DELTA TAU
Data Systems, Inc.

Last revision: 12/5/01

Why PMAC Controllers Are Easy To Use

Delta Tau's PMAC and Turbo PMAC families of controllers justly have the reputation as being the most powerful and flexible in the world. Many people do not realize that PMAC controllers have many features that make them among the easiest to use as well. These features aid in the ease of use for both basic and more advanced applications.

The Basics

Interactive Command Environment

PMAC controllers provide a direct interactive communications environment permitting the user to command and query the controller directly with simple commands. This makes setup, debugging, and diagnostics very easy. In fact, many applications can be accomplished entirely with these simple interactive commands, requiring no stored programs in the PMAC.

High-Level, English-Based Programming Language

Many motion controllers work with assembly-language style short acronyms for their commands, making the programs and command sequences unreadable to anyone not completely immersed in the programming.

PMAC controllers use high-level English-based commands (IF, WHILE, DWELL, CALL, etc.) whose meaning can easily be followed by "casual observers". Furthermore, the "macro substitution" capability in Delta Tau's PC-based editors permits the generation of PMAC programs that read like "natural language".

Automatic Sequencing of Moves

In most computing environments used for motion control, the user program issues a command to start a move (with actual execution of the move handled "underneath" the program). It is then the responsibility of the user's program to figure out when the move has completed, with some structure like:

```
WHILE (MoveOver=0) WAIT
```

in order to sequence the execution of the program properly.

In PMAC controllers, the controller's operating system automatically handles the sequencing of the moves and the associated calculations. This means easier, simpler, shorter, programs that are easier to read as well.

Single Method of Programming Moves

Most motion controllers have completely different methods for programming point-to-point moves and blended, or contoured moves. Also there are completely different methods for programming moves that are a function of time, and moves that are a function of a master position.

Not so in PMAC controllers. PMAC's unified method towards move programming lets you do all of these types of motion in a single style of programming. Usually, just a single variable needs to be changed in order to change the mode of operation.

Programming in Engineering Coordinates

Many motion controllers require you to program the motion in the raw units of encoder counts. Not so with PMAC controllers – they allow you to enter the relationship between the raw measurement units and the desired user (engineering) units once in the setup, and then program the motion in terms of the engineering units from there on. If you want to change the engineering units (e.g. from metric to English), you only have to change a single number for the axis.

For example, if Motor #1 has 1-micron measurement resolution, and you wish to program it as the X-axis in units of millimeters, you simply define:

```
#1->1000X
```

This command assigns Motor 1 to the X-axis with 1 programming unit (mm) of X equal to 1000 counts of Motor 1. If you wish to change this to inches, you simply change the definition to:

```
#1->25400X
```

Centralized Control for Easy Coordination

Many vendors are pushing distributed motion control schemes, with single-axis smart drives connected on some kind of network, ring, or Fieldbus. The appeal of these systems is a simplification of wiring – a single slender cable can supposedly transmit all of the information needed between the system's central intelligence and the relatively self-sufficient smart drives.

This scheme has its place for simple, uncoordinated motions. However, as soon as even simple coordination between axes is required, it becomes very difficult to do with distributed intelligence, even for a task as basic as reacting properly on one drive to a fault on another. Sophisticated coordination concepts, such as circular interpolation or contouring, can be virtually impossible to do properly in a system with highly distributed intelligence. Many companies have started down the road of a network of distributed smart drives, only to abandon the approach when it proved too difficult to program and debug. Very often in these distributed systems, data has to be handled by four processors: the central computer's main processor, the central computer's network/bus communications processor, the drive's network/bus communications processor, and the drive's main processor.

By contrast, PMAC controllers use a highly centralized software approach, where all the key information needed for tight coordination is immediately accessible by PMAC's processor. Multi-axis moves in a single PMAC "coordinate system" are automatically fully coordinated.

Even if distributed hardware is required for wiring simplification, PMAC's MACRO ring provides single-strand (optical or electrical) communications at speeds and determinism levels high enough that the control software can be entirely centralized – including all of the servo loops. This preserves the same ease of programming as with a centralized-hardware approach.

Asynchronous PLC Program Execution

Whether the move sequencing is done explicitly in the motion program or automatically by the controller's operating system, there is usually a need for other calculations to be done that are asynchronous to the sequence of motion. Many controllers do not have this type of multi-tasking capability, and it is tremendously difficult to implement these types of algorithms.

PMAC controllers have "PLC" programs that are designed to run asynchronously to the motion programs and independent from them. This makes it very easy to implement additional types of calculations.

True Floating-Point Math Capabilities

Many motion controllers do not provide true floating-point math capabilities, often carrying just a little bit of fixed-point fractional resolution in some variables. In these systems, the user must take a great deal of care to see that calculations do not run into overflow, underflow, and round-off/truncation problems. PMAC controllers offer a full floating-point math library (using 48-bit floating-point variables) that automatically provides a dynamic range sufficient for virtually all motion control applications encountered so far. This means that the programmer can concentrate on the application, not on the limitations of the controller.

Automatic Variable Type Matching

Most computing environments, including most motion controllers, require you to match the types of your variables carefully – bit, byte, "short" and "long" variables, fixed-point and floating-point variables – and explicitly perform any conversions between types. This can be time-consuming and confusing for the vast majority of motion control users who are not computer-science experts.

PMAC controllers automatically handle all type matching of variables, so the user does not have to worry about any of these issues. It is literally possible to combine single-bit Boolean variables, short fixed-point variables, and double-word floating-point variables directly in a single expression.

Fast Servos with High Resolution and Small Delay – Easy Tuning

Many people regard fast servo update times with minimal computation delays and high-resolution outputs as something needed for only super-precision or super-fast applications. However, even in "standard" applications, the reduced "phase lag" and "quantization noise" that a fast, precision servo provides makes the tuning of the system far easier.

While any digital servo loop adds phase lag compared to an ideal analog loop, PMAC's highly optimized loops add a phase lag that is often an order of magnitude less than competing controllers. Large amounts of phase lag can make it very difficult to tune the system for good response, and can limit the performance of the system.

Hardware Position Capture and Compare

PMAC controllers have super-fast position-capture and position-compare functions built into the hardware circuitry – no software intervention is required. The hardware capture circuits mean that you automatically get a precise position reading at the instant of your trigger (for homing, probing, or registration). You do not need to carefully evaluate your speeds and delays to determine if your capture is accurate enough – it always is!

Because the position compare outputs trigger instantly when the preset actual position is reached, many things become easier. For instance, it is not necessary to keep the servo errors within the bounds of your required output tolerances. With this feature, it does not matter how big the servo error is; the output will still be triggered precisely by the actual position register. This feature can eliminate a great deal of development time that would be required for super-accurate tuning.

Beyond the Basics I – PMAC's Powerful Built-In Functions

The breadth and depth of PMAC's feature set means that PMAC can handle many problems automatically and simply that are very difficult to do with simpler controllers.

Servo-Loop Adjustments

PMAC controllers provide several terms in their servo algorithms that make it very easy to compensate for imperfections in the physical system. In controllers without these terms, it is much more difficult, and sometimes not possible, to solve these problems. These terms include:

- Deadband compensation: These terms make it possible to create a deadband automatically, or to compensate for a physical deadband in the system. For example, proportional hydraulic valves can have significant deadband in their zero-crossing compared to the more expensive servo valves, but many users find that PMAC's deadband compensation corrects for this enough that they can use the less-expensive and lower-maintenance proportional valves.
- Output offset parameter: This term makes it possible to compensate digitally for analog-circuit offsets, no pot-tweaking is required.
- Friction feedforward compensation: Traditionally, Coulomb ("dry") friction, a non-linear effect, could not be compensated for directly. Attempts to compensate for the lags that this friction introduced by using integral gain during motion often resulted in overshoot at the endpoint. PMAC controllers have a special term to compensate directly for this effect, without causing other problems.
- Backlash compensation: Most motion control systems use only a position sensor on the motor for simplicity and cost-effectiveness. However, this leaves the system unable to directly measure the effect of backlash in the gearing. If the magnitude of the backlash can be characterized, PMAC can automatically compensate for the backlash, either with a constant value, or varying as a function of position, eliminating the need to ship the system with an expensive sensor on the load to correct for this problem.

Straightforward Dual Feedback Support

Building-Block Trajectories

Cutter-Radius Compensation

In many applications, particularly those involving cutting shapes, the path of the cutting tool must be offset from the path given along the edge (2D) or surface (3D) of the shape. If the controller cannot do these calculations automatically, as most simple controllers cannot, the applications

programmer must figure out the offsets. Particularly difficult is calculating the offset intersection points, especially when arc moves are used.
PMAC controllers have built into them 2D and 3D (Turbo PMACs only) cutter-radius compensation algorithms that do these calculations automatically, permitting the user to focus on the part shape desired, and not the internal details of what the controller needs to do to create the part shape.

Kinematic Calculations

Increasingly, people are using mechanisms with non-Cartesian geometries to achieve higher performance levels and/or lower cost and weight. This increases the complexity of programming the tool's path and destination points, which most users will still want to specify in the Cartesian coordinates of the tip.

Most controllers force you to program the motion in the coordinates of the actual joints, or motors, a difficult and non-intuitive task, but Turbo PMAC controllers permit you to enter special subroutines with the "forward kinematic" (joint-to-tip) and "inverse kinematic" (tip-to-joint) conversions. It then automatically calls these subroutines at the appropriate times, thus permitting the user to program the motion as if it were a Cartesian system.

Lookahead Control

With most controllers, the responsibility for ensuring that programmed motion does not violate machine constraints such as maximum acceleration falls on the user as he programs each move. Many controllers can handle these constraints well for simple point-to-point moves, but almost none can handle them for anything more complex, especially when the required acceleration and deceleration must occur over many programmed moves. This puts the onus on the user, who must not only check every move against the constraints, but understand thoroughly how the controller blends moves together to devise programs that do not violate constraints, but are reasonably efficient in executing the sequence within constraints.

PMAC's "lookahead" control algorithms automatically scan far ahead in the programmed move sequence to identify potential violations of machine constraints – position, velocity, and acceleration. They then slow the trajectory as needed so as not to violate any of the constraints. All the user must do is tell PMAC what the constraints are, then give it the motion sequence – PMAC automatically does the rest.

Beyond the Basics II – PMAC Lets You Create New Features

Access to All Internal Registers

Loops Around Loops