

# Stage Accuracy and Settling Time for ASI Stages

The reproducibility and accuracy of a servo stage usually hinges on its behavior while seeking its final position. The stage may be required to come to rest within one to a few encoder counts of the desired location. Mechanical effects including vibration, inertia, and friction can frustrate efforts to land precisely. Careful tuning of the servo feedback parameters can help the system to quickly settle on target, but it usually takes longer to land exactly on target than if we allow a bit of “finishing error”. (Here we ignore the accuracy of the encoder itself, which is higher for linear encoders than for rotary encoders as described [elsewhere](#).)

ASI’s servo controllers have two conceptually-related settings for landing precision. The “finish error” (“[PCROS](#)” or “[PC](#)”) is the allowable deviation from target position before the motors are turned off and the move deemed to be complete. The “drift error” (“[ERROR](#)” or “[E](#)”) is the post-landing error allowed before the motors are re-energized to once again bring the stage to target. By default the “drift error” is set to a value significantly larger than the “finish error” to provide sufficient hysteresis so that the stage will not constantly be attempting to re-adjust its position.

One common task is to make a series of moves as fast as possible, e.g. to traverse an XY array and snap one image at each XY position while the stage is stationary. For such situations it is highly recommended to use TTL signaling to avoid the time to communicate with the host PC (the time to send a serial command is usually relatively short, but the detecting when a move is complete using status polling can add significant time). Specifically, TTL pulses can be used to initiate moves and the controller can output a TTL pulse when moves are completed. See documentation for the [TTL command](#).

The basic move profile is set by the acceleration time and the maximum speed. For long moves, there is a ramp phase where the target speed increases linearly to reach the maximum speed  $S$  after the acceleration time  $AC$ . Then the target speed remains constant at the maximum speed until the ramp down phase, during which the target speed decreases linearly to 0 over the acceleration time  $AC$ . For short moves – specifically for moves shorter than the product of settings ( $AC * S$ ) – the maximum speed is never reached, the ramps are truncated, and the target velocity profile looks triangular. A final component of the total move time is how long the servos take to land or settle within the finish error when in nears the target position. This settling time is the only aspect of the move time that is not directly controlled by a firmware setting, but it is affected by how well the stage is tuned via various firmware settings as described below.

## Small Moves

The  $AC$  value is the duration of the ramp up (and ramp down) assuming that you reach full speed. The product of settings ( $AC * S$ ) represents the distance traversed while ramping up to maximum speed and then ramping back down; and for any move shorter than this product the maximum speed will never be reached. The units for  $AC$  are milliseconds and  $S$  in millimeters per second, so their product has units of micrometers.

The actual ramp up/down distances (including internal truncation and rounding) can be inferred from the output of the (“[INFO](#)” or “[I](#)”) command: look under “Ramp Length” and then divide by the encoder

counts per millimeter (the “Ramp Length” is the distance covered by both the ramp up and the ramp down).

For example, with default 4 TPI stage with  $S=5$  [mm/s] and ramp time  $AC=100$  [ms], the distance covered during the ramps is  $5 \times 100 = 500$   $\mu\text{m}$ . Another way of saying this is that for moves shorter than 500  $\mu\text{m}$  the maximum speed will never be reached. For a move of 500  $\mu\text{m}$  the elapsed time would be 200 milliseconds (100 ms ramp up and then another for ramp down) plus any settling time at the end.

For move distances shorter than the  $AC \times S$  product, the ramps will be truncated and the maximum speed will never be reached. In this situation, the time taken for ramping up both up and down in milliseconds is  $2 * \sqrt{[\text{total travel in } \mu\text{m}] * [AC \text{ in ms}] / [S \text{ in mm/s}]}$ . For example, for a 5  $\mu\text{m}$  move with  $AC=25\text{ms}$  and  $S=1.25\text{mm/s}$ , the total ramp up/down time is  $2 * \sqrt{5 \times 25 / 1.25} = 20$  ms, or 10ms up and 10ms down, and the motor will only reach  $10\text{ms} / 25\text{ms} = 40\%$  of the maximum speed (0.5mm/s) at the peak of its truncated ramp. For a 1  $\mu\text{m}$  move we expect the total ramp up/down time to be 9ms.

When dealing with move times significantly less than 100ms take the calculated numbers with a grain of salt: the intrinsic time constant of the motor (how long it takes to respond to an electrical input) is roughly 7ms.

For move distances longer than the  $AC \times S$  product, the minimum move time for a move distance  $D$  is  $2 \times AC + (D - AC \times S) / S$ . This assumes zero settling time and no backlash move (note that with rotary encoders the backlash move is required for good bidirectional repeatability).

## Tuning Stages to Minimize Move Time

After specifying basic settings including the speed, acceleration time, and finish error (PC setting, AKA “finish error”), the overall goal of tuning is to minimize the settling time by eliminating any hunting that might occur to land at the target within the allowed finish error. Fast settling is most likely when the hardware and motor constants do most of the heavy lifting, with feedback keeping things on track, instead of the feedback doing most of the heavy lifting. Until early 2020 we would occasionally recommend PID settings to customers, but we would only do Step 4 below and we were leaving a lot of potential performance on the table by not tuning the motor constants.

Here is the recommended algorithm as of September 2020. Make sure to load/mount the stage as it will be used before beginning.

### 1. Hardware tuning **AA** and **AZ**.

1. Increase AA until the stage starts to go haywire (oscillations and trouble landing), then decrease by a couple. The final AA value varies slightly depending on stage and card (maybe load too?) but is often in the low 90s; see [this page](#) for details about AA adjustment. Don't forget to send **AZ** <axis> every time you change the axis' AA setting. This ideally is done running relatively “cold” because the AA threshold where things go haywire increases slightly with temperature (including self-heating when you have been using the stage). In the long run we hope to have the AA optimization be able to be done by a one-time serial command just like the AZ is today. If you ever observe stage oscillations you know to reduce AA.

### 2. Move parameters **AC**, **PC**, **S**, and **B**. Decide on and set values for AC and PC and S and B.

1. Acceleration setting – AC – determines the duration of motor ramp up/down for moves

where the max speed is reached. It is a tradeoff between duration of the ramp speed and wear of the motor gearbox. For typical stages and loads 25ms is a very aggressive setting, 50ms is moderate, and the default is usually 70ms or 100ms. Longer should be used for larger stages and/or with heavy loads.

2. PC is the allowable deviation from the target position to consider the move complete, so if you have a wide tolerance you should specify that now (by default the PC setting is just over 1 encoder count which is smaller than most applications require). Note that landing errors are not accumulated with repeated relative moves because the firmware uses the last target position when initiating a relative move instead of the actual position (see [R](#) command). Note that this setting is specified in millimeters.
3. The maximum speed S can usually be set to at least 90% of the allowed speed (default is 67%), but the maximum possible in practice depends on the stage's load as well as leadscrew pitch.
4. The backlash value B can be set to 0 with linear encoders and for applications where the motion is repeated moves in the same direction, or in applications where repeatability is only required when repeating the same move pattern over and over. A non-zero value means there will be a small final move initiated automatically by the controller ensures the target is approached from the same direction every time, which is a highly effective way to minimize the effect of mechanical lash when the initial position is an unknown direction from the target position. The final move takes non-zero time, even if it is in the same direction as the main move (because the ramp up/down still happens).
3. **Motor parameters KV and KA.** Experiment first to find the best KV and then the best KA. During this step set the feedback parameters KP and KI and KD to 0, and also set the backlash parameter B to 0 (i.e. you are simply seeing how close you can get in “open loop” without PID feedback). To find the best KV, set KA=0 and execute long moves (e.g. 2mm for 4 TPI), see how close you land, and adjust KV (if the move comes up short then increase KV and vice versa). Usually the default KV is within 10% of optimum. Check moves in both directions; there can be asymmetry because the quantized nature of AZ adjustment leaves a bit of residual imbalance. Once you have found the best KV, leave it set and do a similar process while varying KA, but instead of making long moves make short ones (e.g. 100um for 4 TPI) and seeing how accurately you land. The ideal KA value is usually quite low if you have AA tuned well (KA helps compensate for non-optimal AA, so default KA may be too high once you optimize AA).
4. **Feedback parameters KP and KI (and KD if required).** After optimizing the motor parameters, add back non-zero values for feedback parameters KP and KI (KD is almost always 0), as well as for backlash B if a backlash move is desired. The goal here is to find the KP and KI that are large enough to land quickly and accurately but not be so large that the stage overshoots and has to “hunt”. The recommended method is to view the time to complete the move over many trials; an excellent way to do this is to use a repeating TTL trigger to initiate the move and then watch the output TTL trigger on move completion and observe the total move time on an oscilloscope, e.g. give a relative move and then use TTL X=2 Y=2; letting the oscilloscope average or turning on persistence will help see trends over many moves. Long move times mean that hunting is happening. Default KP and KI are a good starting point but you may need to decrease them after the tuning steps above. Joystick moves use the feedback parameters but not the motor parameters, so if you care about moving with a joystick then double-check that joystick moves are still possible after changing the feedback constants.
5. **Save the settings using SS Z command.** Note that for Tiger you need to prepend the card address because it is a card-addressed command unlike all the others mentioned here. It may also be worth making note of the settings above in case the firmware ever needs to be updated; they are all displayed if you use the [INFO](#) command. All of the above settings except the AA will be lost if firmware needs to be reflashed or when the SS X command is issued.

## Empirical Observations Tuning a 4 TPI LE Stage

We still have plenty to learn, but here are some observations from tuning a 4 TPI stage with linear encoders for 360um moves in late September 2020.

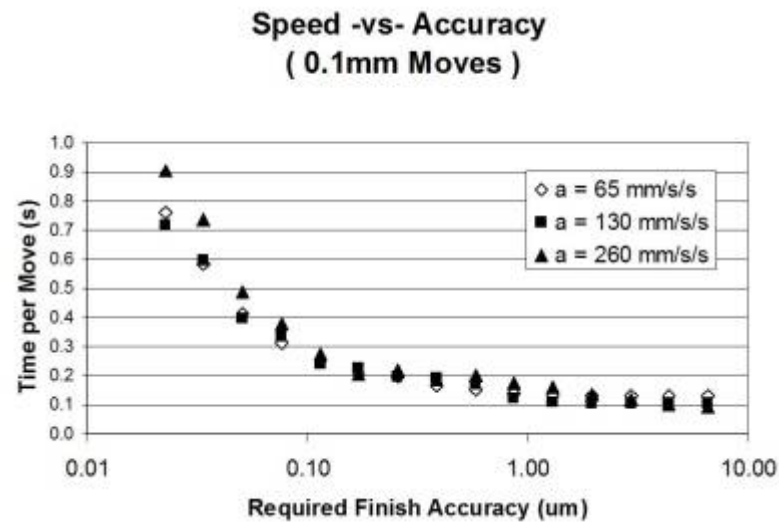
- Relaxing PC to 0.0001, decreasing AC to 30, and boosting S to 7 made a huge difference in average time off the bat. I attribute that mostly to relaxed PC (finish error). By just changing these settings, a good fraction of moves completed quickly but there were still a lot of moves that took almost as long as before (i.e. required significant hunting to land).
- Tighter PC makes hunting more likely and hence increases the move time. Given the next point, tuning is increasingly important if you require a tight tolerance on landing (i.e. small PC).
- Tuning motor constants and feedback constants didn't change the best-case move times, but it makes the best case happen consistently. Whereas hunting was common with the default parameters even after relaxing PC, it practically never happened after tuning.
- Once mostly tuned, reducing KP made hunting less likely but also slightly increased the average move time. To minimize total move time over hundreds of moves then it may be better to increase KP and tolerate occasional hunting.
- (side note: currently the KI quantum is really too big on MS2000 controllers. As of Tiger v3.31 the quantum is reduced by 16-fold but the change has not been applied to MS2000 firmware as of Sep 2020. In the plan is to make the KI and KD constants be given in absolute units instead of units that change depending on how many axes are on the controller (as is the situation today).


## Original Description and Tests



The following section was written ~2004 and apparently did not utilize TTL triggering to remove communication overhead. While the basic principles are unchanged some of the exact numbers may not match ASI's current products.

What does this mean in terms of stage speed and settling time? We tried to answer that question by running a test where we recorded the time to make many short moves using different accuracy and acceleration parameters. We used Image Pro software, which queried the stage for its busy status about every 12 milliseconds. As soon as the busy would clear, the program would issue the next move command. We measured the time to make a whole series of moves, so besides the actual move and settling time there is also the serial communications overhead and the average polling time included in the measurement.



 Click to Enlarge

The chart above shows the results of this test on a typical ASI stage equipped with linear encoders. The encoder resolution was 20nm, so the best accuracy tested required that the stage settle to within one encoder count. Top speed for this stage is about 6.5mm/s but for short moves the stage never gets up to full speed. However, since acceleration is an adjustable parameter, we show move times for three different rates of acceleration. Note that lower values of acceleration reduce the time per move when high accuracy moves are desired. The jerky motion takes longer to settle, but if you don't care as much about accuracy, then the higher acceleration will get the stage to target somewhat sooner.

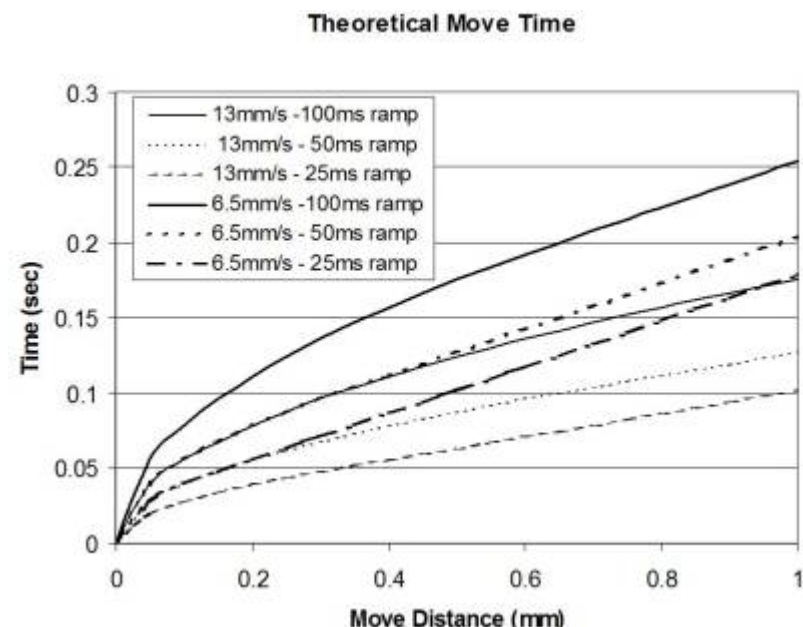
A look at the fastest settling times gives an account of the effect of communications overhead. With allowable error of more than 5 microns, the stage essentially gets to target without having to correct its initial move. The table below shows the observed move time and the calculated time it would take to actually ramp up and down to get to the target for the given acceleration used.

Average Move Time	Acceleration	$t=2\sqrt{d \over a}$	Overhead
129 ms	65 mm/s <sup>2</sup>	78 ms	51 ms
105 ms	130 mm/s <sup>2</sup>	55 ms	50 ms
91 ms	260 mm/s <sup>2</sup>	38 ms	51 ms

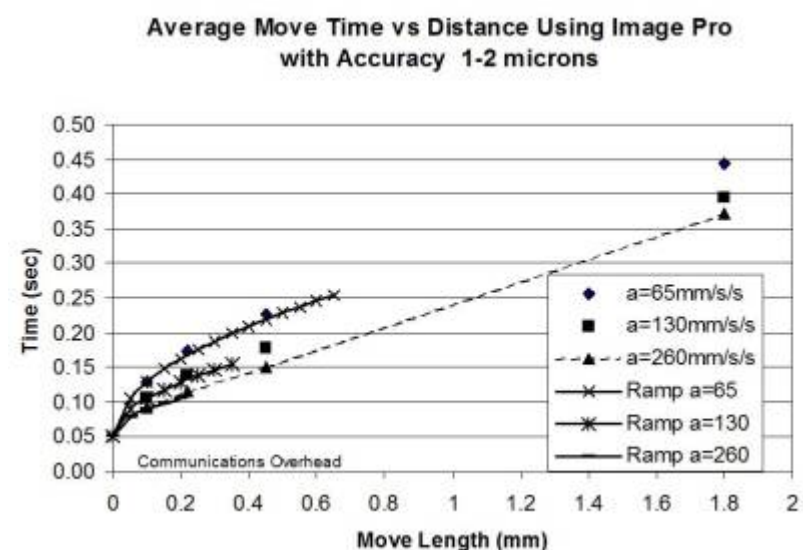
The difference is the amount of overhead left in the control system. We see that when we subtract out the actual time the stage is moving we are left with overhead of about 50ms. Of this 50 ms two major components will be 1) the time it takes to send the move command via the serial interface at 9600 baud, and 2) the average wait time for the serial polling process to detect the not-busy condition once it occurs. Sending the approximately 20 characters for the move command at 9600 baud will take about 20ms (it would be correspondingly faster for 115200 baud). The polling delay is often about 10ms (this depends on the computer operating system), making the total accountable overhead about 30ms. This leaves 20ms unaccounted for, which may be due to a combination of sources including latency in the Image Pro macro program, 2 to 8 ms latency in the MS2000 controller to set up the moves and clear busy flag, and additional computer serial port latency.


A related issue to settling time is how to best choose the lead screw pitch for quick moves. Again, accuracy becomes an issue, since doubling the speed of the stage by using a fast pitch lead screw will in general double the inaccuracy. The rate of acceleration is also a parameter that needs consideration. The motors used in ASI stages are brushed DC motors that will give years of trouble

free use if not abused. Multiple rapid acceleration and de-acceleration are not good for the motor and can lead to failures. A ramp time of 20ms or more is recommended to ensure long motor life. The chart below shows the theoretical time it should take to make sub-millimeter moves for various lead screw and ramp time combinations. The top speed of our standard 6.35mm pitch lead screw stage is about 6.5mm/s. ASI can also supply stages with 12.7mm pitch lead screws with 13mm/s top speed for applications where the added speed is desirable.



 Click to Enlarge



 Click to Enlarge

The chart above shows the measured average move time for different length moves assuming the controller doesn't have to correct for the error on the move. This usually means allowing errors of one to two microns. Using various ramp times from 100ms to 25ms and top stage speed of 6.4mm/s, the time to complete moves is shown as a function of move length. The calculated time for ramp up and down at a given acceleration is also shown. The measured data closely matches the expected value if we include about 50ms of communications overhead that we observed previously.

For systems that need to minimize all sources of latency, the MS2000 may be commanded to move via TTL pulses, with move-completion signaled by TTL pulses from the controller. In this mode, a ring buffer is used to store move locations.



[xystage](#), [ms2000](#), [tiger](#), [tech note](#)

From:

<https://asiimaging.com/docs/> - **Applied Scientific Instrumentation**

Permanent link:

[https://asiimaging.com/docs/stage\\_accuracy\\_settling\\_time\\_for\\_asi\\_stage](https://asiimaging.com/docs/stage_accuracy_settling_time_for_asi_stage)

Last update: **2023/12/12 19:24**

