

6 The Servo Loop

6.1 Overview

For the servo-based chipsets (MC2100, MC2300, MC2800, MC3110, MC3310, MC58000) a positional servo loop is used as part of the basic method of determining the motor command output. Chapter 4 made references to the PID loop that exists in PMD's servo based products. It was mentioned that the output of the PID represents a desired motor torque. The function of the servo loop is to match as closely as possible the commanded position, which comes from the trajectory generator, and the actual motor position.

To accomplish this the profile generator's *commanded value* is combined with the actual encoder position to create a position error, which is then passed through a digital PID-type servo filter. The scaled result of the filter calculation is the *motor command* (also referred to as the torque signal), which is output as either a PWM signal to the motor amplifier, or a 16-bit input to a D/A Converter. In the context of multiphase motors, the *motor command* is broken down into components (phases) based on the current commutation angle before being sent to the amplifier.

6.2 PID loop algorithm

The servo filter used with the servo-based chipsets is a proportional-integral-derivative (PID) algorithm, with velocity and acceleration feed-forward terms and an output scale factor. An integration limit provides an upper bound for the accumulated error. An optional bias value can be added to the filter calculation to produce the final motor output command. A limiting value for the filter output provides additional constraint. This limit is set using the command `SetMotorLimit`.

The PID+V_{ff}+A_{ff} formula, including the scale factor and bias terms, is as follows:

$$\text{Output}_n = \left[K_p E_n + K_d (E_k - E_{(k-1)}) + \sum_{j=0}^n E_j \times Ki / 256 + K_{vff} (CmdVel / 4) + K_{aff} (CmdAccel \times 8) \right] \times K_{out} / 65536 + \text{Bias}$$

where	E _n	are the accumulated error terms
	K _I	is the Integral Gain
	K _d	is the Derivative Gain
	K _p	is the Proportional Gain
	K _{aff}	is the Acceleration feed-forward
	K _{vff}	is the Velocity feed-forward
	Bias	is the DC motor offset
	K _{out}	is the scale factor for the output command.

All filter parameters, the motor output command limit, and the motor bias are programmable, so that the filter may be fine-tuned to any application. The parameter ranges, formats and interpretations are shown in the following table:

Term	Name	Representation & Range
I_{lim}	Integration Limit	unsigned 32 bits (0 to 2,147,483,647)
K_i	Integral Gain	unsigned 16 bits (0 to 32767)
K_d	Derivative Gain	unsigned 16 bits (0 to 32767)
K_p	Proportional Gain	unsigned 16 bits (0 to 32767)
K_{aff}	Acceleration feed-forward	unsigned 16 bits (0 to 32767)
K_{vff}	Velocity feed-forward	unsigned 16 bits (0 to 32767)
K_{out}	Output scale factor	unsigned 16 bits (0 to 32767)
Bias	DC motor offset	signed 16 bits (-32768 to 32,767)
	Motor command limit	unsigned 16 bits (0 to 32767)

The structure of the digital filter is shown in Figure 6.1.

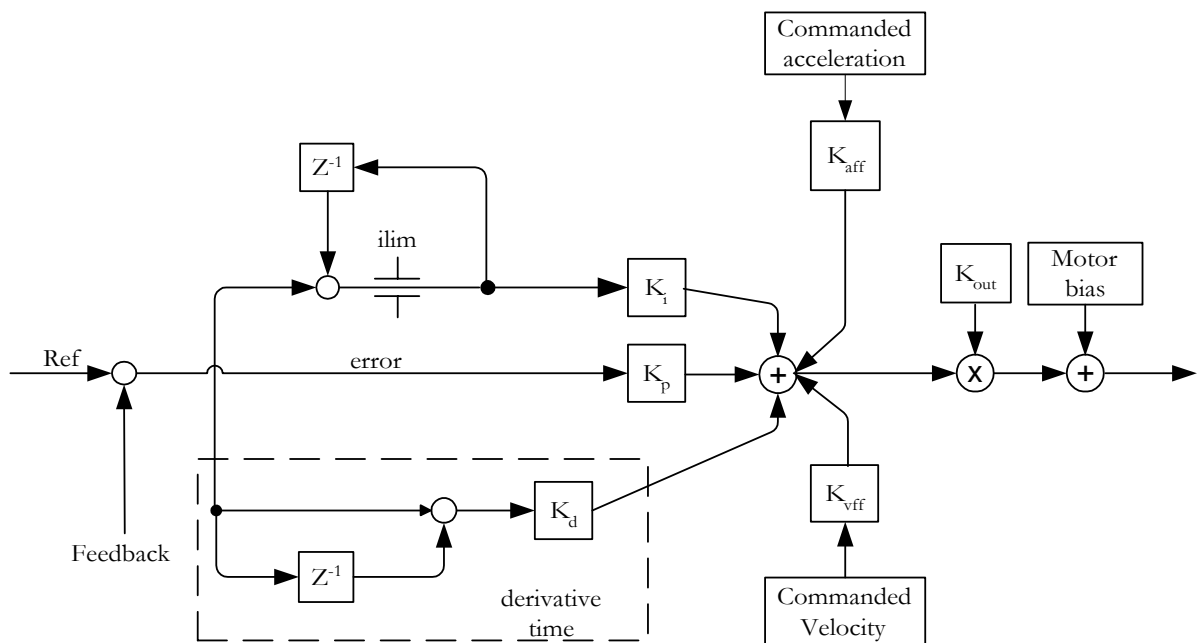


Figure 6.1 Digital Servo Filter

6.3 Motor bias

When an axis is subject to a net external force in one direction (such as a vertical axis pulled downward by gravity), the servo filter can compensate for it by adding a constant DC bias to the filter output. The bias value is set using the host instruction **SetMotorBias**. It can be read back using the command **GetMotorBias**.

6.4 Output scaling

The K_{out} parameter can be used to scale down the output of the PID filter in situations that require it. It does this by multiplying the filter result by $K_{out}/65536$. It has the effect of

increasing the usable range of K_p , which is typically programmed in the 1 to 150 range when no output scaling is done. The K_{out} value is set using the host instruction **SetKout**. It can be read back using the command **GetKout**.

6.5 Output limit

The motor output limit prevents the filter output from exceeding a boundary magnitude in either direction. If the filter produces a value greater than the limit, the motor command takes the limiting value. The motor limit value is set using the host instruction **SetMotorLimit**. It can be read back using the command **GetMotorLimit**.

The motor limit applies only in closed-loop mode. It does not affect the motor command value set by the host in open-loop mode.