PMAC tuning

Wayne Lewis

Osprey DCS

2018-03-09

Overview

This presentation covers configuration and setup of a Delta Tau Turbo PMAC. It includes information on:

- tuning of current loop (relevant for GeoBrick LV)
- position loop tuning
- encoder configuration
- setup and initialization
- coordinate system configuration
- EPICS interface

Motor types

The GeoBrick LV is capable of driving:

- two-phase stepper motors
- three-phase brushless servo motors
- brushed servo motors

Axis feedback

An encoder is used to measure the position of the motion axis. It is typically located as close as possible to the final mechanical component as possible to eliminate any mechanical imperfections in the drive train.

If a stepper motor is used, an encoder is not required. The controller can count the steps sent to the motor, and use this as the input to the position control loop.

A brushless servo motor needs to have an encoder on the motor shaft to allow the controller to perform commutation. An application using brushless servo motors will often have two encoders - one on the motor shaft, one on the load.

Encoder types

Encoders can be either incremental or absolute. An absolute encoder provides an absolute indication of position on controller startup, removing the need for a homing process. An axis using an incremental encoder must be moved to a known reference position prior to normal usage. All positions are measured relative to this reference position.

Axis protection

Most motion axes will have end of travel limit switches. When one of these is reached, the controller performs a controlled, rapid, deceleration. The deceleration rate is set by `Ixx15`.

The limit direction sense is important, and must be set in the wiring; there is no option for changing limit direction sense in the configuration. The controller will only stop movement in the direction of the limit that has been activated, which allows the controller to reverse direction away from an active limit.

The limits switches must be electrically normally closed, and open when the limit is reached. The PMAC does not have the capability of using normally open limit switches.

Limit switch disable

The axis limit switches can be overridden by setting the $20000 bit of `Ixx24` to 1. This will disable the limit switches for this axis. This is an appropriate setting if an axis has no limit switches (e.g., a rotary stage). For most stages, limit switches should only be disabled in exceptional circumstances, and with extreme care.

Overriding limit switches

Extreme caution must be exercised when disabling limit switches. The PMAC will not stop motion when a disabled limit switch is reached, and equipment damage may occur.

Initialization PLC

PLC 1 is typically used as a setup PLC. It runs first on controller startup (assuming there is no PLC 0) and performs one-time startup configuration for the controller. Setup includes:

- clearing amplifier faults
- setting motor type in amplifiers
- defining lookahead buffers for coordinate systems
- enabling and disabling all other PLCs
- setting up absolute encoders

Encoder conversion table

A reasonable default encoder conversion table for the GeoBrick-LV could have the following:

- 8 axes of incremental encoders (I8000-I8007)
- 8 axes of direct microstepping configuration (I8008-I8031)

If the controller is configured to accept absolute encoders, the following additional default configuration could be added:

- 8 axes of absolute encoders (I8032-I8047)

The remaining encoder conversion table entries can be used for custom encoder configuration.

Motor type setup

GeoBrick LV amplifiers must be configured for the motor type. Refer to **MOTOR TYPE AND PROTECTION POWER-ON PLCS** section in GeoBrick LV User Manual.

Stepper motor amplifier setup - Channel 1

`CMD"WX:78014,F8CDFE"` - set stepper mode

`CMD"WX:78014,F84DFE"` - clear amplifier faults

Brushless servo motor amplifier setup - Channel 1

`CMD"WX:78014,F8CCFE"` - set servo mode

`CMD"WX:78014,F84CFE"` - clear amplifier faults

Current loop tuning

When using the internal amplifier, the current loop must be tuned prior to performing any other configuration. The following parameters are set during stepper motor current loop tuning:

Parameter	Function
`Ixx77`	Motor current
`Ixx61`	Current loop integral gain
`Ixx62`	Current loop forward path proportional gain
`Ixx76`	Current loop back path proportional gain

Table 1: Current loop tuning parameters

Current loop tuning values

For stepper motors, the following are potential starting values for the tuning parameters:

Parameter	Value	Comment
`Ixx77`	Variable	Motor phase current in mA
`Ixx61`	0.035	Increase to speed up response
`Ixx62`	0.35	Decrease to speed up response
`Ixx76`	0.75	Often set to zero, then increase to reduce overshoot

Table 2: Current loop tuning values

Servo motor setup

When using the internal amplifier to drive a brushless servo motor, a number of parameters need to be changed.

Parameter	Description	Value
`Ixx70`	Number of commutation cycles	>=1
`Ixx71`	Encoder counts per N commutation cycles	Variable
`Ixx72`	Commutation phase angle	683 (see SRM)

Table 3: Servo motor parameters

Commutation angle warning

Commutation phase angle

Make sure the `Ixx72` value is set correctly for the motor type. Using the GeoBrick LV internal amplifier, it should be set to 512 for a two phase stepper, and to 683 for a three phase servo. Refer to the Software Reference Manual and the GeoBrick LV Manual for more information.
If these values are incorrectly set, it is possible to put the current loop into positive feedback, which can destroy the motor.

Servo motor parameters

`Ixx70` - **number of commutation cycles**

`Ixx71` - **encoder counts per N commutation cycles**

These two variables must be set so that the ratio of $\frac{Ixx71}{Ixx70}$ is the number of encoder counts per electrical cycle of the motor. This can usually be found from the motor and encoder specification sheets. If this is not set correctly, then the controller will not be able to change the phase energization at the correct time, and the motor will not turn properly.

Axis direction configuration - limit switches

The limit switch direction sense cannot be changed in the controller configuration. If the limit directions are incorrect, they must be changed in the wiring to the controller. These should be set consistent with the facility coordinate system standard or some other standard.

Axis direction configuration - motor direction

Check the motion direction by issuing an open loop command (e.g., **#1o1**) and observing the motion direction with respect to the limit switches.

For a stepper motor, the motion direction can be reversed using the `Ixx70` parameter. It should be set to 1 or -1. Inverting the sign will reverse the direction of the stepper motor. A stepper motor can also be reversed by changing the polarity of one pair of phase wires.

For servo motors, the direction is reversed by switching two of the phase wires.

Axis direction configuration - encoder direction

Incremental encoder direction can be switched using the I7mn0. For a typical quadrature encoder, the value of I7mn0 should be 3 or 7. Changing between these two values reverses the direction of the encoder counter.

Absolute encoders cannot have their direction changed in the controller configuration. It may be possible to change the direction in the encoder itself. However, for some absolute encoders (Renishaw Resolute, AMOSIN), the encoder direction is a function of the scale orientation. In this case, the scale must be installed in the correct direction. The only way to invert the encoder direction is to physically reverse the scale.

Axis direction configuration - summary

At the end of the axis direction configuration process, the following polarities must be consistent:

- limit switches
- motor direction
- encoder direction

Axis direction configuration - consequences of error

If the encoder and motor are not consistent, the position loop will drive the motor in the wrong direction with respect to the encoder. This will very quickly result in a fatal following error.

Limit switch direction

If the limits are not consistent with the motor direction, the motor will not stop when it hits the limit. Be certain that the limits are consistent with the motion direction early in the commissioning process.

Position loop algorithm

The Delta Tau controller uses a PID algorithm with standard feedback gains (proportional, integral, derivative) and two main feedforward gains (velocity and acceleration).

Feedback gains act on both setpoint and readback terms (i.e., error) while the feedforward terms act only on the setpoint.

Refer to the `PID/Feedforward/Notch Servo Filter` diagram in the Turbo PMAC User Manual.

Position loop tuning objectives

The purpose of a motion axis can be split into two broad categories:

- point to point movement
- trajectory movement

Position loop tuning can optimize one or the other of these.

Position loop tuning - disclaimer

Note on guidelines

All the guidelines in the following pages are suggestions only. They are intended to be used by personnel familiar with position loop tuning and the equipment being commissioned.
The user must take into account any specific characteristics associated with the loop being tuned. They must also ensure that all safety issues have been considered and suitably managed.

Position loop tuning - point to point

Most motion axes will be tuned for point to point movement. The objective here is fast and accurate movement to the target position. Minimizing overshoot can be useful if there is any mechanical hysteresis in the drivetrain.

Parameter	Description	Typical value
`Ixx30`	Proportional gain	Variable
`Ixx31`	Derivative gain	0
`Ixx32`	Velocity feedforward gain	Variable
`Ixx33`	Integral gain	0
`Ixx35`	Acceleration feedforward gain	0

Table 4: Position loop tuning parameters - point to point

Position loop tuning - other parameters

Parameter	Description	Typical value
`Ixx11`	Fatal following error	Variable
`Ixx28`	In-position band	160
`Ixx34`	Integration mode	1
`Ixx69`	Output limit	Variable

Table 5: Position loop tuning additional parameters

Fatal following error

`Ixx11` - **Fatal following error**

This needs to be set large enough that a normal move does not result in a fatal following error, but small enough that unusual circumstances can be detected and the motor tripped. Examples include a stalled motor, jammed motion stage, or hard stop.

Care must be taken with moves that are smaller than the fatal following error, as the following error may never reach the fatal following error limit unless the system overshoots the target position. For this reason, the fatal following error limit should be set as small as possible.

Warning following error

Note that the warning following error (`Ixx12`) is not used in the PMAC, other than setting a status flag. This value is typically set to half the fatal following error, but unless a specific application is checking the warning following error status, the value will not affect the axis operation.

In-position band

`Ixx28` - **In-position band**

This parameter is used to define when the controller reports the axis as being in position. This status is used by the EPICS motor record to signal move completion. It is also used by PLC 7 to initiate the timer for moving to holding current or amplifier shutdown. Setting this is application specific and depends on the end user requirements.

In-position band

`Ixx34` - **Integration mode**

This parameter is typically set to 0 for trajectory move tuning. The setting does not matter if the integral gain is zero.

`Ixx69` - **Output limit**

This parameter sets the maximum output from the PID loop. For a stepper motor, it limits the maximum speed of the motor. It should be set somewhere below the speed at which the stepper motor stalls. This will prevent the position loop output becoming too large and causing the motor to stall. If the motor stalls, the loop will typically trip soon on fatal following error.

Position loop tuning parameter guidelines

Starting point: proportional and velocity feedforward only

`Ixx30` - **Proportional gain**

If the relationship between encoder counts and motor steps is known, use this ratio to adjust `Ixx30` from default value. e.g., if 10 encoder counts per motor step, reduce `Ixx30` by factor of 10. Be conservative, smaller gains may result in fatal following error, but will not be unstable. Better to start small and increase.

Proportional gain has a multiplying effect on all other gains, including feedforward and feedback gains.

Position loop tuning parameter guidelines

`Ixx32` - **Velocity feedforward gain**

Typically start around 1000 and adjust up or down depending on whether actual position is leading (reduce K_{vff}) or lagging (increase K_{vff}) commanded position.

Position loop tuning - trajectory

Tuning for trajectory movement attempts to ensure the actual motor position follows the commanded trajectory as closely as possible, minimizing following error at all times during the movement. This is important for axes that are used for fly scanning applications, and for motion axes that are part of coordinate systems.

Parameter	Value	Description
Ixx33	Integral gain	> 0
Ixx34	Integration mode	0

Table 6: Position loop tuning parameters - trajectory

Position loop tuning - trajectory

The integration mode should be set to zero to allow integral action throughout the move. This will allow the controller to compensate for any accumulated error as the move progresses.

A consequence of this is that the move will often overshoot due to the non-zero integral term in the PID equation. This will result in increased post-move settling time - a tradeoff for the improved trajectory tracking.

PLC programs

PLC programs run continuously unless explicitly disabled. Examples of applications include:

- Controller initialization
- Holding current and motor deactivation
- CPU and status reporting
- Homing
- Position reporting for coordinate systems

Motion programs

Motion programs:

- require a coordinate system definition
- are run on demand
- allow complex motion sequences and checks

Coordinate systems

One particularly powerful feature of the PMAC is the ability to implement coordinated moves. This allows one or more virtual axes to be defined in terms of their relationship with the physical axes. The user can request a move in the coordinate system axis, and the controller handles the calculation of the required position and speed of the physical axes.

Kinematics terminology

Forward kinematics

The equations that convert the physical motor positions to the coordinate system axis positions.

Inverse kinematics

The equations that convert the coordinate system axis positions to the physical motor positions.

The forward and inverse kinematic equations must be consistent.

Converting between the two sets of equations involves inverting the equations, but this is not always feasible for non-linear equations. Regardless of the mechanism of arriving at the forward and inverse kinematic solutions, the consistency requirement remains.

Kinematic definitions

If the relationship between the physical and coordinate system axes is linear, then the kinematics can be defined directly using these linear expressions.

If the relationship is more complex, then the forward and inverse kinematic equations can be specified explicitly. Explicit definition can also be used for linear relationships.

Explicit definition of the kinematics allows for numerical solutions of kinematic relationships that do not have a closed form solution. Examples of this include the forward kinematic solution for parallel robots (e.g., hexapods), and inverse kinematic solution for some serial robots.

Coordinate system axes

When defining a coordinate system, the number of coordinate system axes and the number of physical axes must be equal. Otherwise the system is either over- or under-constrained. In some cases, the resulting coordinate system axes may be held at zero or some other constant value, but they must still be calculated in the kinematics.

An example of an ‘unwanted’ coordinate system axis is ‘skew’ or ‘offset’ in a gantry system.

Motion programs and coordinate systems

A motion program must be run on a coordinate system.

The physical motors must be assigned to a coordinate system.

The kinematic relationship between the physical axes and the coordinate system axes must be defined.

The motion program allows for a complex sequence of moves to be performed. It handles checking for move completion, making the sequencing relatively simple.

If a sequence is required on a single axis, the relationship between the coordinate system axis and the physical axis can be 1:1, or some other relationship to convert to the desired engineering units.

Coordinate system position reporting

The PMAC uses the forward and inverse kinematics as part of move calculation, but does not automatically report the current coordinate system axis positions as a function of the physical axis positions. If the coordinate system axis positions are required, a PLC must be written that implements the forward kinematic calculations and stores the results in variables accessible by the user.

Coordinate system examples

- Slit blades $\rightarrow$ gap and center
- Gantry systems $\rightarrow$ linear translation and skew
- 3 jack kinematic mount $\rightarrow$ translation, pitch, roll
- 4 jack non-kinematic system $\rightarrow$ translation, pitch, roll, twist

I variables - commonly used

I variable	Description
Ixx00	Axis enable
Ixx01	Commutation enable
Ixx03	Position loop input
Ixx04	Velocity loop input
Ixx12	Fatal following error
Ixx22	Jog speed
Ixx23	Homing speed and direction
Ixx24	Flag settings
Ixx28	In-position band
Ixx69	Position loop output limit
Ixx77	Stepper current

Table 7: Common setup I variables

I variables - tuning

I variable	Description
Ixx30	Proportional gain
Ixx31	Derivative gain
Ixx33	Velocity feedforward gain
Ixx34	Integration mode
Ixx35	Acceleration feedforward
Ixx61	Current loop integral gain
Ixx62	Current loop forward proportional gain
Ixx76	Current loop backward proportional gain

Table 8: Current and position loop tuning I variables

I variables - servo motors

I variable	Description
`Ixx70`	Number of commutation cycles
`Ixx71`	Counts per `Ixx70` commutation cycles
`Ixx73`	Phase finding output value
`Ixx74`	Phase finding time
`Ixx75`	Phase position offset

Table 9: Servo motor I variables

I variables - absolute encoders

I variable	Description
`Ixx10`	Power-on servo position address
`Ixx81`	Power-on phase position address
`Ixx91`	Power-on phase position format
`Ixx95`	Power-on servo position format

Table 10: Servo motor I variables