In this issue of Design Trends:

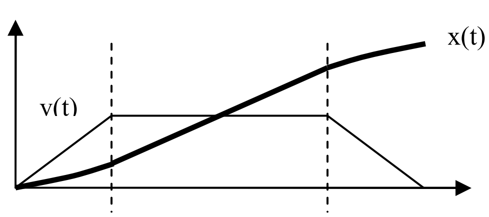
Motion Path Planning With PVT

The large majority of motion trajectories are either trapezoidal or S-curve profiles. In the case of trapezoidal profiles, the velocity is ramped linearly, and in the case of S-curve profiles, the acceleration is ramped linearly. The resulting position profile is 2nd and 3rd order respectively. Although these standard profiles suffice for point-to-point moves, they are not useful for more complex position profiles or distributed coordinated multi-axis motion. This article aims to introduce PVT (Position-Velocity-Time) as an alternative, more flexible method to create motion profiles.

General Position Path Planning

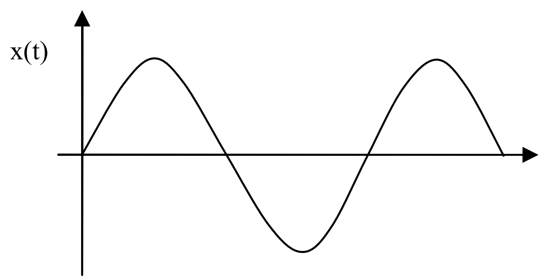
The objective of position path planning (or trajectory generation) is of course to dynamically create a position reference that results in a controlled move from point A to point B. In some cases, it is not important what the actual profile is between A and B, sometimes it is. For example, on a CNC, a move that returns the tool to a home position does not impose specific restrictions on the path that is followed (other then avoiding a collision with the work piece), whereas actual machining requires the tool to follow a very specific path.

In general, the reference position x(t) can be any mathematical formula, continuous or piece-wise continuous. For example, a trapezoidal move is a piece-wise continuous curve:



Note that it is the velocity that has a trapezoidal shape, not the position.

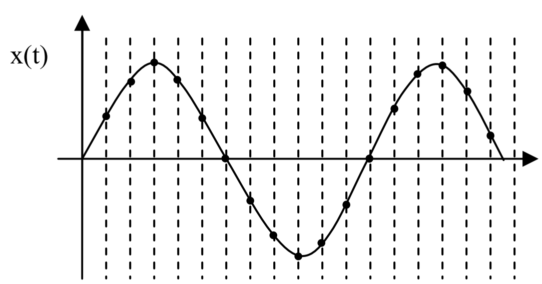
In the case of a circle in an X-Y plane, the position along X is continuous:



For path planning purposes, position profiles are expressed as polynomials. It is always possible to introduce a polynomial of sufficiently high degree to approximate any mathematical profile (Taylor series). For example:

$$x(t) = \sin(\omega t) = \omega t - \frac{(\omega t)^3}{3!} + \frac{(\omega t)^5}{5!} - ...$$

This however gets unwieldy quickly. A better approach is to segment the profile into smaller pieces, and approximate those smaller pieces with a polynomial.



So in a way, the original curve is "sampled" and then reconstructed as well as possible. Determination of the segment time length will be discussed below, and is driven by general sampling theory rules.

A first approach is to use linear approximation, i.e. approach each piece with:

$$x(t) = x(t_i) + v_i * (t - t_i)$$
$$t_i \le t \le t_{i+1}$$

The segment start and end positions determine the coefficient v_i. Of course this means that the velocity (v_i) is constant for each segment piece and not smooth.

Another approach is to use a second order (quadratic approximation):

$$x(t) = x(t_i) + b_i * (t - t_i) + a_i * (t - t_i)^2$$
$$t_i \le t \le t_{i+1}$$

In this case, there are 2 coefficients that need to be determined. One condition will of course be the final position of the segment. For the other condition there is a choice. One can select continuity of the velocity for example. This leads to:

$$b_i = b_{i-1} + 2 * a_{i-1} * h$$
$$a_i = \frac{x(t_{i+1}) - x(t_i) - b_i * h}{h^2}$$

Here h is the segment time $t_{i+1} - t_i$. It is also assumed that the start velocity is zero. The a and b coefficients depend on the previous segment coefficients.

Another choice would be to specify the velocity at the end of the segment, which leads to:

$$a_i = \frac{x_i - x_{i+1} + h * v_{i+1}}{h^2}$$
$$b_i = \frac{2 * (x_{i+1} - x_i)}{h} - v_{i+1}$$

This however leads to discontinuities in velocity (the start velocity of a segment does not match the end velocity of the previous segment).

The next logical approach is a 3rd order polynomial:

$$x(t) = x(t_i) + c_i * (t - t_i) + b_i * (t - t_i)^2$$
$$+ a_i * (t - t_i)^3$$

Now there are 3 coefficients that need to be determined. One condition is again the final position of the segment. The 2 other conditions can be continuity of velocity and acceleration at the segment starting point. Or specification of the start and end velocity of each segment (which also then provides continuity of velocity at each segment extremity).
The first case leads to a similar solution as the 2nd order polynomial. The second case leads to:

$$a_i = -\frac{2 * (x_{i+1} - x_i)}{h^3} + \frac{v_{i+1} + v_i}{h^2}$$
$$b_i = \frac{3 * (x_{i+1} - x_i)}{h^2} - \frac{2 * v_i + v_{i+1}}{h}$$
$$c_i = v_i$$

The advantage of this approach is that the coefficients of a particular segment do not depend on previous segments' coefficients but only on the segment start and end point positions and velocities. This is more computationally friendly. The drawback is that velocities have to be know, not just positions.

This exercise can of course be continued. A 5th order polynomial can be used with either specification of position, velocity and acceleration at each segment extremity. Or, alternatively one can assume continuity of velocity, acceleration and further derivatives.

A last approach is to fit a curve through multiple points (via some weighted error method). Although this provides optimal control of the "fidelity", it is much more complex and computationally demanding.

It turns out that the method of using a 3rd order polynomial with position and velocity specification is a good trade-off between computational requirements and error. The requirement of knowing segment point velocities is typically not a problem as the mathematical formula for position is typically known (velocity is the first derivative of position). This method is typically referred to as PVT.

Example of a PVT path

Assume that the required position profile is a sinusoidal curve:

$$x(t) = A * (\cos(\omega t) - 1)$$

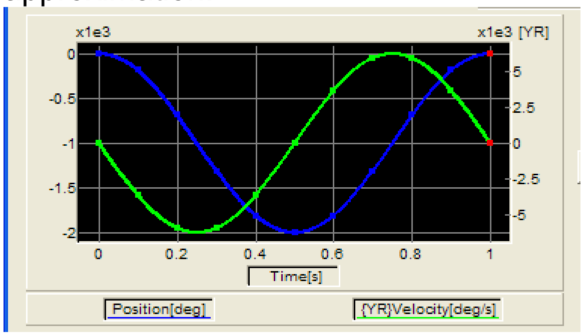
And that the period is 1 second (i.e. ω is 2π). Hence the velocity profile is:

$$v(t) = -A * \omega * \sin(\omega t)$$

We can arbitrarily create 100 millisecond segments, resulting in the following PVT table (with A=1000):

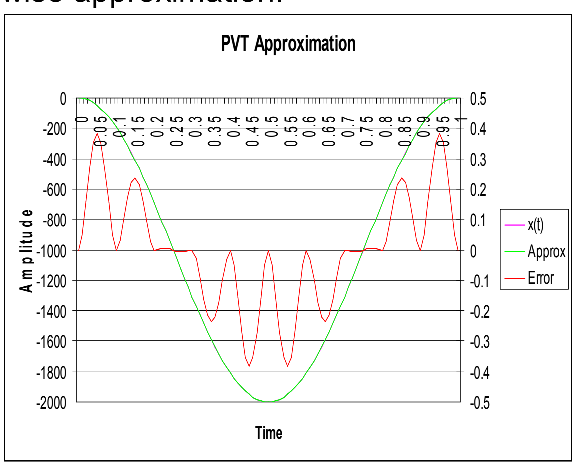
Time	$x(t_i)$	$v(t_i)$
0	0	0
0.1	-191	-3693
0.2	-691	-5976
0.3	-1309	-5976
0.4	-1809	-3693
0.5	-2000	0
0.6	-1809	3693
0.7	-1309	5976
0.8	-691	5976
0.9	-191	3693
1	0	0

These 10 points will then result in the following profile, after the 3rd order PVT approximation:



The blue curve is the position profile and the green curve is the velocity profile. The dots on the curves represent the segment points.

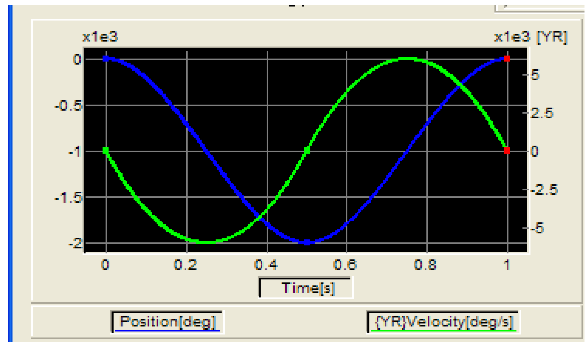
It is interesting to consider what the position profile error is, due to the piece wise approximation.



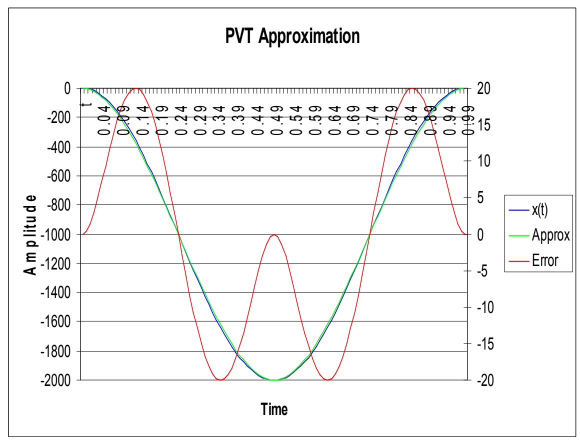
The figure above shows the actual position and its piece-wise PVT approximation (indistinguishable). Also plotted is the error (difference between actual and approximation) on the right axis. The worst case error is +/- 0.4 on a signal that ranges between 0 and 2000. So clearly the approximation is extremely good.

As a comparison, let us approximate the original curve with only 2 segments.

Time	$x(t_i)$	$v(t_i)$
0	0	0
0.5	-2000	0
1	0	0



Of course this is very crude, but mapping the error shows that even in this extreme case the errors are quite reasonable:



On a full scale, the actual and approximation are barely distinguishable, maximum error is +/-20 (again on a 0 to 2000 signal).

Segment Time Selection

As can be seen from the previous example, the segment times can be relatively large, compared to the actual move profile. Obviously, choosing a shorter segment time will reduce errors, but there is certainly a point of diminishing returns.

Although it is difficult to analytically determine a segment time value based on maximum allowed error, a good starting point is a factor of 10-20 of the highest frequency that must be tracked. This derives from data sampled system theory. In addition, once can consider a variable segment time, longer for slow varying positions (i.e. when velocity is low) and shorter for faster position changes (i.e. when velocity is higher).

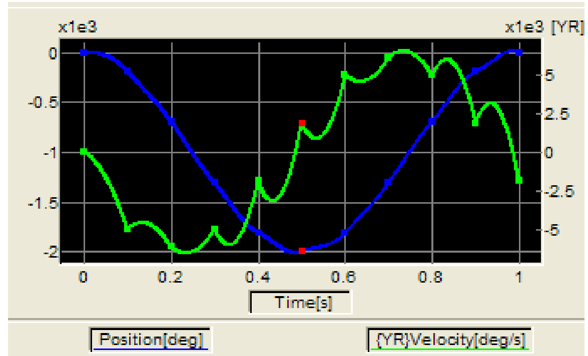
Importance of Velocity Values

Actually more important than the precise segment time, is the requirement to utilize accurate velocity values in the segment points. Below are the curves for the example above but using average velocity of the segment instead of actual segment point velocity.

The above table with average velocities is:

Time	$x(t_i)$	$v(t_i)$
0	0	-1910
0.1	-191	-5000
0.2	-691	-6180
0.3	-1309	-5000
0.4	-1809	-1910
0.5	-2000	1910
0.6	-1809	5000
0.7	-1309	6180
0.8	-691	5000
0.9	-191	1910
1	0	-1910

The resulting profile is:



Although the position profile still goes through the segment position points, the velocity profile has a lot of “wiggle”. This is due to the fact that the velocity is “forced” to an incorrect value at the segment points. This creates curvature in the velocity profile that deviates from the correct velocity (as the derivative of position at the segment points).

Conclusion

The PVT method is an easy to implement algorithm for arbitrary position trajectory generation. Segment times can be relatively large, without much loss of precision. The PVT points can either be stored in a table locally or can be parsed over a communication network to various axes. This allows implementation of a distributed multi-axis system without the need for a very high-speed deterministic network infrastructure.

Nippon Pulse SLP Acculine Stage

The SLP Acculine Series is a family of high speed, linear shaft motor-based, positioning stages. As an all-inclusive stage, the SLP stage provides integrated shaft support within the housing and simplifies the transition from conventional ball-screw systems. Because this stage system features a lightweight, compact linear shaft drive, the SLP is a low-profile, high-precision product. The built-in shaft motor is a maintenance free device and also eliminates sound and dust production. The SLP series features the smallest dead-zone of any similar stage system available on the market today. In addition, there are currently no stages on the market that match the SLP series' force-to-volume ratio, making it an outstanding solution for those applications with space limitations.

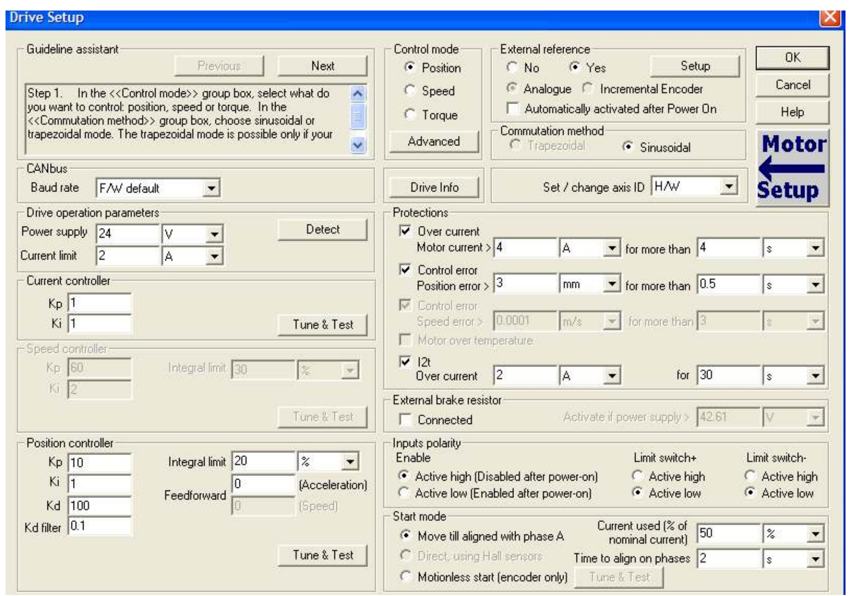
- Available in 3 sizes: SLP15, SLP25, SLP35
- High thrust, high speed, high responsiveness
- High precision
- Simple design and easy installation
- No-contact drive means low noise, long lifespan, and maintenance-free
- Integrated Linear Brushless Shaft Motor
- Integrated 1 micrometer resolution linear optical encoder
- Various standard lengths up to 1300mm travel
- Up to 3.5G acceleration and 3 m/s maximum speed

Product Feature: TSM Encoder-only Commutation Start-up

Introduction

Brushless motor commutation is of course at the basis of any properly working brushless servo motor based system. In the case of sinusoidal commutation, sufficient position resolution is required in order to create sufficiently smooth sinusoidal motor currents. Typically, an incremental encoder is used for this purpose. Although incremental encoders are very cost-effective and available with a wide range of resolutions, they always start from zero when first powered up. Brushless commutation on the other hand requires knowledge of the absolute motor position on power-up. The simplest solution is the addition of Hall sensors. This allows the system to know absolute position within 60 electrical degrees and allows trapezoidal commutation until the first Hall edge is encountered. The downside is of course additional components and wiring. Alternatively, a true absolute encoder can be considered but this increases the cost and limits the resolution.

Technosoft Encoder-only Commutation Approach #1

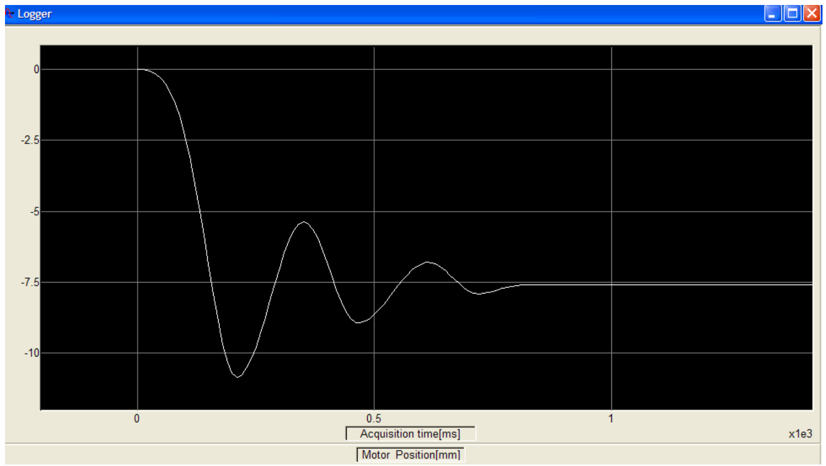


Technosoft drives allow incremental encoder only commutation via a phase-align mechanism. Current is driven through 2 different sets of phases sequentially. When current is driven through a first set of phases, the motor permanent magnets will align themselves and establish a base position. After a configurable settling time, another set of phases is energized. This will cause another magnet re-alignment, with a rest position at 120 electrical degrees away from the first position. At this point, the absolute position of the permanent magnets is known and sinusoidal commutation is possible. This approach causes physical motion of the motor of course. What the final amount of load motion will be depends on the number of poles (or the pole pitch in case of linear motors) and the mechanical transmission between motor and load.

Although this approach is simple and often useable with higher pole count motors, it has 2 significant drawbacks:

- Potentially large load movements upon initialization
- Potentially incorrect phasing due to friction or gravity

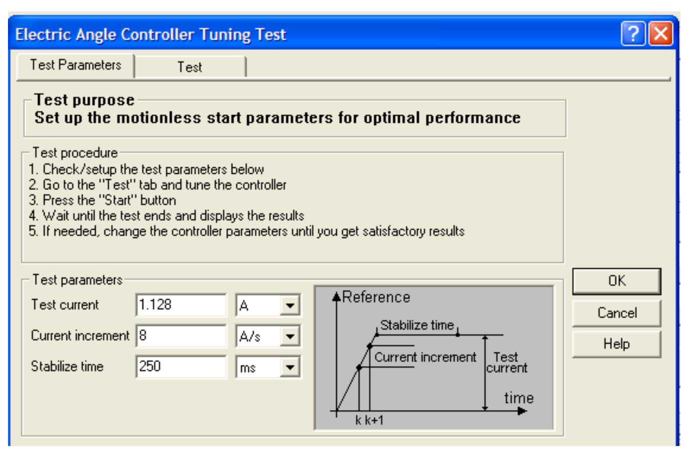
Below is a picture of the actual motion of a 30 mm pitch linear motor during this phase alignment:



In this case, there is practically no friction in the system and large oscillations occur.

Because this phase alignment approach assumes that the motor will align itself accurately with the energized phase, any misalignment due to friction, gravity or a mechanical hard stop will cause an error in the absolute angle calculation and lead to incorrect commutation (worst case one can be 180 degrees off, leading to polarity reversal).

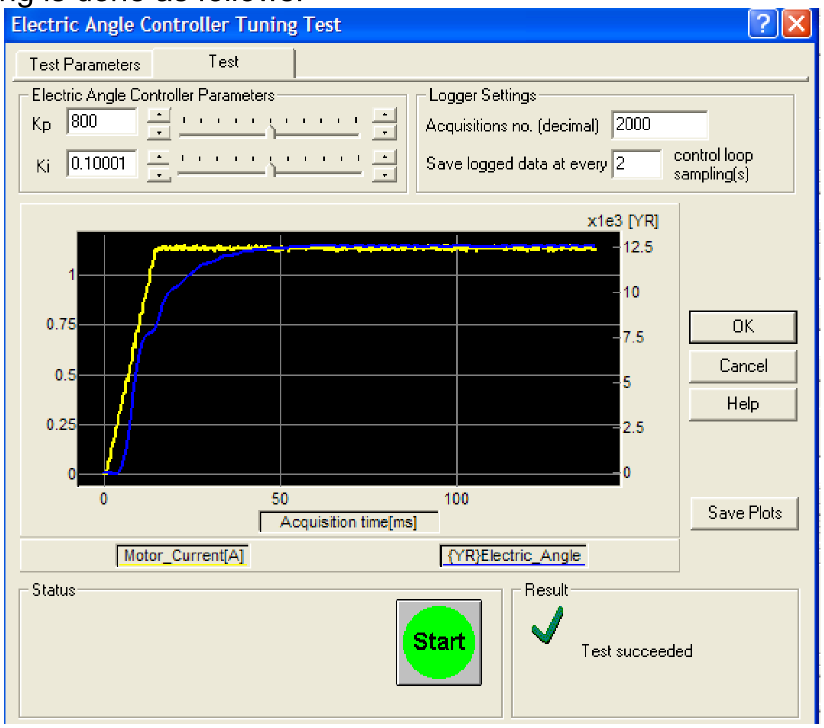
Technosoft Encoder-only Commutation Approach #2



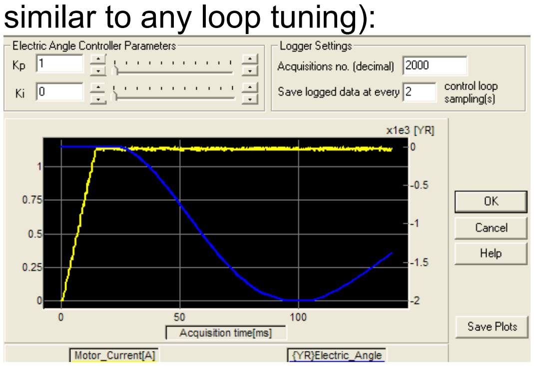
The second incremental encoder-only commutation method is much more refined and eliminates quasi any motion during phase finding. The basic concept behind this approach is the search for an electrical angle that results in no torque or force. One recalls from the torque production in a brushless motor that at 90 degrees the torque production is optimal. However, at 0 and 180 degrees the torque production is zero. This algorithm, because it attempts to find an angle that causes no torque, results in extremely little motion (on the order of a few encoder counts).

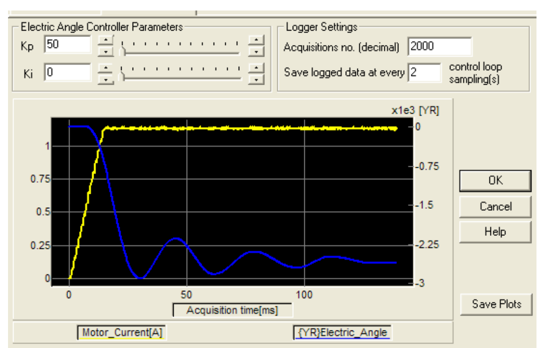
The algorithm used to find the phase angle is an actual PI-controller, and can be adjusted by the user. Prior to tuning, the user can set the current level and timing used during the phase finding.

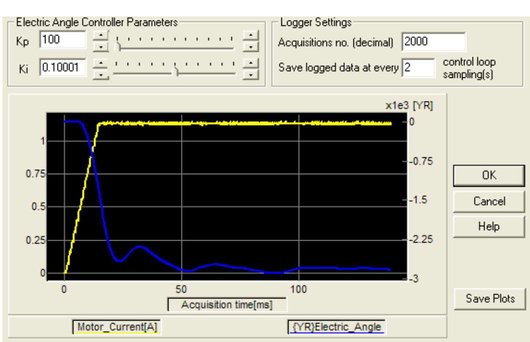
The actual tuning is done as follows:

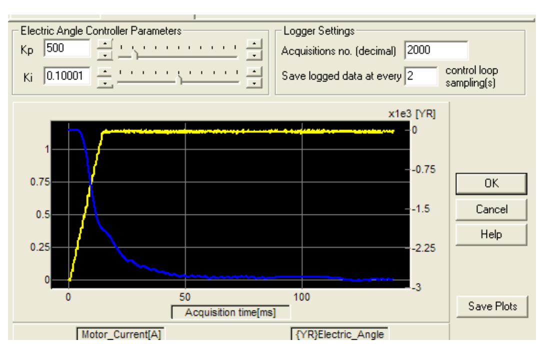


To show the actual tuning process, below are a few steps of the iteration (which is very similar to any loop tuning):









Application Solution: Cartesian Robot Control

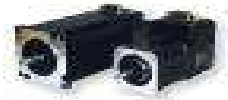
Bio-pharmaceutical companies, medical device manufacturers, research labs and hospitals all make use of a variety of lab automation process equipment for pipetting, diagnostic testing, mixing, incubating and device construction just to name a few. At the heart of all of this equipment is the tried and true, 3 axis, "Pick and Place" assembly. Early on these "Cartesian arms" were either large, dedicated standalone units or simpler, semi-automatic units connected to a PC. With the increasing incorporation of embedded controls for both motion and logic control, these units have become more capable to respond to the demands of their users. End users wanting faster processing times, more and varied testing capabilities, lower operating costs and the ability to combine equipment for evolving needs have created the following criteria for OEM builders of lab automation equipment, which use 3 axis "Pick and Place" units:

1) smaller footprint and volume
2) low power consumption for potential use in mobile areas – field use, war zones, disaster zones
3) common, modular components for changing test and manufacturing needs
4) lower costs by using integrated components
5) ability to operate as an independent piece of process equipment or networked
6) compatible with standard PC Bus architecture
7) increased throughput via faster speed and/or multiple function capable systems using items such as "test-on-chip" components
8) Easy to use set-up/control software

The challenge for OEM builders and integrators, therefore, has increasingly been the search for suppliers of cost effective, easy to use and capable integrated components not only for motion control but also for machine logic. Traditional control architecture includes either a PC, PLC or custom control board (or a hybrid mix of them all) as its center-piece which connects to stepper/servo drives, field I/O and potentially an HMI of some sort. This solution involves multiple vendors, generally multiple software packages for set-up and programming, increased complexity in wiring and programming. Motion control component vendors have been scrambling to provide simpler, more integrated solutions as a result.

One motion control company in particular, Arcus Technology Inc. of Livermore, CA. (http://www.arcus-technology.com) has developed a line of intelligent stepper components which are ideally suited for the core control/motion control architecture of a 3+ axes Cartesian robot unit. The function of the remainder of this article will illustrate how to accomplish the controls for a Cartesian pick and place unit using the following components from Arcus Technology:

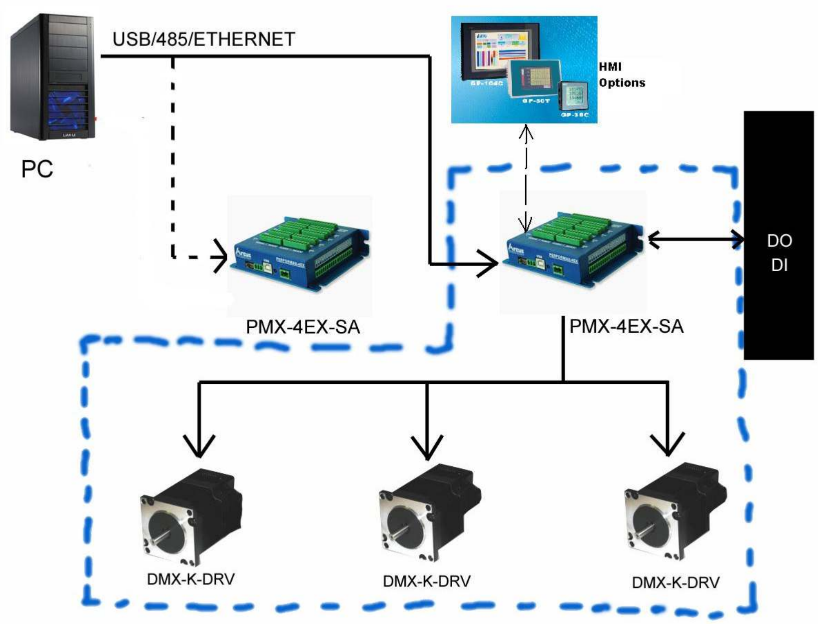
- DMX-J/K-SA: integrated motor, encoder, drive and controller package



- PMX-4EX-SA: 4 axes controller, opto I/O, encoder inputs, and USB/RS-485 communication interfaces

- DMX-K-DRV: integrated stepper motor, micro-step driver

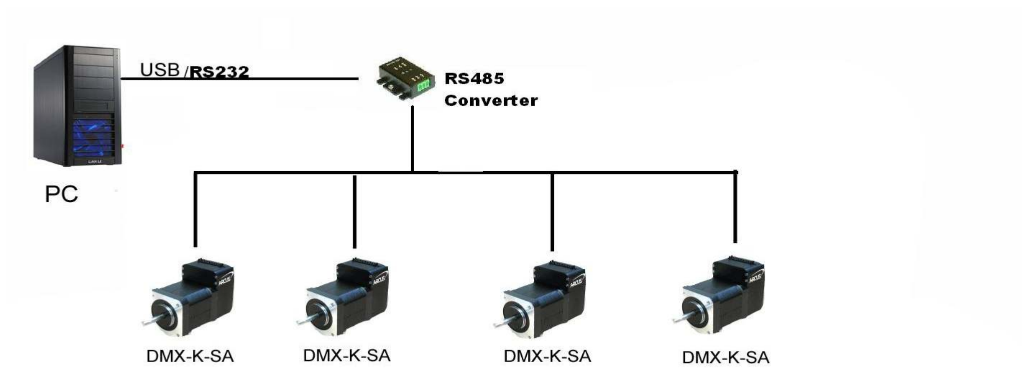
Because the Arcus components are intelligent and integrated, we have two possible layouts for the motor/drives/controls based on how much intelligence (logic and motion) we want to have distributed and where it is to be placed. The figure below shows a potential hybrid system that could be run with or without the use of a PC depending on the complexity needed. With the PC incorporated, data logging and analysis become possible. If the PC is not used (see the block with the dashed line), motion and logic control are accomplished via Arcus' BASIC like language within the PMX-4EX. It is also expandable via USB, RS485 or Ethernet, allowing multiple nodes or machines to share info. The diagram above illustrates a very flexible and modular control scheme. Since the block identified with the



blue dashed line contains a PMX-4EX-SA, the code for motion and logic could be programmed to run a simple 3 axes Cartesian arm in “standalone fashion”. The DMX-K-DRV units are compact and house both the motor and drive with an additional encoder possible on the NEMA17/23 frame sizes. The encoder provides the possibility to run closed loop stepper using Arcus Technology's “Step-n-Loop” algorithm thus providing servo-like performance without the need for tuning a typical servo loop. For smaller size needs, the DMX-K-DRV is available as a NEMA11 frame size package. With the use of an optional HMI touch-screen, even the simple, standalone design becomes flexible as it allows for flexibility in programming recipes, changing variables for a “shell program”, etc. This PMX-4EX-SA based “building block” can also be connected to a PC via USB, RS-485 or Ethernet, either singularly or in groups to create a larger, flexible system. Combining the PC with the 4EX not only allows for a sophisticated GUI, but gives the capability to create a hybrid , mixed system which allows the designer to chose where intelligence is located – locally near the process and/or distributed.

Programming via the PC over the USB port is done via Arcus' Performax COMM DLL which allows programming in C++, VB6.0, LabView and Python, in addition to be able to send ASCII commands to the controller. Programming options are also available using RS-485 and Ethernet similarly. Linux drivers are also now available for USB2.0.

If the intention or existing design architecture is PC centric and there is no need for the “redundancy” of the PMX-4EX-SA due to space limitations, power limitations, etc., then we can create a much more distributed scenario while still minimizing wiring and allowing both motors and some local intelligence close to the actual actuation mechanism of the Cartesian arm. See figure below:



In the architecture diagrammed above, the PC becomes the core of the control scheme. The difference is that we have chosen the Arcus DMX-K-SA. This component is an intelligent, standalone, motor-drive and controller package, eliminating the need for the 4EX controller in Fig. 1. This may be the case where a small, modular robot arm is being built for inclusion into an existing machine or line. This would necessitate only the control for motion being at the motor " ready to go" to receive commands from the PC. Another case would have the motion (simple, sequential point to point moves) being stored on the DMX-K-SA controller and then commanded to execute via the PC, since it would be handling the overall logic. In this case, any coordinated or interpolated motion would be commanded via the PC as well. As we can see in Fig 2, the PC takes on a much greater role – possibly running a custom program controlling various parts of a machine, logging and analyzing data, running a SCADA package, etc. The DMX-K-SA provides a compact, integrated solution where space limitations, wiring runs, and distributed motion control are a priority. By providing intelligent and integrated stepper components Arcus Technology has given OEMs and Integrators both the modularity and flexibility to meet the demands of today's Cartesian robot arms.

For more information about any of the above topics or general questions or comments, please contact us:

Motion Designs
contact@motion-designs.com
Tel 805.504.6177

Motion Designs is a technical sales and engineering company with extensive machine and motion control experience. We work with some of the best manufacturers in the industry as witnessed by our present line card:

- www.amosin.com: AMO manufactures induction based precision linear and angle measurement encoders.
- www.arcus-technology.com: Arcus Technology manufactures stepper motor, drive and controller technology, providing USB, Ethernet and Mod-Bus connectivity.
- www.nipponpulse.com: Nippon Pulse manufactures the unique linear shaft motor, a direct drive linear brushless servo motor.
- www.shinano.com: Shinano Kenshi manufacturers cost effective brushless servo motors and assemblies.
- www.stegmann.com : Stegmann is a leader in high performance motor feedback solutions.
- www.technosoftmotion.com : TSM is a leading DSP motion control technology company specialized in the development, design and manufacture of digital motor drive products and custom motion systems.