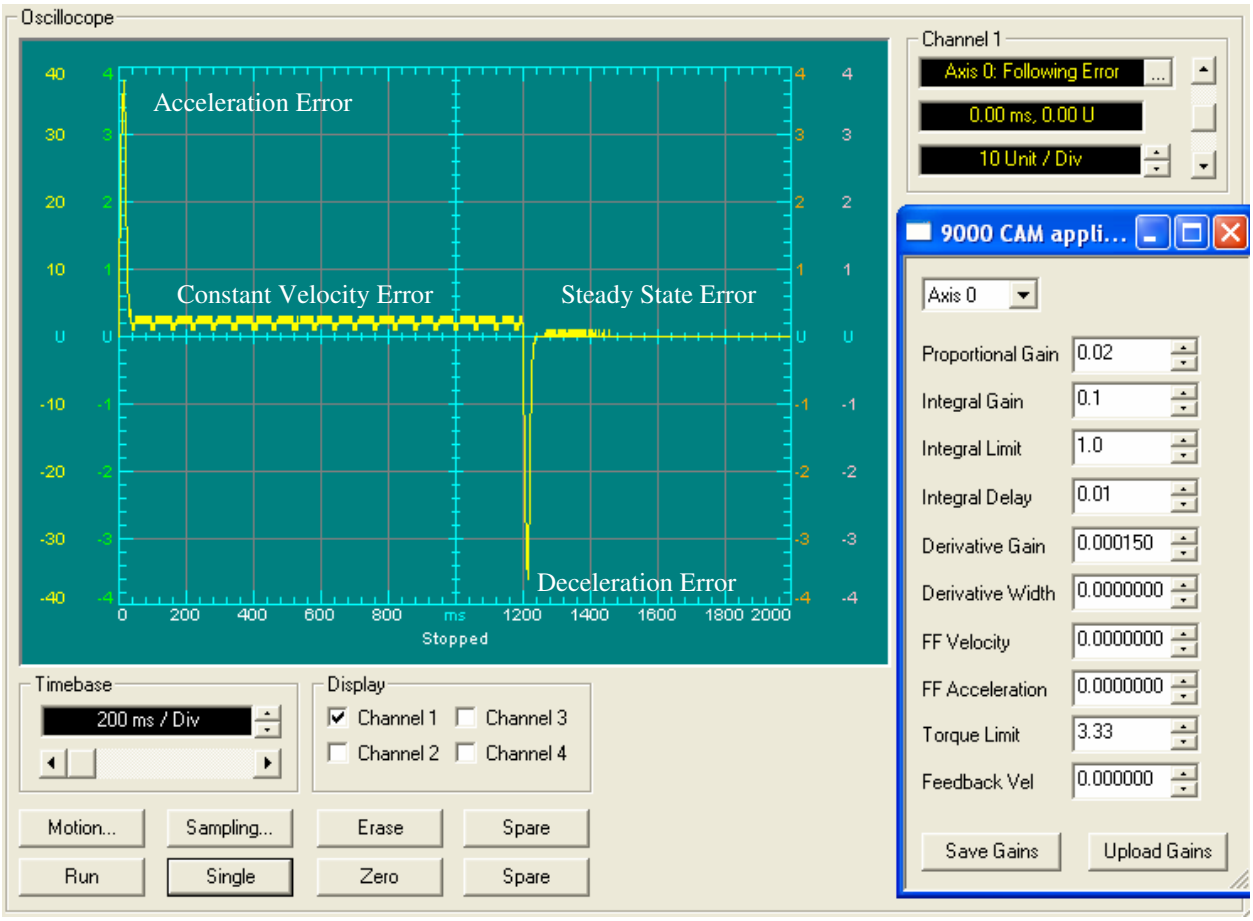


The above trace is the result of just increasing PGAIN to 0.02 and adjusting my Channel 1 units/div setting to 10. Adjust your scope channel settings as needed to display the information with a good resolution. You can see in the scope trace above that I am bouncing back and forth between -1 and -2 counts of error during the constant velocity portion of the move.

Note that with this gain the following error has all but disappeared. As you can see, we still have some ringing. We need to add some additional DGAIN to reduce this ringing. Again, if excessive DGAIN causes the axis to make audible noise, it will be necessary to reduce PGAIN instead.



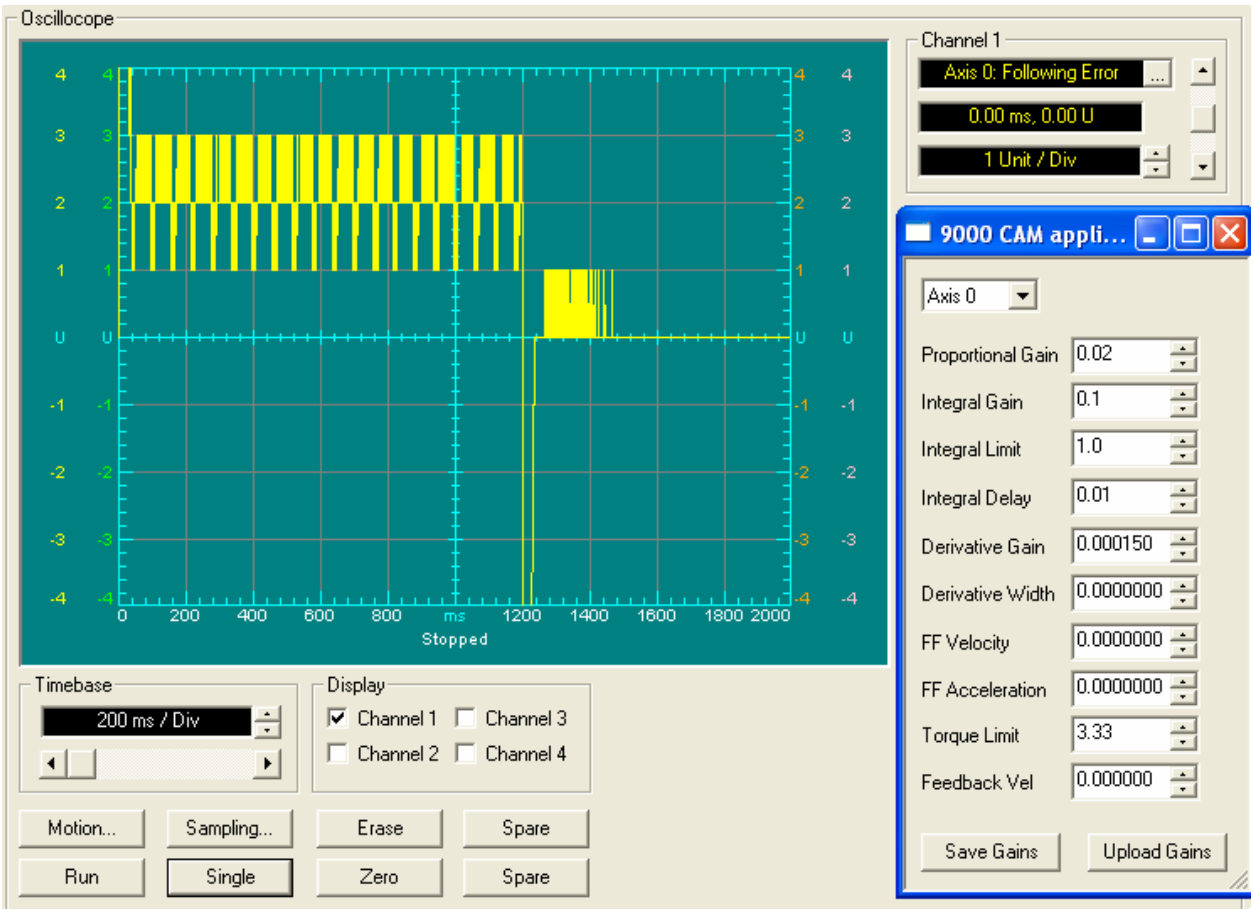
Increasing the DGAIN to 0.000150 removed the ringing. We are still not concerned with the spikes of following error in the ACCEL and the DECEL portion of the profile. Next, we can take care of any steady state errors by bringing in some IGAIN and ILIMIT.



Setting the ILIMIT to 1 Volt, IGAIN to 0.1 V/Sec/count and our IDELAY to 0.010 seconds, we can see that the steady state following error tapers off after approximate 10 ms. Let's zoom in on the waveform to get a better idea of what is going on in the constantly velocity portion and the steady state error (no motion) portion.

Setting IGAIN without setting ILIMIT to a non-zero value will causet he IGAIN to not have any affect on the following error.

IDELAY or integral delay determines the amount of time, after a move ends, before integration begins. If the value is set to zero, integration is active all the time, even during moves.



Changing Channel 1 Units/Div setting to 1 we can note that the steady state error at the extreme right of the trace is showing 1 pulse error only!

There is some constant velocity error during the AT SPEED portion of the profile.

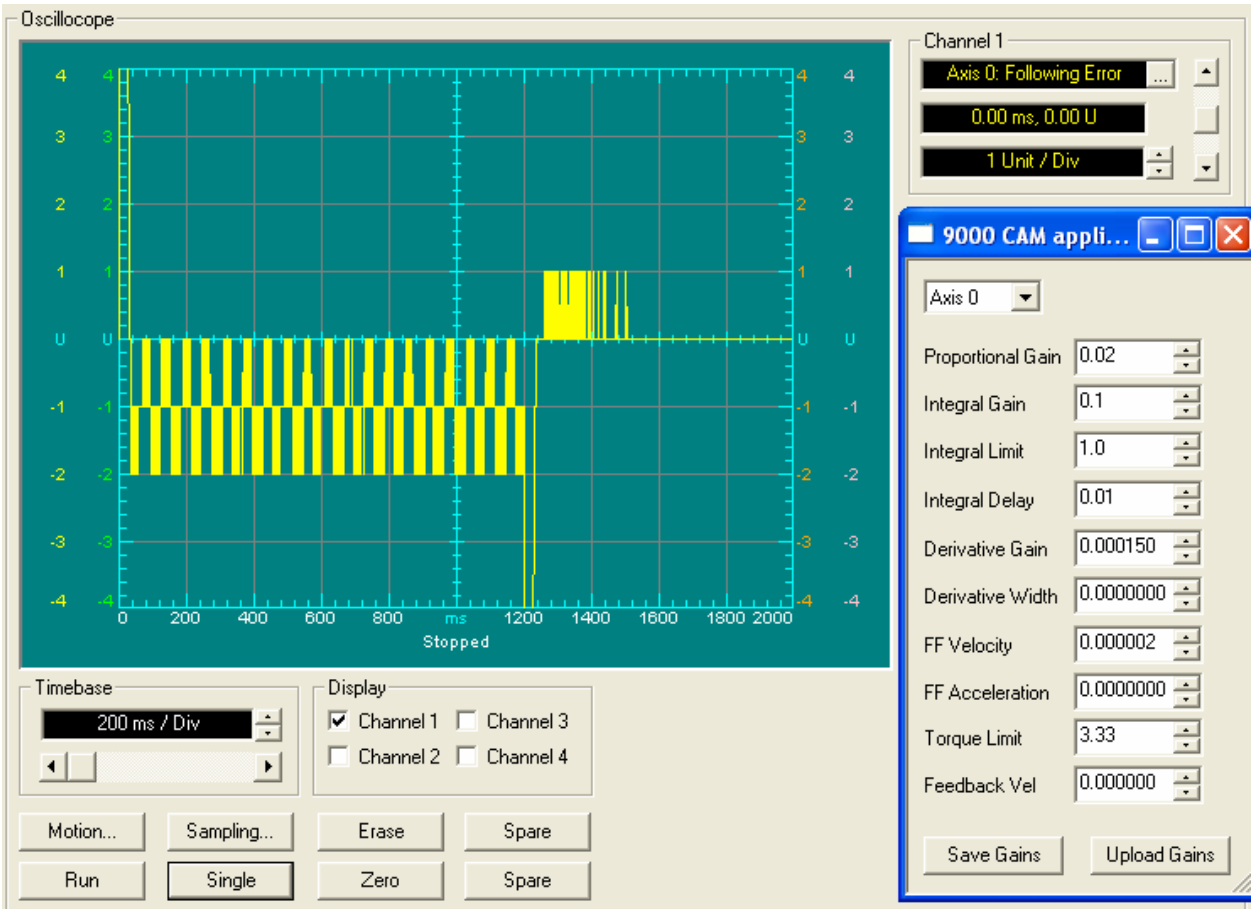
Also, the IGAIN is only a temporary value for now. The IDELAY is also setup for now such that the IGAIN only will start correcting 10 msec after the axis stops been commanded with a profile move. We will correct the AT SPEED portion error with the feed-forwared velocity (FFVEL) parameter.

The ACC and DEC portion error will be fixed by the feed-forward acceleration(FFACC) parameter.

Next we will modify the FFVEL parameter to bring the AT SPEED following error further down. We can do this one of two ways, either guess at a value or calculate the value. Let's try guessing at the value and enter a FFVEL setting of 0.000002.

The value of FFVEL will normally be extremely small similar to our value.

Now we hit the SINGLE button and acquire a new trace with the FFVEL setting.



Notice that now the AT SPEED following error has FLIPPED and is now negative. This means that the FFVelocity parameter was too large and we need to decrease the setting. Guessing at a FF Velocity value is the difficult way to tune.

The FFVel value can be calculated from the following formula.

FFVel = (Pgain in Volts per pulse)* (Following Error In Pulses) / (Velocity In Pulses/Seconds)

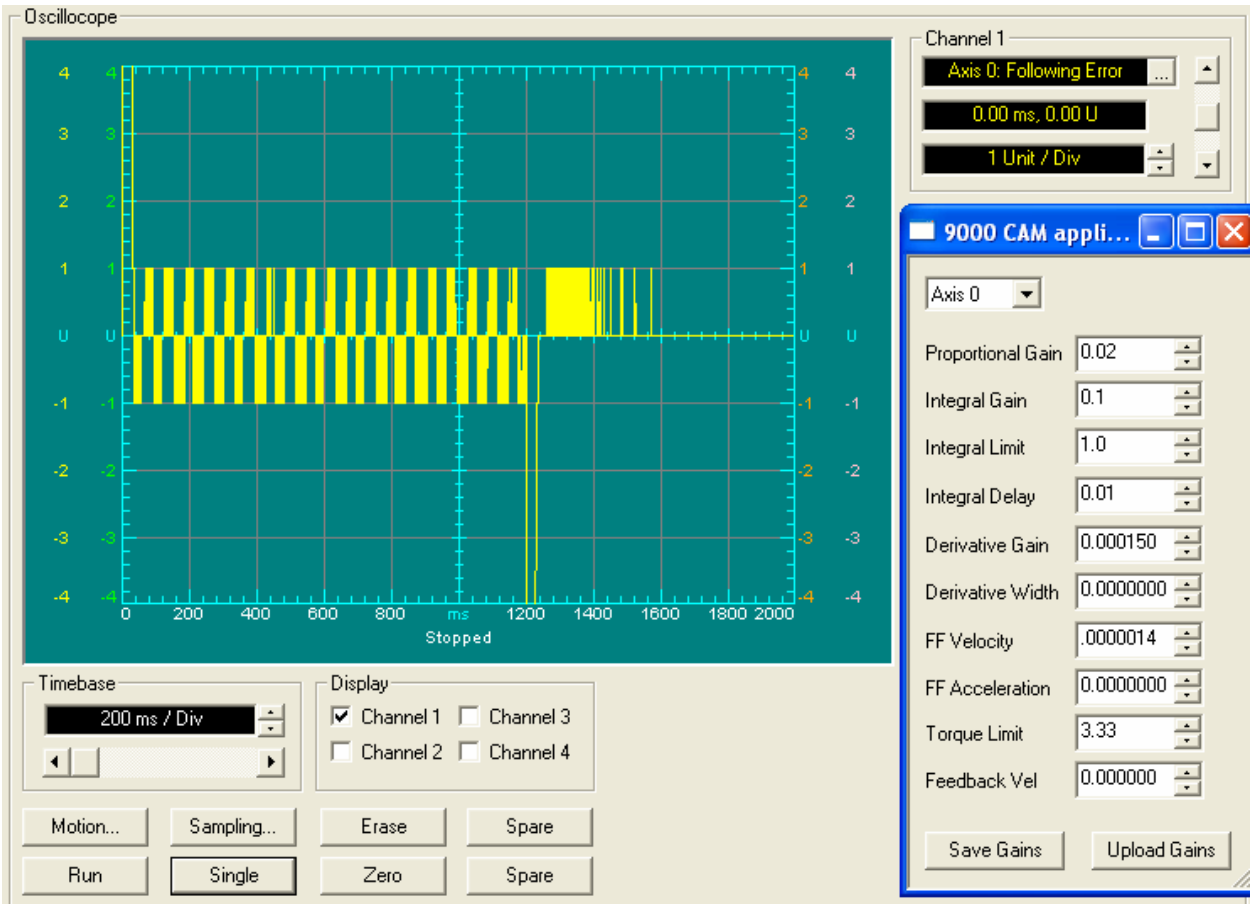
Run a trace with FFVel set to zero and note the constant velocity following error in pulses and also note the velocity in pulses per second.

Then apply the formula to determine the correct FFVel number.

In our example from waveform shown on the previous page:

(0.02 V/pulse x 2.5 pulses error / (500 RPM / 60 sec per minute * 4000 ppr) = 0.0000015

Enter your calculated value and then acquire another trace and adjust as required.

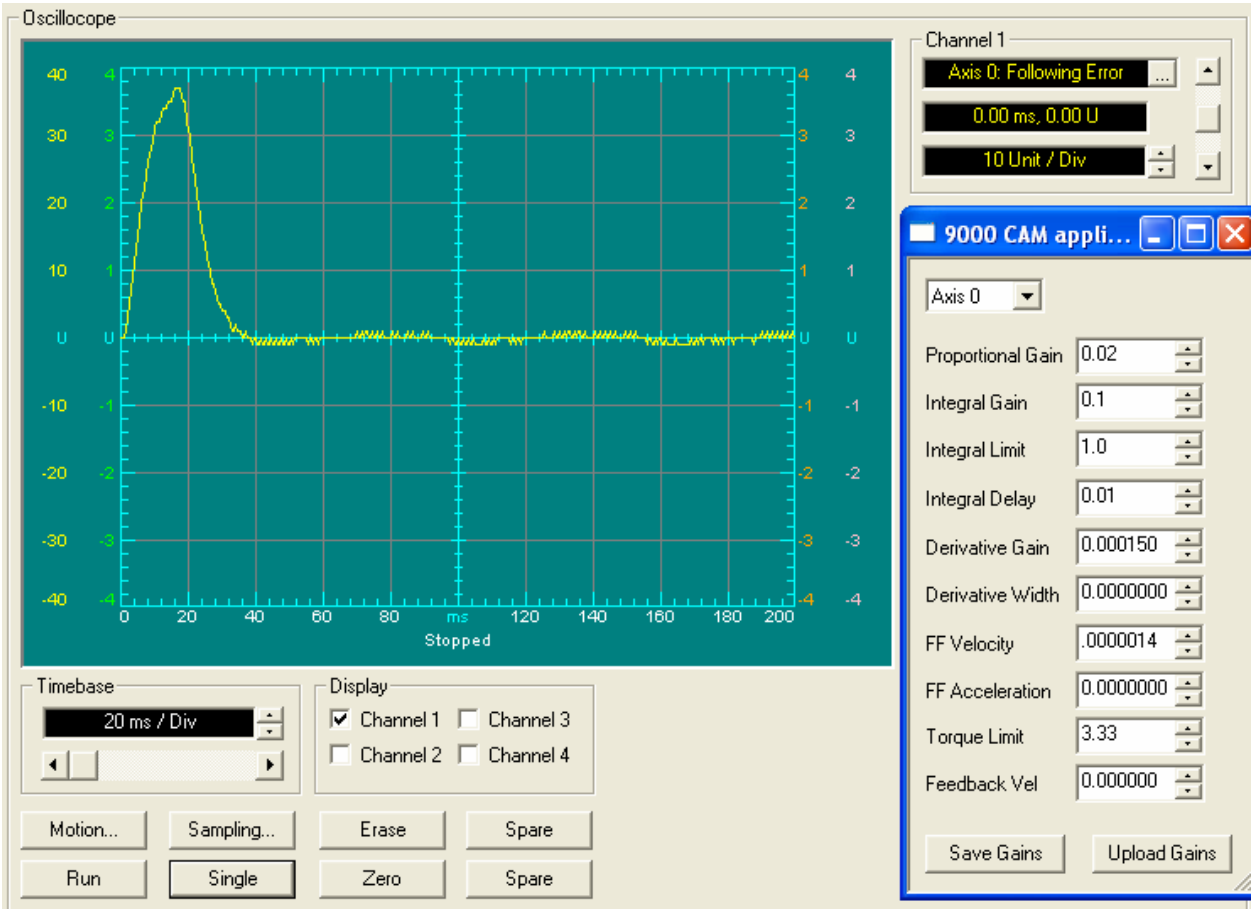


The calculated FFVEL setting was slightly high. It brought the position error down to +/-1 count, but the majority of the time was in the negative region. I tweaked the setting until it was 50/50 for the error being positive vs. negative. Note that this now looks pretty good except for the ACCEL and DECEL portions

Now, we can tackle the ACCEL and DECEL portions. Just as the FFVel value can be calculated from a formula, so can the FFACC setting. Run a trace with FFACC set to zero and note the Acceleration phase following error in pulses and also note the acceleration in pulses per second squared. Then apply the following formula to come up with the correct FFVel number.

FFACC = (Pgain in Volts per pulse)* (Following Error In Pulses) / (Acceleration In Pulses/Second-Squared)

Reducing the Timebase to 20ms/div and increasing the Channel 1 scaling to 10 Units/div we can get a clearer picture of the following error during acceleration.

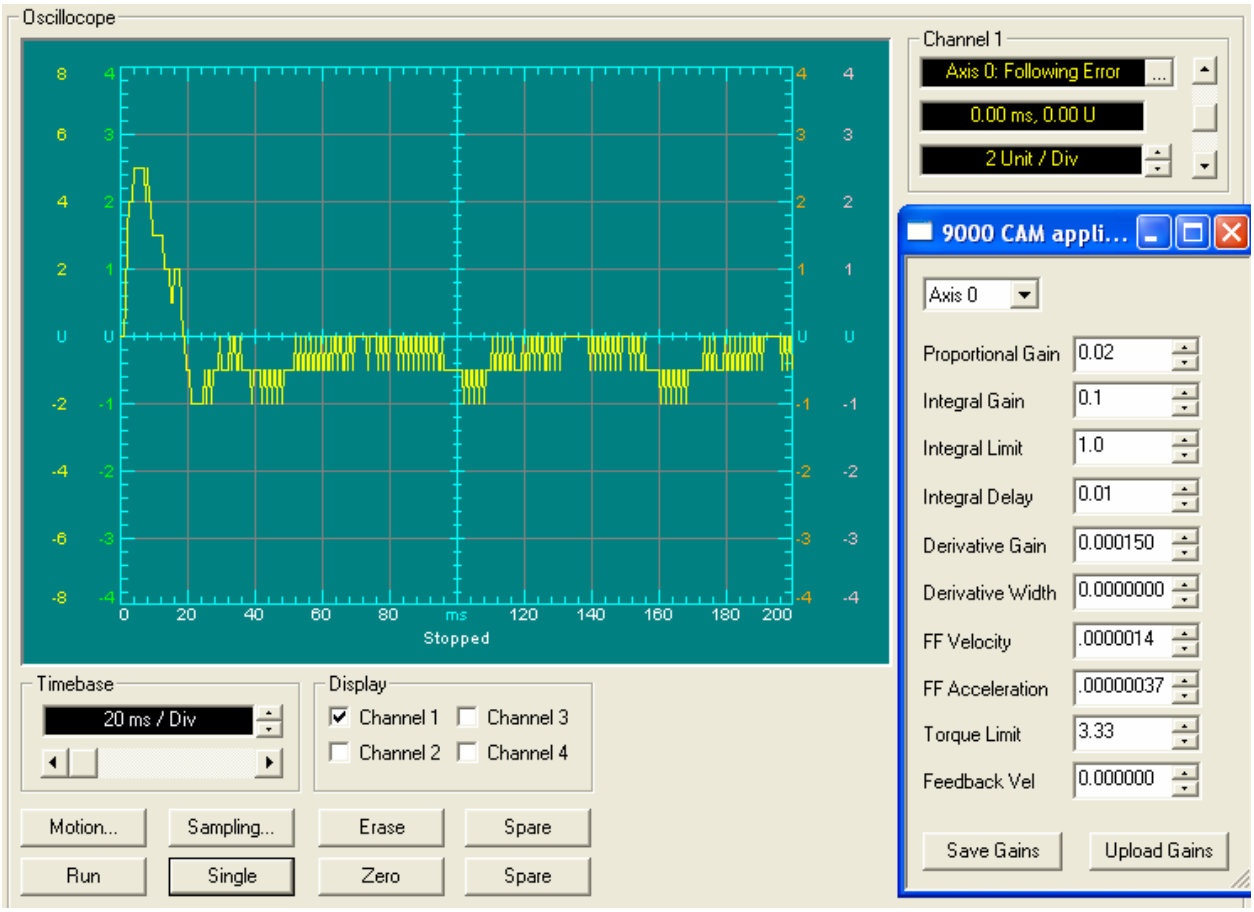


In our example: (0.02 V/pulse x 37 pulses error / (500 rps^2 * 4000 ppr) = 0.00000037 (or 3.7 x 10^{-7})

At this stage the following gain parameters are active:

PGAIN	0.02
IGAIN	0.1
ILIMIT	1
IDELAY	0.01
DGAIN	0.00015
DWIDTH	0
FFVEL	0.0000014
FFACC	0.00000037
TLM	3.33

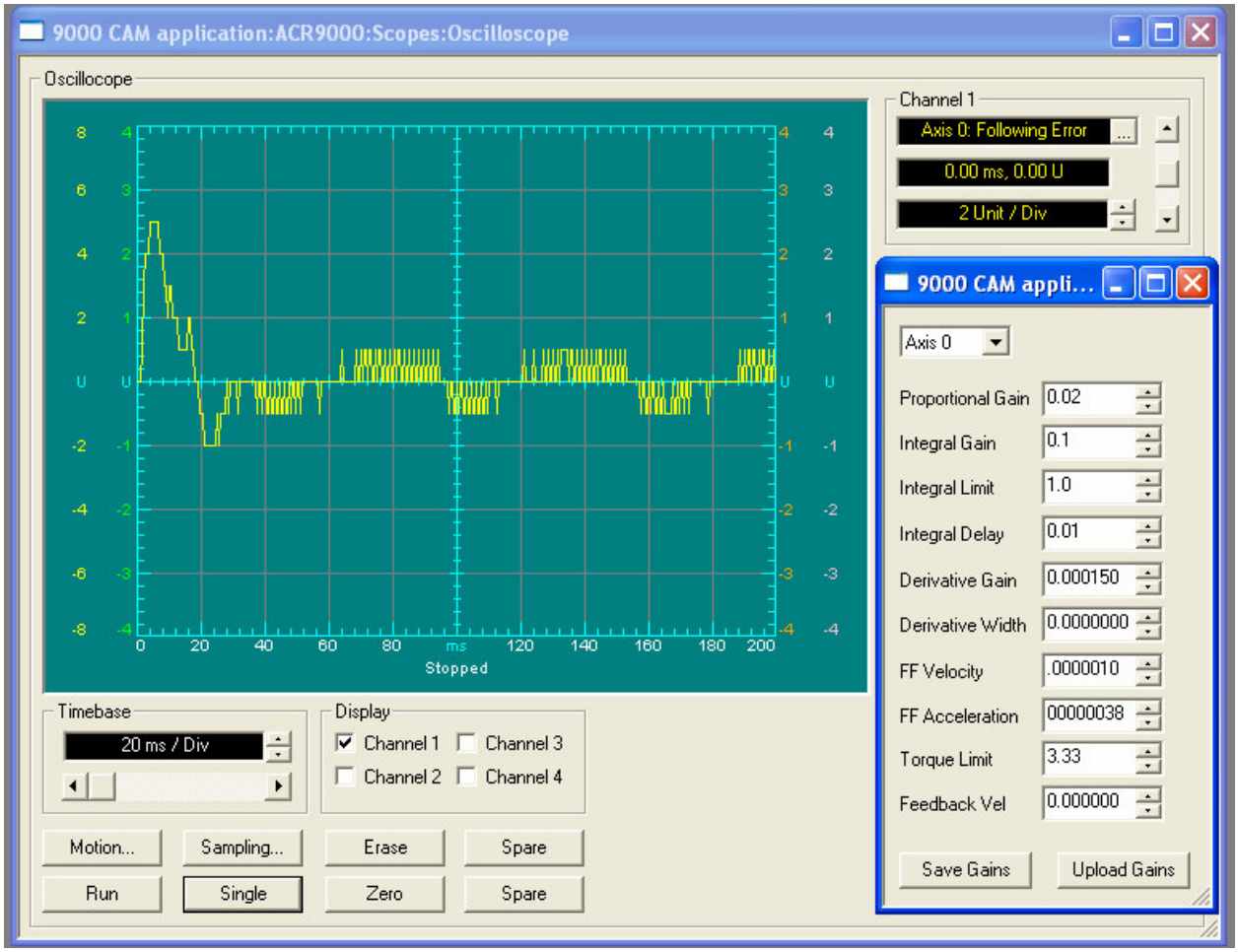
Changing the Units/Div setting to 2, we can get a very clear picture of how small our following error has become.



Note that this did improve the ACCEL and DECEL portions. There is still room for improvement.

It appears that our FFVEL needs a little adjustment.

After additional tweaking of the FFACC and FFVEL values the following trace was captured.



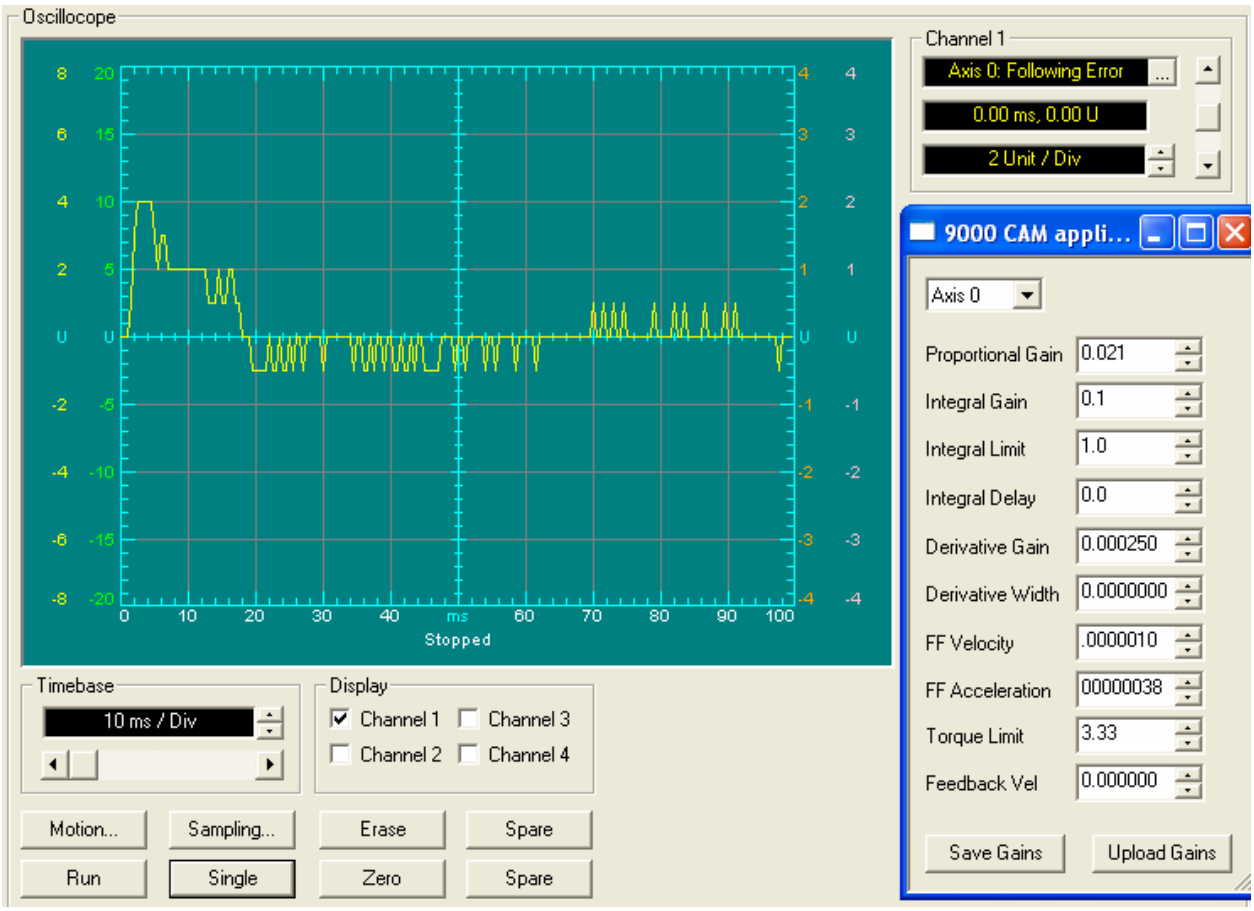
Note that the spikes are now gone and overall trace is looking very good. We only have a maximum of 5 counts of error on acceleration. The following gain parameters are active at this time.

PGAIN	0.02
IGAIN	0.1
ILIMIT	1
IDELAY	0.01
DGAIN	0.00015
DWIDTH	0
FFVEL	0.0000010
FFACC	0.00000038
TLM	3.33

We can still tweak the DGAIN and the PGAIN up and down a little to see if we can improve the trace still further.

If the Acceleration Error spike does not respond to FFACC changes, it may be that the Torque Limit (TLM) is limiting our torque output on the motor. You can determine this by selecting Channel 2 in the Display area and then selecting OBJECTPARAMETERS => DAC PARAMETERS => DAC OUTPUT => AXIS0 for Channel 2 setup. If the DAC ouput reaches 3.33V, then increase the Torque Limit setting and try again. If DAC reaches 10V then the motor is undersized or the amplifier is not set up properly.

Lastly, we will adjust IGAIN and enable it to work at all times. Setting IDELAY to zero enables the IGAIN to be active at all times, including during the move. The timebase has changed to 10ms/Div.



No drastic improvement but we will leave it here, as the higher gain will combat load disturbances better than the lower value as long as the motor is not buzzing or unstable.

Setting the IDELAY to zero allows the IGAIN to work at all times instead of after the Profile move stops. This helps take care of continuous load changes affecting the stability of the system.

To Filter or not to filter

On top of the main nine main parameters, we still have two separate 2nd Order BI-QUAD FILTERS that have not been used yet. These can be utilized to either filter out a band of undesirable frequency from the DAC output or implement a LOPASS filter to reduce the bandwidth of the DAC output.

Servo amplifiers from different manufacturers vary tremendously in their bandwidth. Some of the analog servo amplifiers might have a bandwidth as high as 500Khz while the digital ones might be as low as a few hundred Hz. All of this with the machine mechanics might have the bandwidth substantially lower than 100 Hz. As an example, a motor running a Disk drive in a PC might have a bandwidth on the order of 500Hz.

Depending on the above factors, one can implement a lopass filter at a 1000Hz to reduce some sharp peaks going to the amplifier. Most cases will not need the filter at all. In the case where a NOTCH filter is needed or a LOPASS filter is needed, the following commands can be implemented.

LOPASS X1000 :REM This sets up one of the BiQuads as a LOPASS filter for X Axis with 1000Hz 3db rollover

.

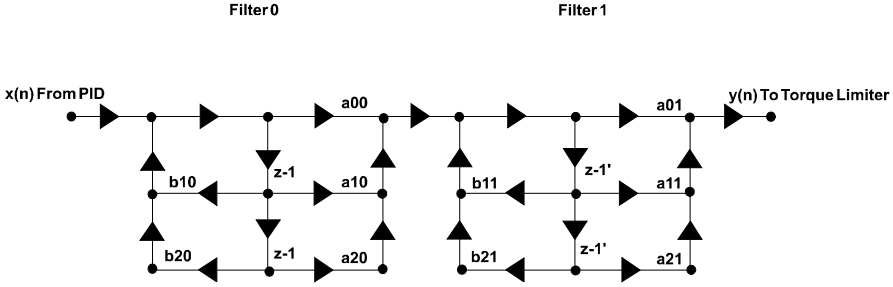
NOTCH X(500, 50) REM This sets up one of the BiQuads as a NOTCH filter at 500Hz with 50Hz bandwidth.

The necessity of the above entirely depends on each application. The scope display will let the user know if either of these are helping or hurting the tuning effort. Playing with these must be the last step in the tuning process. The user can also not even use the LOPASS and the NOTCH command and choose to load the POLES and ZERO information directly into the CO-EFFECIENTS of these cascaded filters directly.

Filter Mathematics.

There are two separate filters that are cascaded and inserted between the Output of the PID filter and the input of the Torque Limiter (TLM). When the filters are not in use, the output of the PID is fed directly into the input of the Torque Limiter.

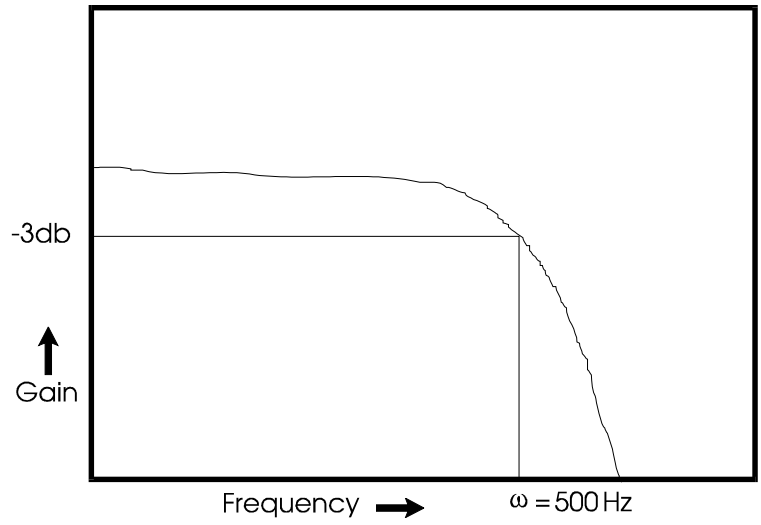
The bottom line reason for the two filters is to suppress certain range of frequencies from showing up at the final DAC output. The NOTCH filter suppresses frequencies between a LOWER threshold and a HIGHER threshold. The LOPASS filter suppresses all frequencies higher than the threshold frequency. Here the word suppression is used loosely as the frequencies are not simply cut off but rather the strength of the frequencies being suppressed is greatly attenuated. In the case where the machine has a certain frequency range that sets up oscillations, the control can ensure that it "skips" that range in its DAC output so as not to cause the machine to go unstable.



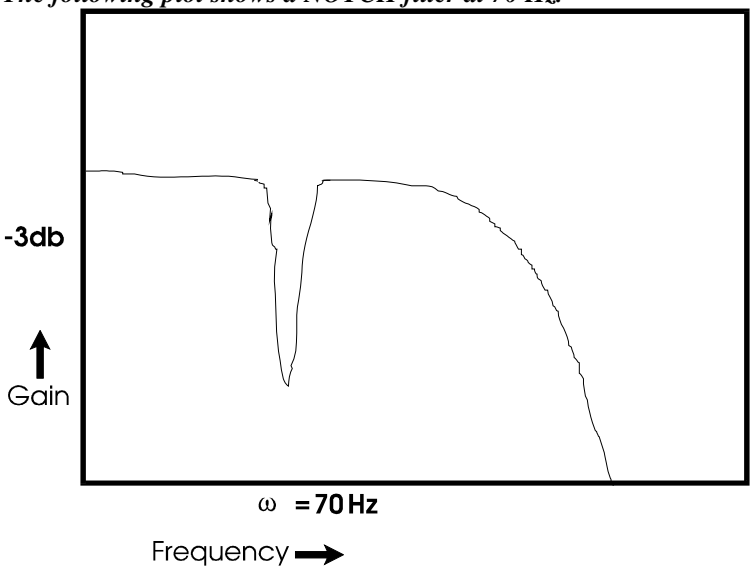
The above two filters have the following equation in the Z Domain.

$$H(z) = \left[\frac{a00 + a10\,z^{-1} + a20\,z^{-2}}{1 - b10\,z^{-1} - b20\,z^{-2}}\right]\left[\frac{a01 + a11\,z^{-1} + a21\,z^{-2}}{1 - b11\,z^{-1} - b21\,z^{-2}}\right]$$

In the current example, the servo period was running at 2500 Hz. Any filter frequencies approaching the servo period frequency will be useless. Trying frequencies in the range of 400Hz to 1000Hz it was determined that the performance was not improving so the filter was disabled. The following plot show a lopass filter at 500 Hz

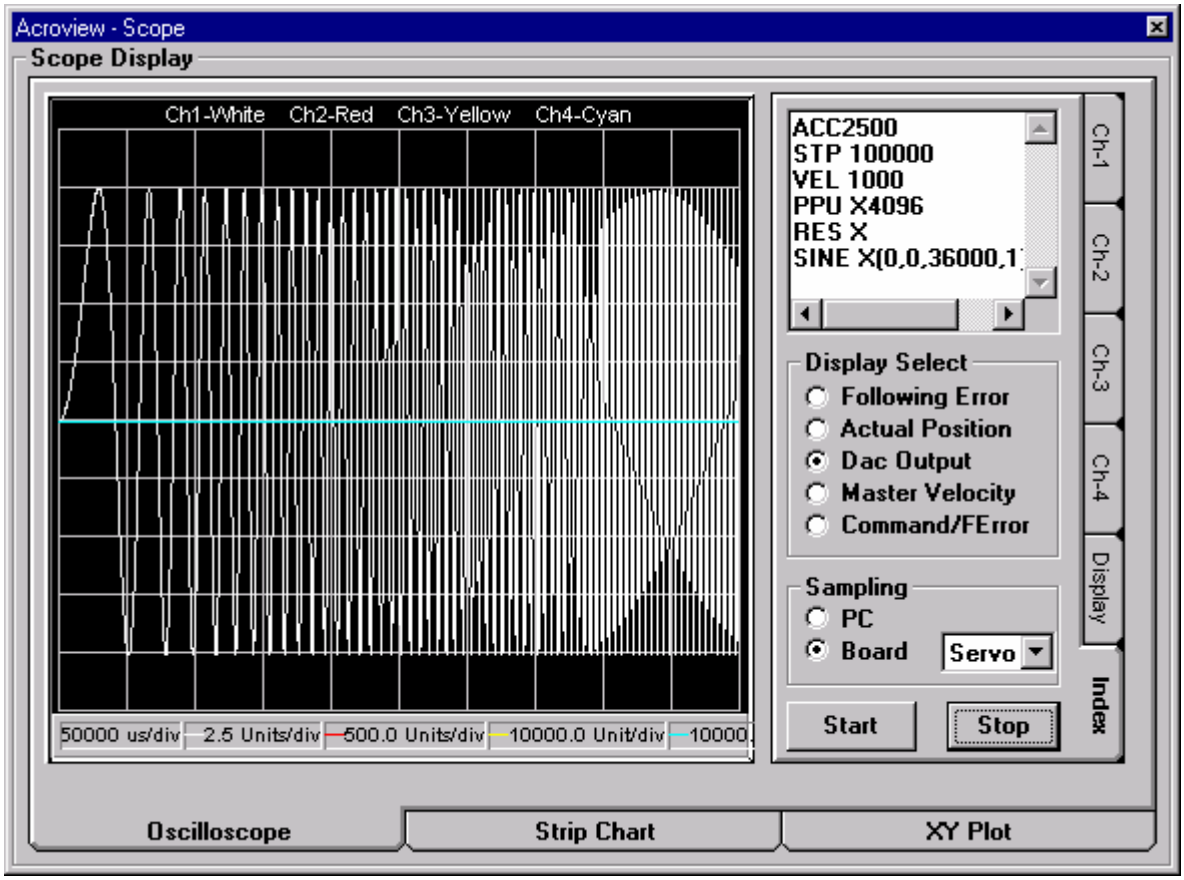


The following plot shows a NOTCH filter at 70 Hz.



In all probability, the tuning for this example was done on a test bed and therefore the motor was running under no load conditions. Any potential machine harmonics and other subtleties that would require us to use one or both filters were not present.

As an example the controller will generate a pure sine wave pattern. The DAC output will be sampled and displayed in the Scope. For this test, the motor/amplifier will be disconnected. The Sine wave patter is generated by commanding the SINE command with the final velocity set to 1000Hz and the accel set to 2500cps per second and a total move distance of 100 revolutions (36000 degrees). In reality, this will result in the SINE waves starting at very low frequency and accelerating to 1KHz in about 400 msecs. So on the Scope display in Acroview, the left-hand side of the trace will represent low frequencies and the right side will show frequencies closing in towards 1Khz. On purpose the amplitude of the sine wave will be kept small enough as not to cause the DAC to go into saturation. This will allow us to view the SINE wave with a +/-10 volt swing. The following trace shows the raw unfiltered DAC output. The DAC channel is CH2 and is setup for showing 2.5volts per div. DAC gain is set so that 1 REV of following error gives 10 volts (PGAIN = 0.00244). With the motor disabled and the PPU set for 4096 pulses (1/2 of 16 bit DAC scale), Sine wave of 1 Unit will give us 10 volts on the DAC.

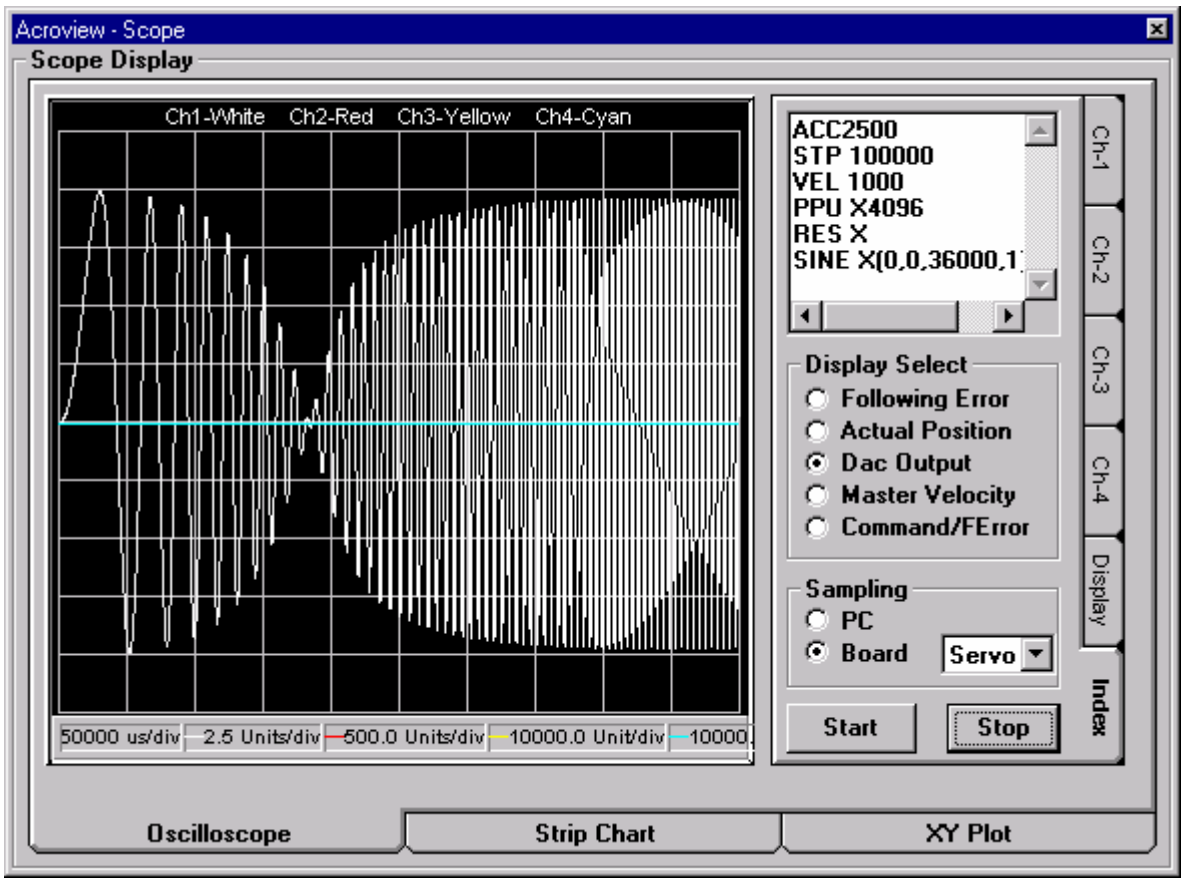


Then first we will observe the effects of a LOPASS filter on the DAC output and then the effect of the NOTCH filter. The period for the controller will be changed to 5000 Hz. Then the NOTCH filter will be introduced.

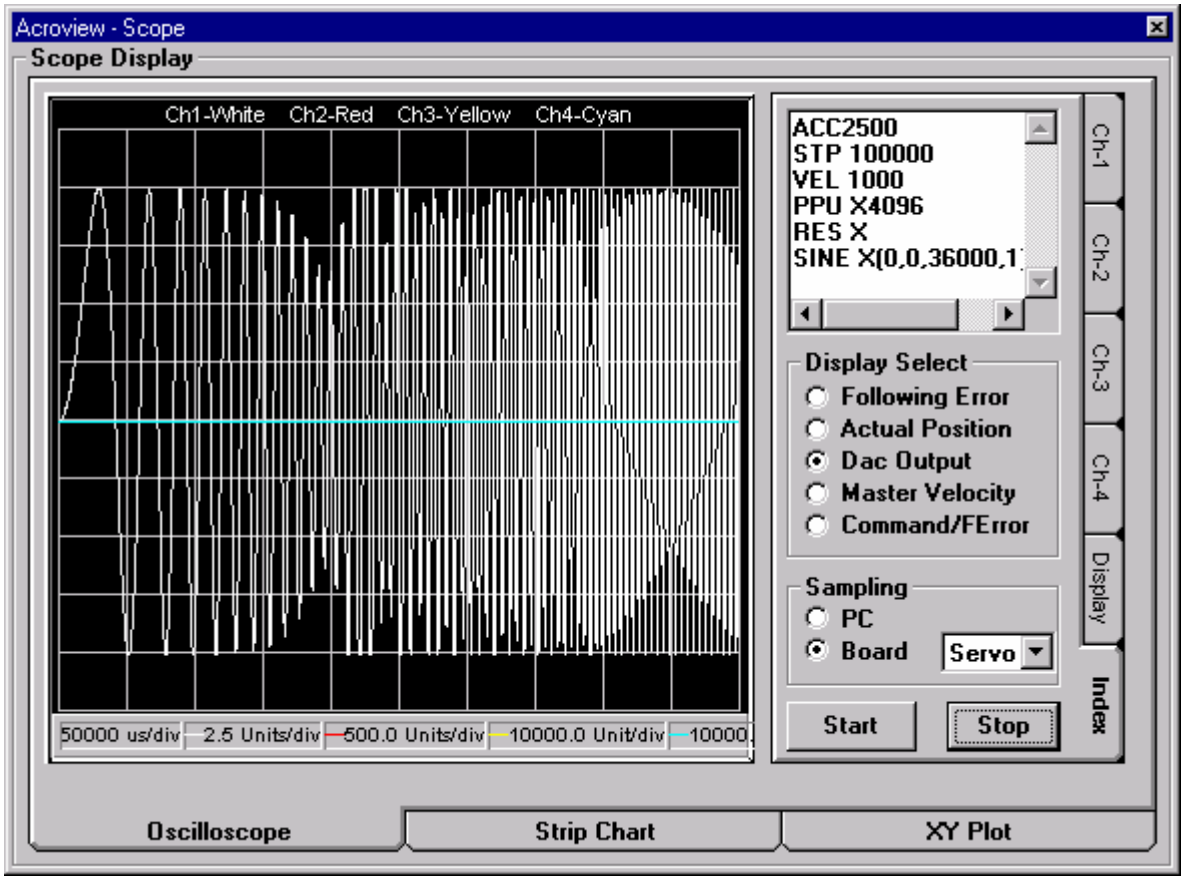
NOTCH FILTERING.

The NOTCH filter has two parameters. The first is the NOTCH frequency. This is the frequency that causes the particular mechanics to oscillate uncontrollably. This is the frequency we have to suppress from the DAC. The second parameter is the bandwidth of the notch. The larger the bandwidth the higher the attenuation that the filter will offer. On the flip side, the larger the bandwidth, the more range of frequencies that will get killed uselessly. The idea is to only reduce or attenuate the signal as much as needed to eliminate the excitation of the mechanics of the system. If the NOTCH width is too large the entire system will become unstable and unusable. The width of the NOTCH can be specified as low as 1 Hertz. But at this value, the attenuation might be very little. If the width is made as wide as the NOTCH center frequency, the DAC output will completely go to ZERO at the undesirable frequency. The user should start at the low value of 1 and work upwards until the problem gets fixed. The following trace shows the effect of calling up a NOTCH filter at 100Hz with the NOTCH width of 100 Hz. The Servo Update was setup at 5000Hz. The following command was used

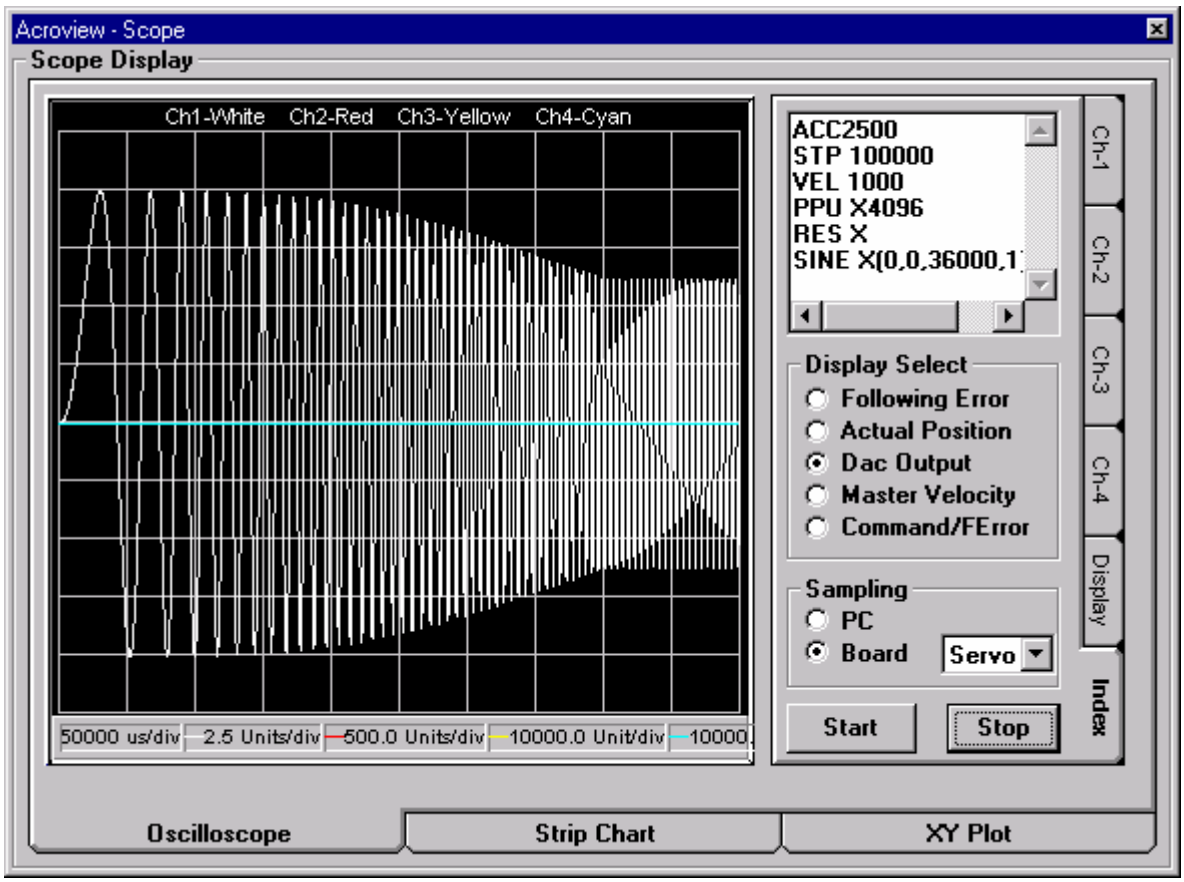
NOTCH X(100,100)



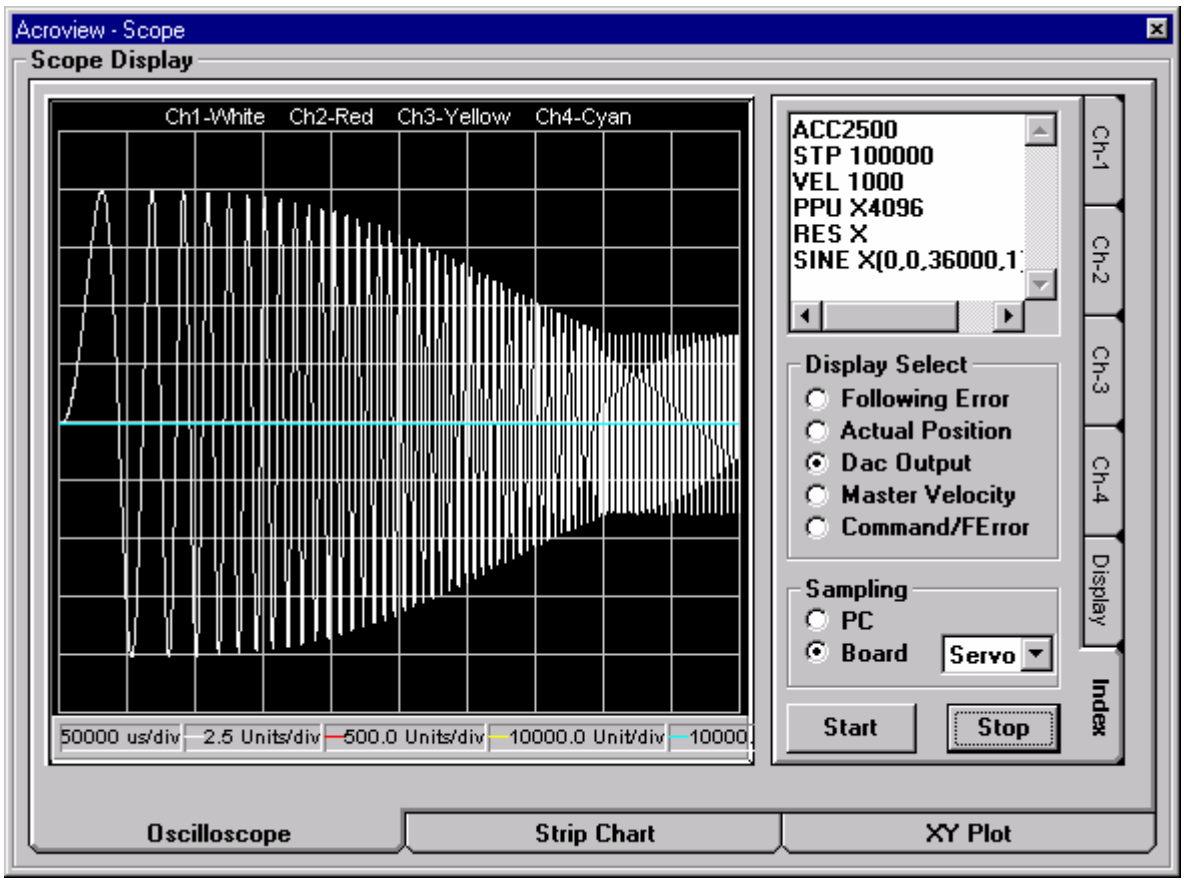
Notice that the DAC signal goes to ZERO at the NOTCH frequency. This is shown purely for the sake of understanding the command, as there is such a large gap in the output frequency that the machine would not be stable with such a large notch. The next trace shows a more reasonable choice of NOTCH width.



Here, the signal dips slightly at the undesirable frequency and comes right back up within a few cycles. The key is to determine whether the amount of attenuation is adequate to suppress the harmonics in the mechanics. This can be achieved via some experimenting. Usually most methods of excitation of the mechanics can tell the system designer not only what frequencies must be suppressed but also how small or large the bandwidth of such frequencies is.
Next we will cover the LOPASS filter. Unlike the NOTCH filter which needs careful investigation, the low pass filter frequency can be chosen such that it is well beyond the calculated bandwidth of the mechanics. The low pass frequency must not exceed 10% of the servo update frequency. The following trace shows a lopass filter set for 200Hz with the Servo Update set at 5000Hz. The following command was used.
LOPASS X200



In the above case the lopass filter can be seen to be attenuating all frequencies starting at 200 Hz. Note that both the LOPASS command and the NOTCH command each use ONE bi-quad filter. If the NOTCH filter is not being used, adding a second lopass filter by duplicating the filter parameters from FILTER0 to FILTER1 can increase the slope of the LOPASS filter. The following trace was obtained by doing just that.



This was achieved by copying the filter0 parameters into filter1 parameters.
P12341=P12336
P12342=P12337
P12343=P12338
P12344=P12339
P12345=P12340

Note that the attenuation is now sharper than the previous trace.
The user is free to design any type of filter by loading values into the filter co-efficients if so desired. The "BIQUAD FILTER ACTIVATE" bit flag controls the usage of the filters. The user must first disable the bit. Then change the filter parameters as needed and reactivate the filter again.

Tuning with DUAL FEEDBACK encoder arrangement.

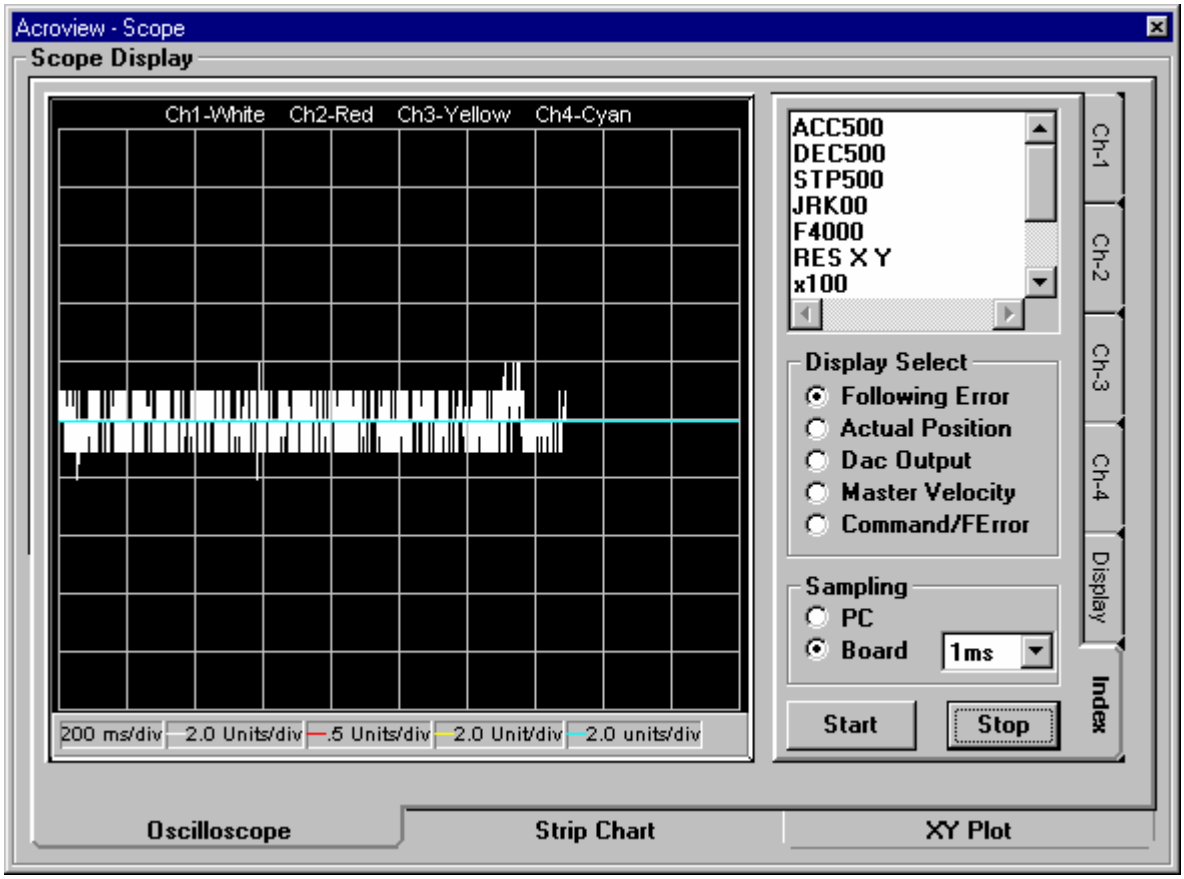
In some cases the user will have two encoders for each axis. The purpose of the two encoder feedback is to allow the position feedback to be taken from the encoder mounted directly on the load. A second encoder is then mounted on the motor shaft. Doing this is entirely optional. The pros and con can be argued either way. In any case when there is a separate encoder feedback specifically mounted on the motor and is different from the position feedback encoder mounted on the load, the FBVel allows the user to inject the motor velocity signal into the SUMMATION point that is fed into the dual bi-quad Digital filter stage. The signal can be thought of as a 'software tachometer' signal and is subtracted from the output of the PID loop. This is shown in the ACROVIEW-SERVO LOOP screen.

The FBVel parameter dampens the servo loop. It is to be used in addition to the the normal PID parameters not instead of them. The optimization of this parameter must be done by viewing the SCOPE trace during the tuning process. Note that this parameter can not be changed from the GAIN screen in the current version of Acroview. Therefore it can be either changed via the terminal or directly from the Index Text Box on the SCOPE screen.

Be aware that because the current Acroview version does not allow changing this parameter from the GAIN screen, it will not be saved automatically in the user project. Therefore, this parameter must be specially loaded with the desired value for all applicable axes in the user program(s).

Wrap Up.

This completes the TUNING effort. These parameters will serve for the entire range of RPM on this motor as long as the motor has enough power to push the load. The gain settings will work for all RPM and ACC and DEC ranges. For example the following trace was done for the same motor but at 4000 RPM and 100 REVS. The Horizontal scale was changed to 200msec per division to get a better picture.



Note that even though we tuned at a much slower speed, the gain parameters are flawlessly performing even at higher RPM's. Some might choose to tune at the fastest application speed and the fastest application acceleration but there are issues of safety that get magnified at high speed. If one does the tuning properly at the fastest possible required acceleration and moderate speed, the parameters should hold true for the entire range of speeds assuming that the motor/amplifier combination have enough torque to drive the load at the required speed and acceleration.
With the above settings even if the motor sees on the fly load changes, the following error should stay fairly close to zero.