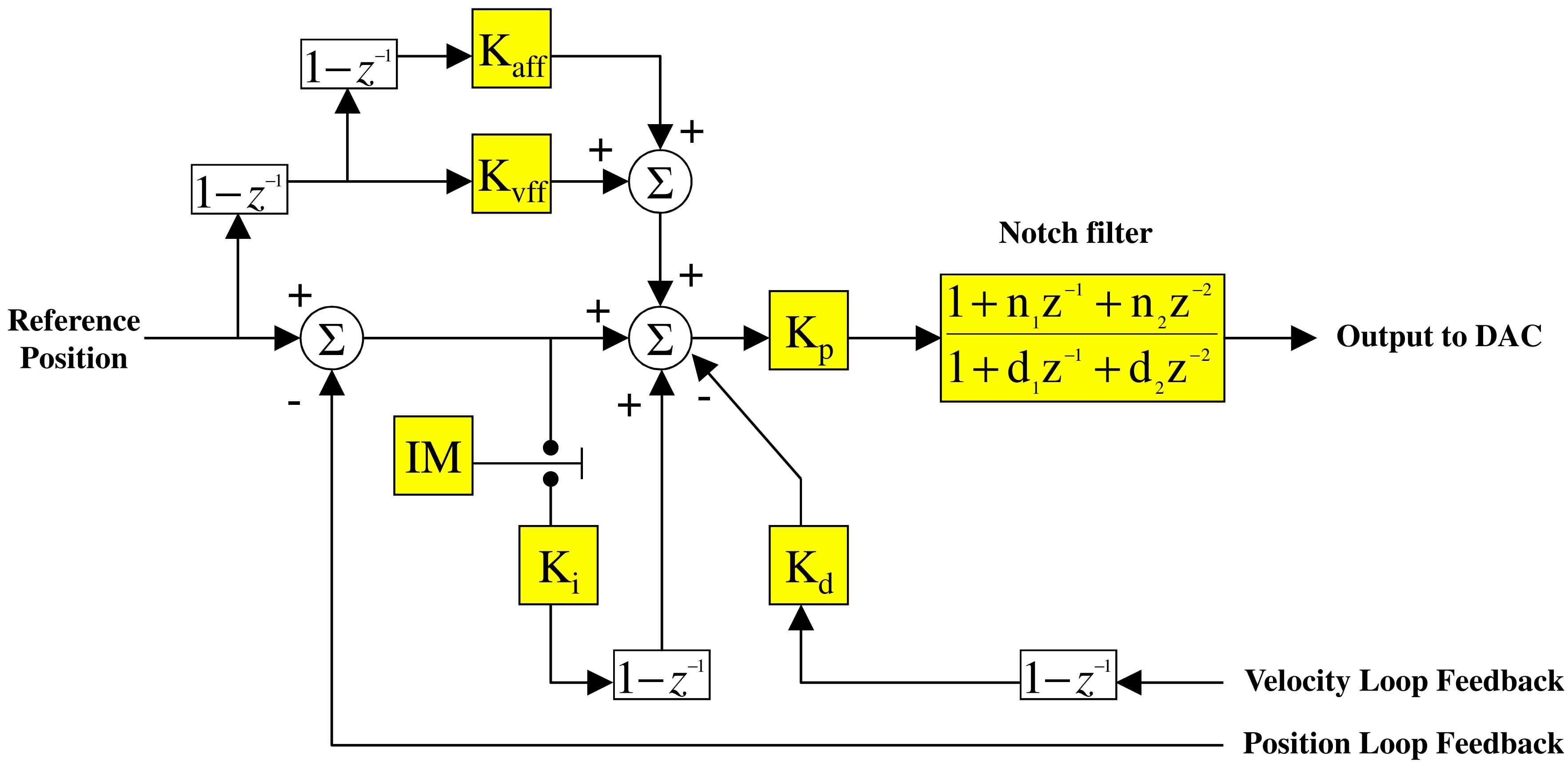
PMAC Servo Control Algorithm

PMAC PID and Notch Filter



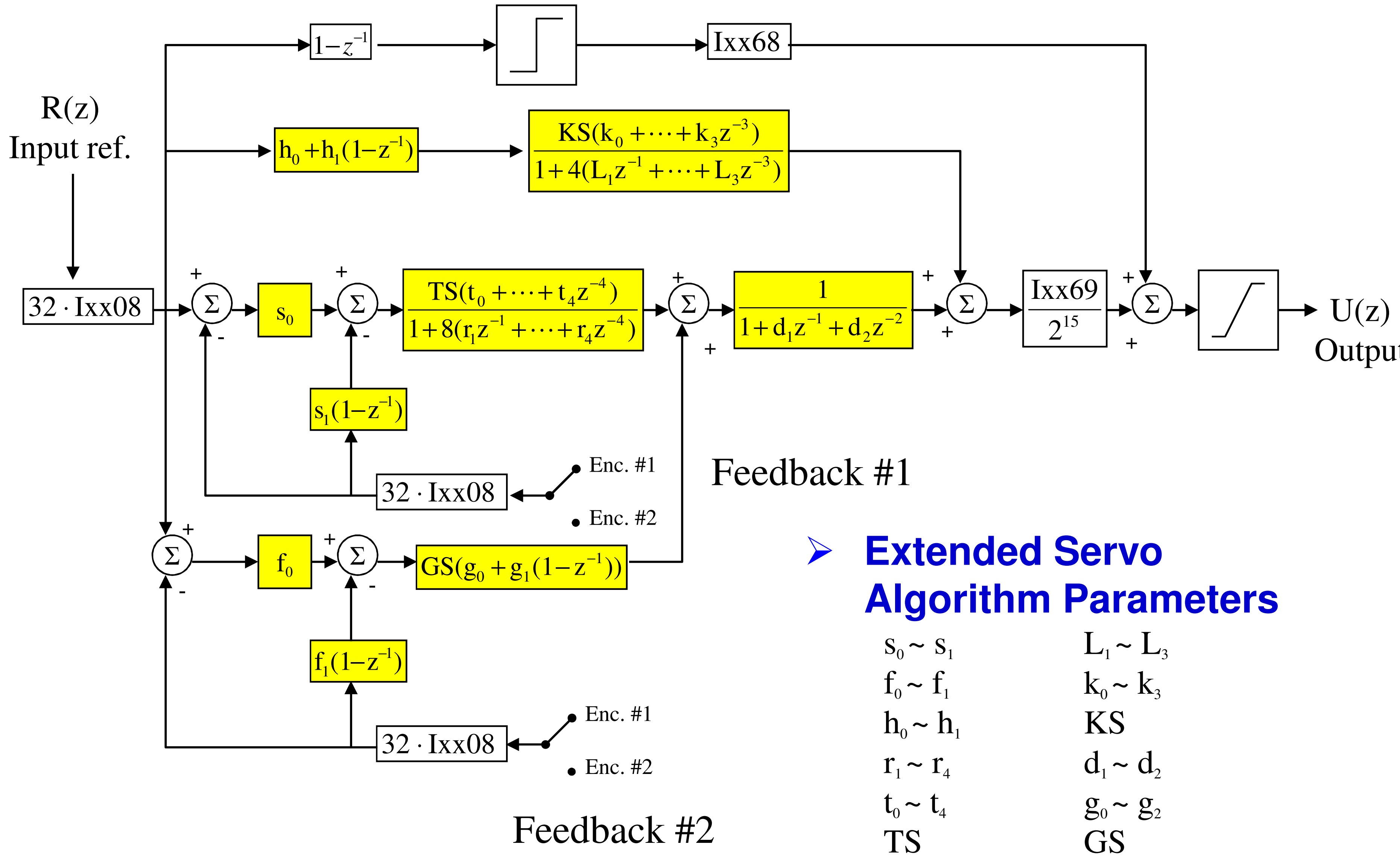
- **PID Parameters**

K_p	Ixx30, Proportional gain
K_d	Ixx31, Derivative gain
K_{vff}	Ixx32, Velocity feedforward gain
K_i	Ixx33, Integral gain
IM	Ixx34, Integration mode
K_{aff}	Ixx35, Acceleration feedforward gain

- **Notch Filter Coefficients**

n_1	Ixx36
n_2	Ixx37
d_1	Ixx38
d_2	Ixx39

PMAC Extended Servo Algorithm

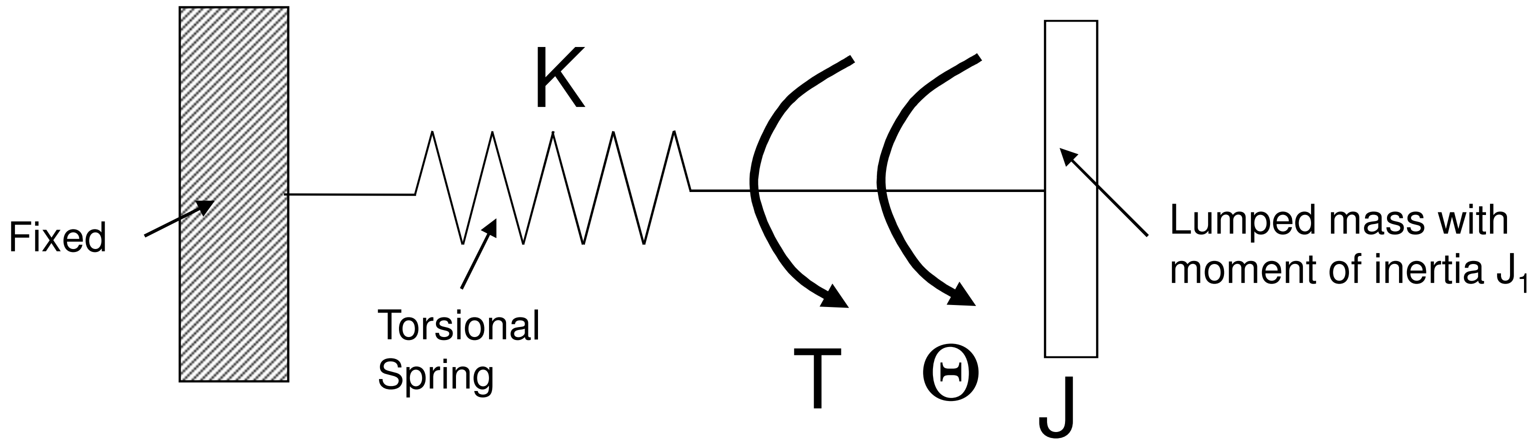


- **Extended Servo Algorithm Parameters**

$s_0 \sim s_1$	$L_1 \sim L_3$
$f_0 \sim f_1$	$k_0 \sim k_3$
$h_0 \sim h_1$	KS
$r_1 \sim r_4$	$d_1 \sim d_2$
$t_0 \sim t_4$	$g_0 \sim g_2$
TS	GS

Note: Must be tuned by an external algorithm (e.g. pole placement).

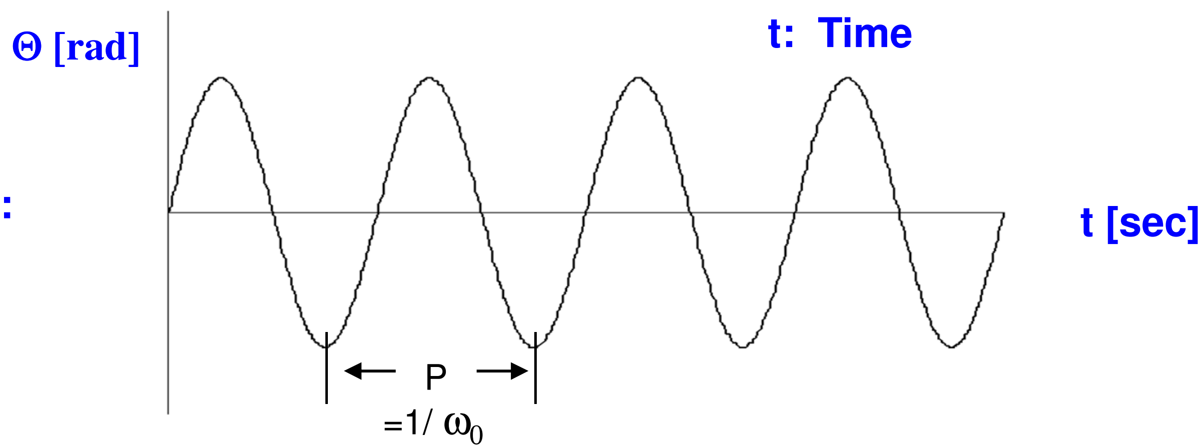
Proportional Control



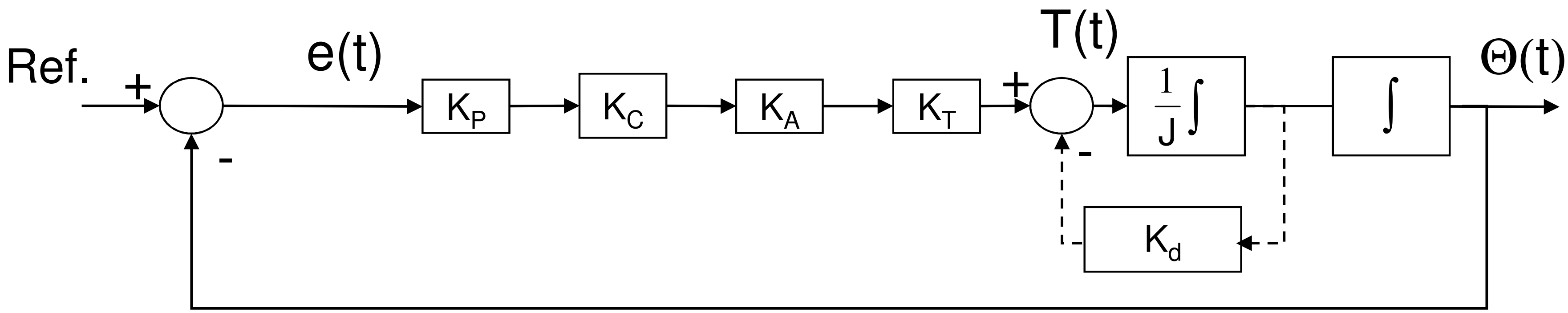
$$T = K \bullet \Theta = -J\frac{d^2\Theta}{dt^2},\ \ \omega_0 = \sqrt{\frac{K}{J}},\ \ \zeta = 0$$

ζ : Damping Ratio
ω_0 : Natural Frequency
Θ : Angular Displacement
T : Input Torque
K: Spring Constant
J: Moment of Inertia
P: Period of Oscillation
t: Time

Example Response:



Proportional Control



$$T(t) = K_p \bullet K_C \bullet K_A \bullet K_T \bullet e(t) = -K_p \bullet K_C \bullet K_A \bullet K_T \bullet \Theta(t) = J \bullet \frac{d^2\Theta}{dt^2}$$

K_C: DAC Conversion Gain
K_A: Amplifier Gain
K_T: Motor Torque Constant

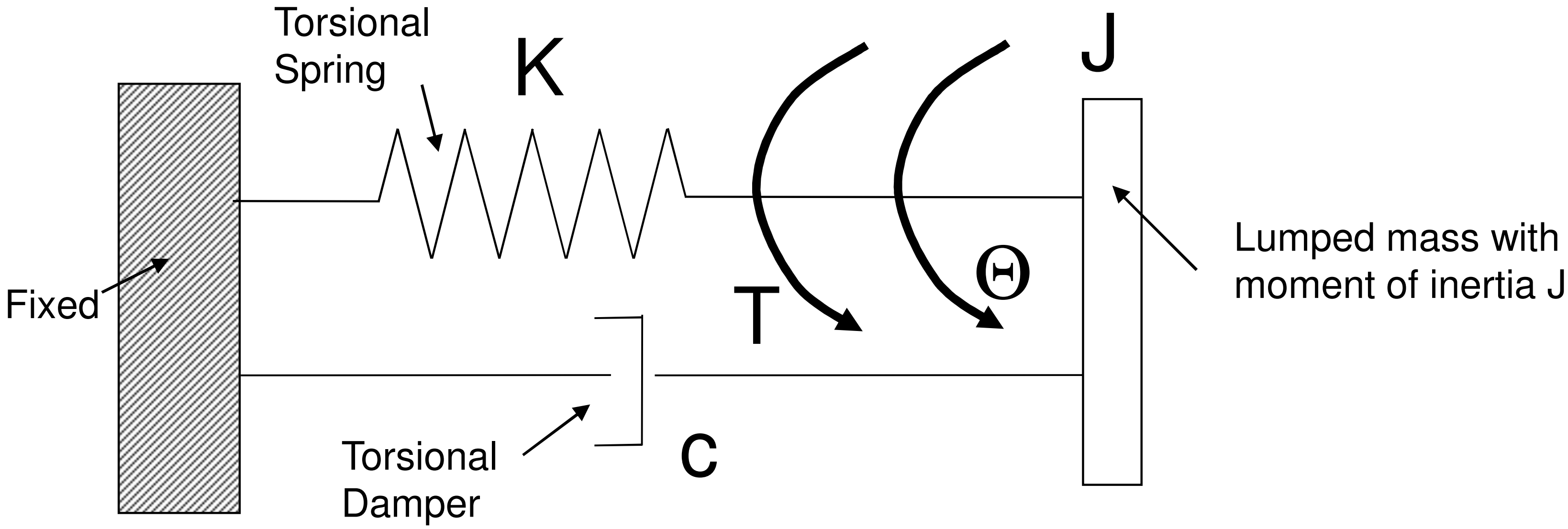
$$\omega_0 = \sqrt{\frac{K_p \bullet K_C \bullet K_A \bullet K_T}{J}}, \quad \zeta = 0$$

This is an <u>undamped</u> Simple Harmonic Oscillator (SHO)

Thus, the proportional gain K_P correlates with spring stiffness.

Higher K_P → higher stiffness

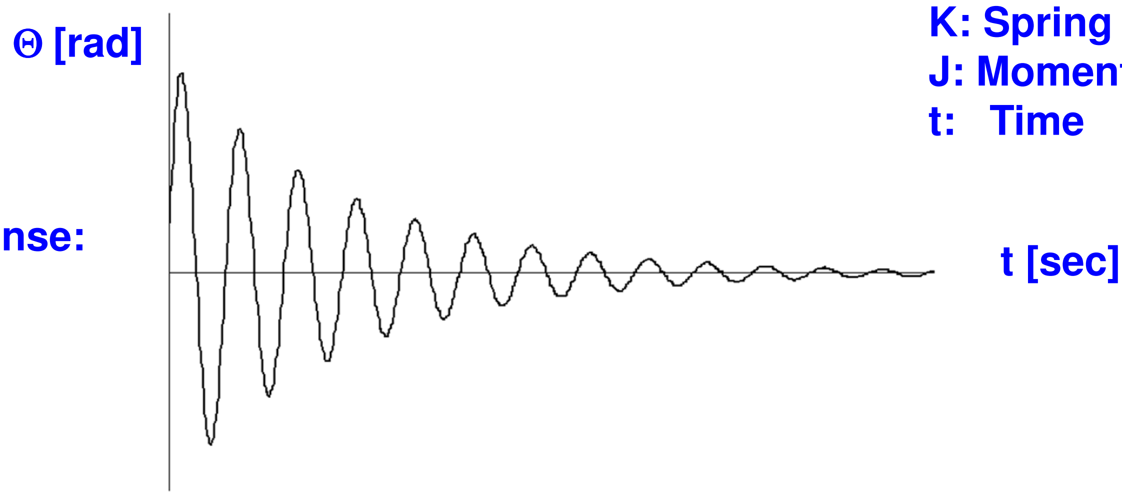
Derivative Control

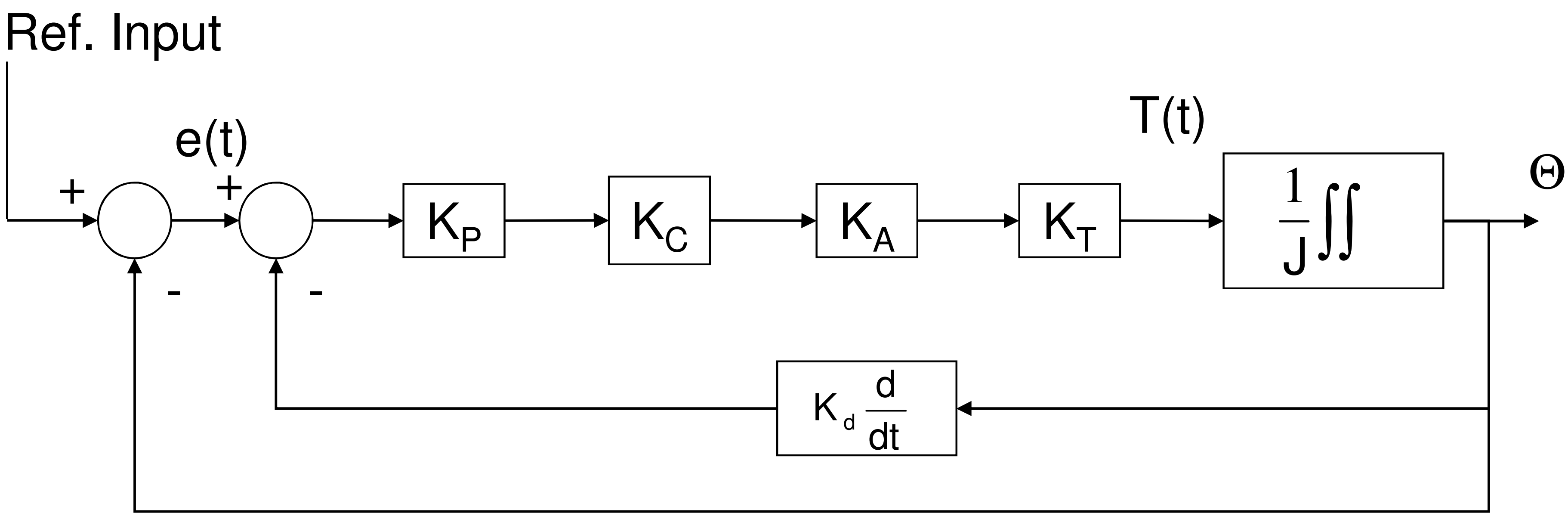


$$T = K \bullet \Theta + c \bullet \frac{d\Theta}{dt} = -J \bullet \frac{d^2\Theta}{dt^2}, \quad \omega_0 = \sqrt{\frac{K}{J}}, \quad \zeta = \frac{c}{2}\sqrt{KJ}$$

ζ : Damping Ratio
ω_0 : Natural Frequency
Θ : Angular Displacement
T : Input Torque
C: Damping Coefficient
K: Spring Constant
J: Moment of Inertia
t: Time

Example Response:





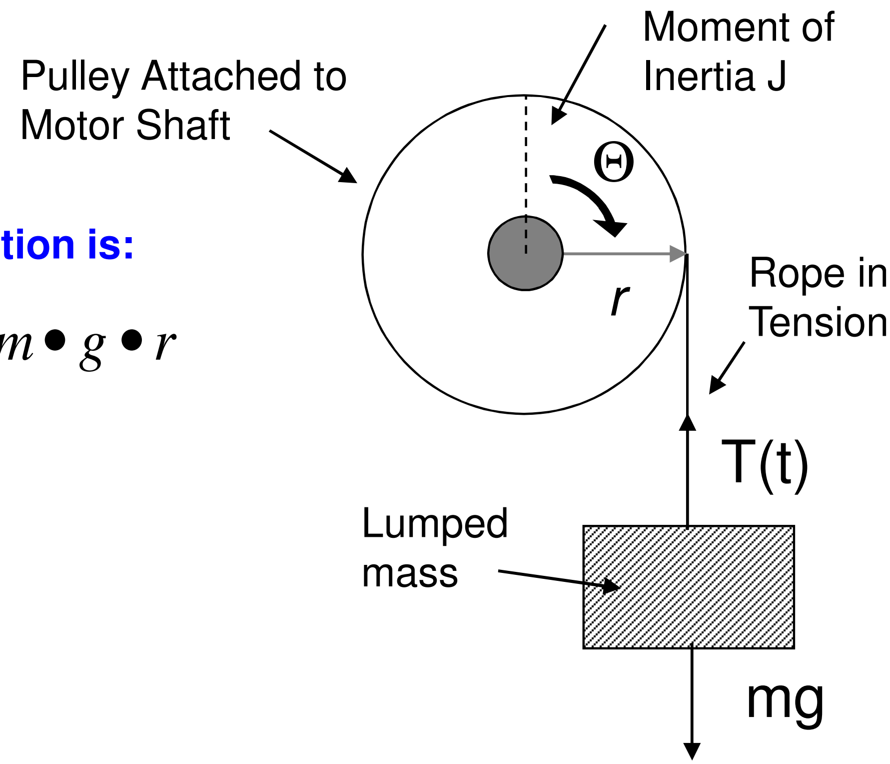
$$T(t) = -K_p \bullet K_C \bullet K_A \bullet K_T \bullet \Theta(t) - K_p \bullet K_C \bullet K_A \bullet K_d \bullet \frac{d\Theta}{dt} = J \bullet \frac{d^2\Theta}{dt^2}$$

This is a damped SHO.

$$\omega_0 = \sqrt{\frac{K_p \bullet K_C \bullet K_A \bullet K_T}{J}}, \quad \zeta = \frac{K_d}{2}\sqrt{\frac{K_p \bullet K_C \bullet K_A \bullet K_T}{J}}$$

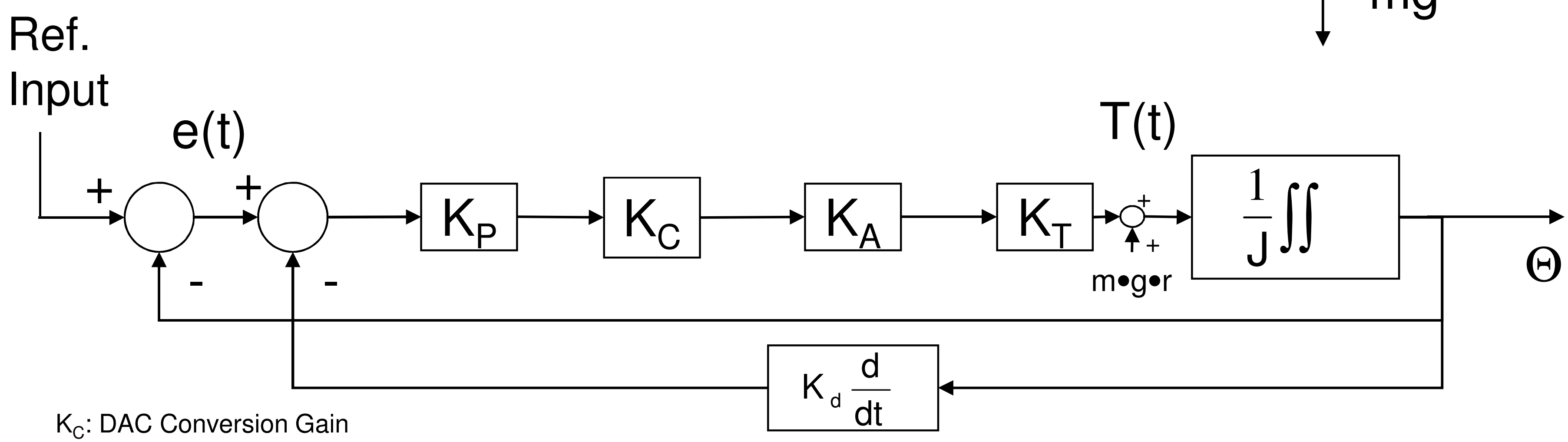
K_C: DAC Conversion Gain
K_A: Amplifier Gain
K_T: Motor Torque Constant

Integral Control



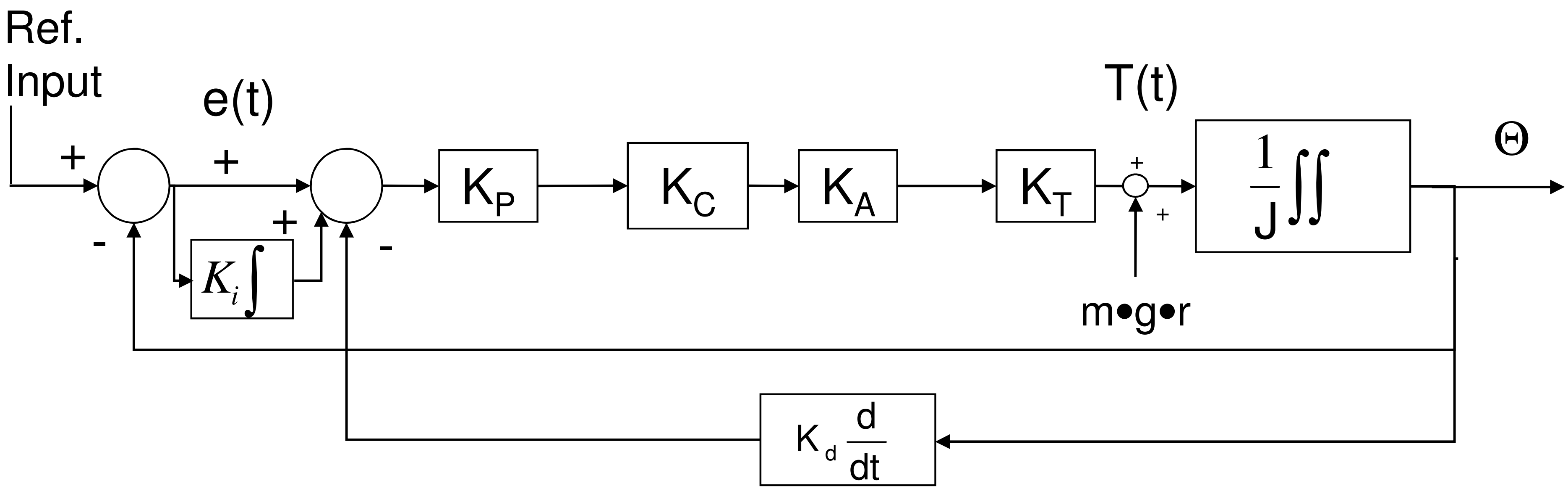
Without the integral, the governing equation is:

$$T(t) = (K_P \bullet K_C \bullet K_A \bullet K_T) \bullet e(t) = m \bullet g \bullet r$$



K_C: DAC Conversion Gain
K_A: Amplifier Gain
K_T: Motor Torque Constant

Adding the Integral



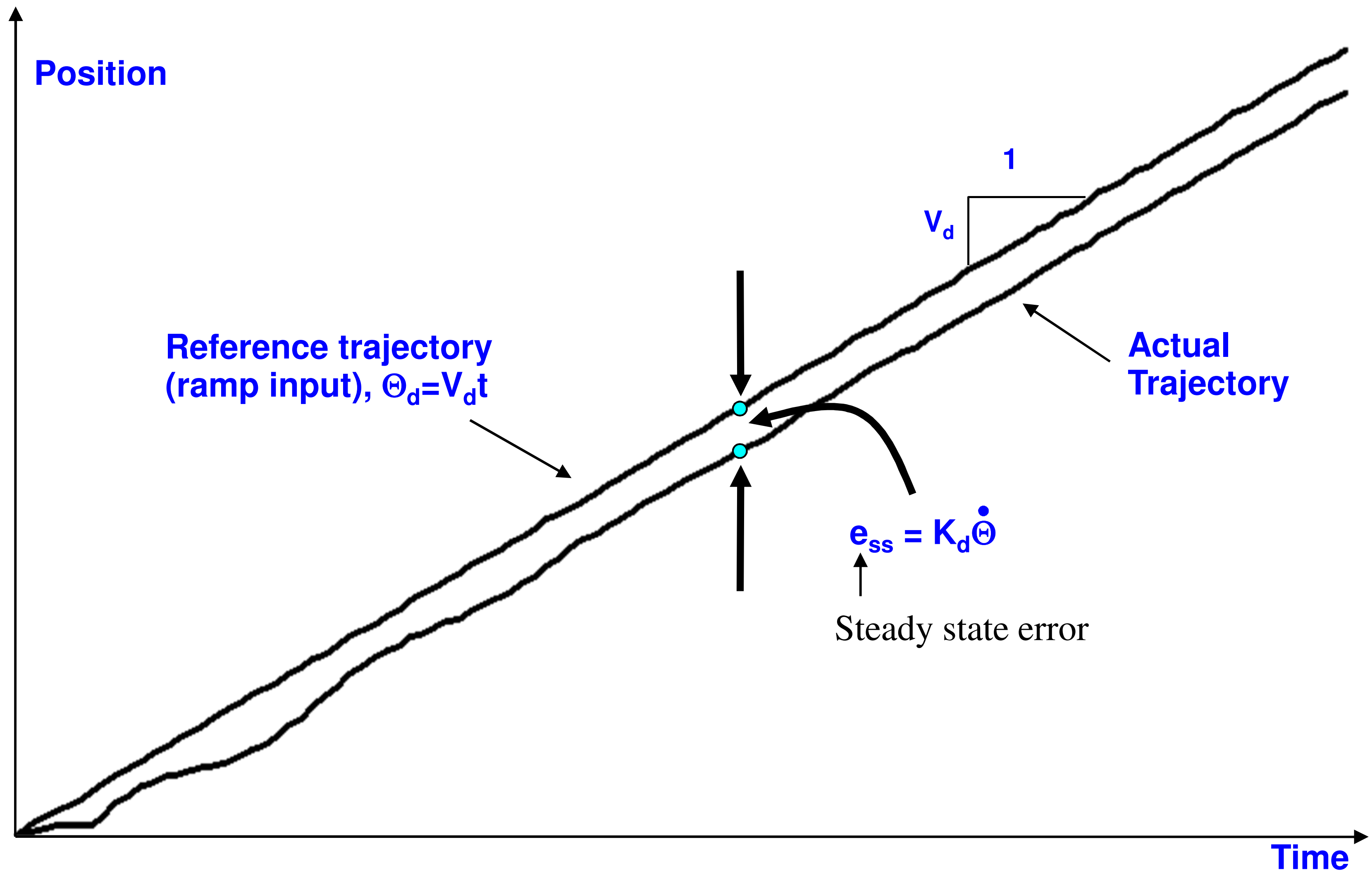
$$T(t) = (K_P \bullet K_C \bullet K_A \bullet K_T) \bullet [K_i \int e(t)dt + e(t) + K_d \frac{d\Theta(t)}{dt}] = m \bullet g \bullet r$$

Therefore as t → ∞, e(t)→ 0

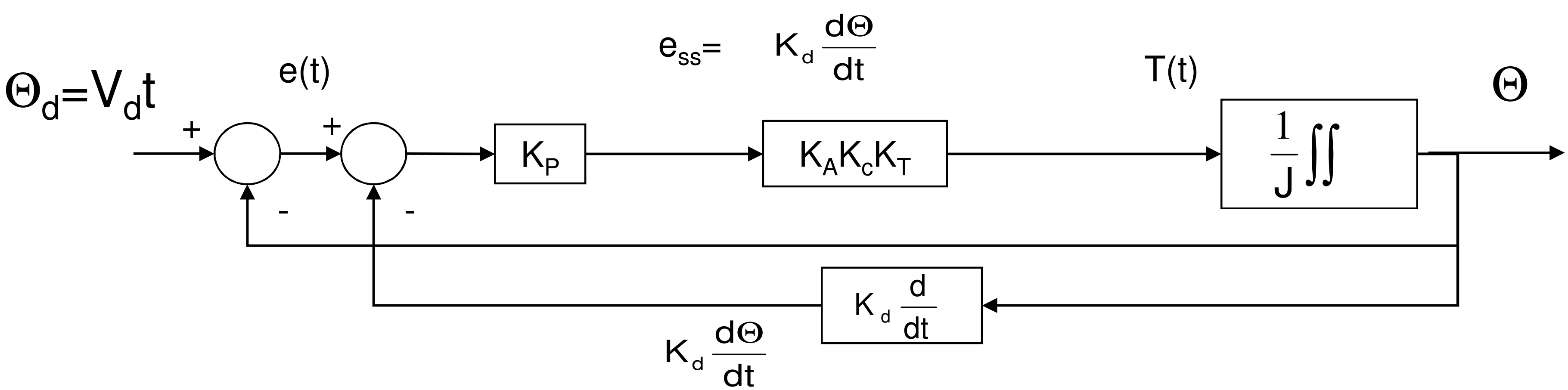
K_C: DAC Conversion Gain
K_A: Amplifier Gain
K_T: Motor Torque Constant

Steady State Errors Due to Constant Speed Trajectory Tracking

Response (assuming friction = 0):



Block Diagram without K_{vff}

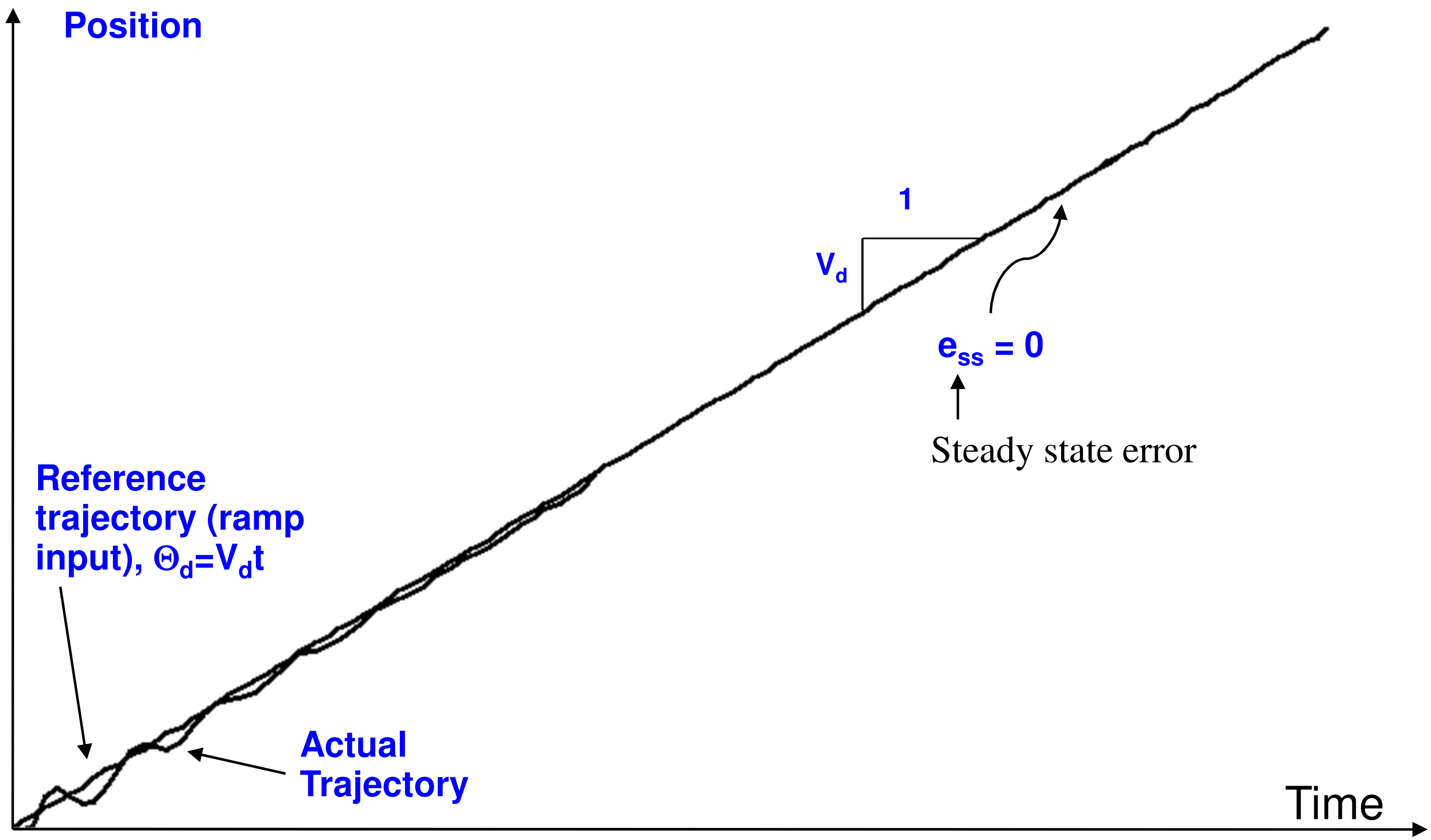


K_C: DAC Conversion Gain
K_A: Amplifier Gain
K_T: Motor Torque Constant

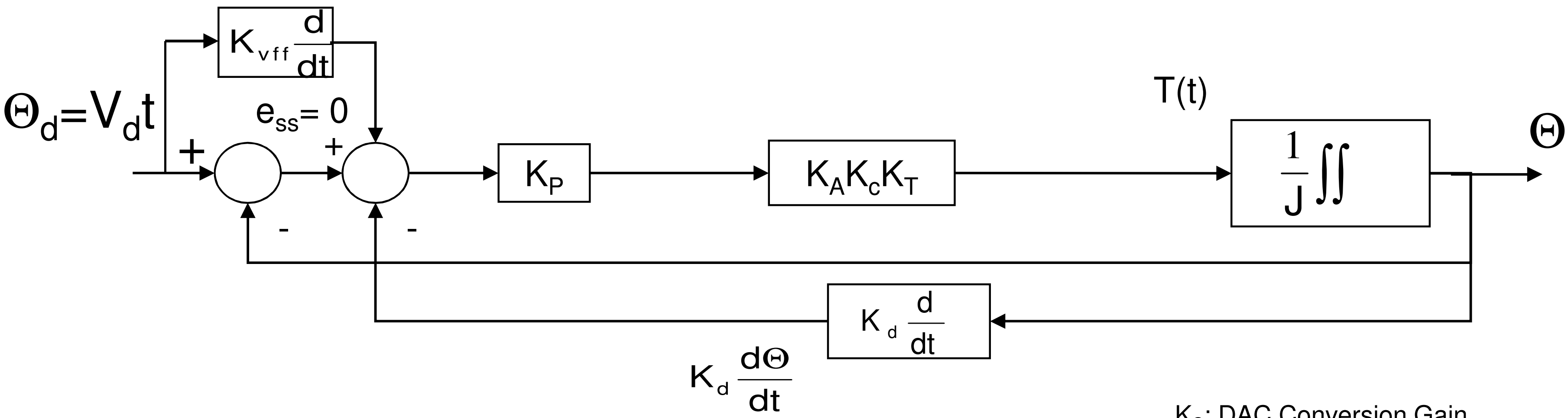
Purpose of Velocity Feedforward

Selecting $K_{vff} = K_d \rightarrow e_{ss} = 0$

No steady state error due to constant velocity tracking



Block Diagram with K_{vff}



K_C: DAC Conversion Gain
K_A: Amplifier Gain
K_T: Motor Torque Constant

Note

Typically, setting $K_{vff} = K_d$ is a good starting place for tuning the servo loop.

Purpose of Acceleration Feedforward

- In general, the trajectories contain higher order time functions:

$$\Theta_d(t) = c_0 + c_1 \bullet t + c_2 \bullet t^2$$

- Example: Constant Jerk Trajectory
  By choosing:

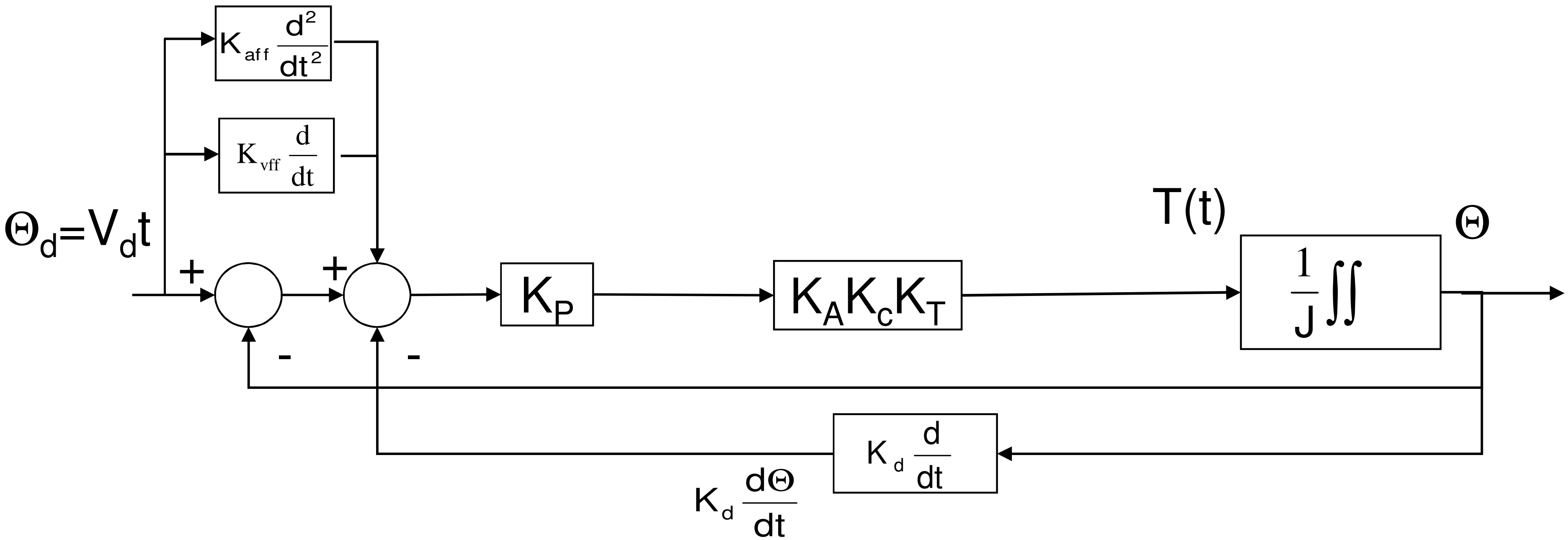
$$K_{aff} = \frac{J_1}{K_P \bullet K_C \bullet K_A \bullet K_T} \quad \rightarrow \quad e(t) = 0$$

$$T(t) = \left(\frac{J_1}{K_P \bullet K_C \bullet K_A \bullet K_T}\right)\frac{d^2\Theta_d}{dt^2}(K_P \bullet K_C \bullet K_A \bullet K_T) = \frac{J_1 \bullet d^2\Theta_d}{dt^2}$$

- This results in no following error for the *Ideal* system; i.e.:

- Since $\dfrac{d^2\Theta_d}{dt^2} = \dfrac{d^2\Theta}{dt^2} \quad \rightarrow \quad \Theta_d = \Theta \quad \rightarrow$ No Following Error

Acceleration Feedforward: K_{aff}



K_C: DAC Conversion Gain
K_A: Amplifier Gain
K_T: Motor Torque Constant

Servo Loop Modifier I-Variables

Ixx59: Motor xx User-Written Servo/Phase Enable

= 0: Use standard PID phase algorithms
= 1: Use custom servo, standard phase algorithms
= 2: Use standard PID, custom phase algorithms
= 3: Use custom servo, phase algorithms

Ixx60: Motor xx Servo Cycle Extension Period [Servo Cycles]

Loop closed every (Ixx60+1) servo interrupts
Useful for slow, low-resolution axes
Useful for process control "axes"

Ixx63: Motor xx Integration Limit [(Ixx33 / 2^{19}) counts * servo cycles]

Maximum integrated error accumulated
When negative, killed on saturation

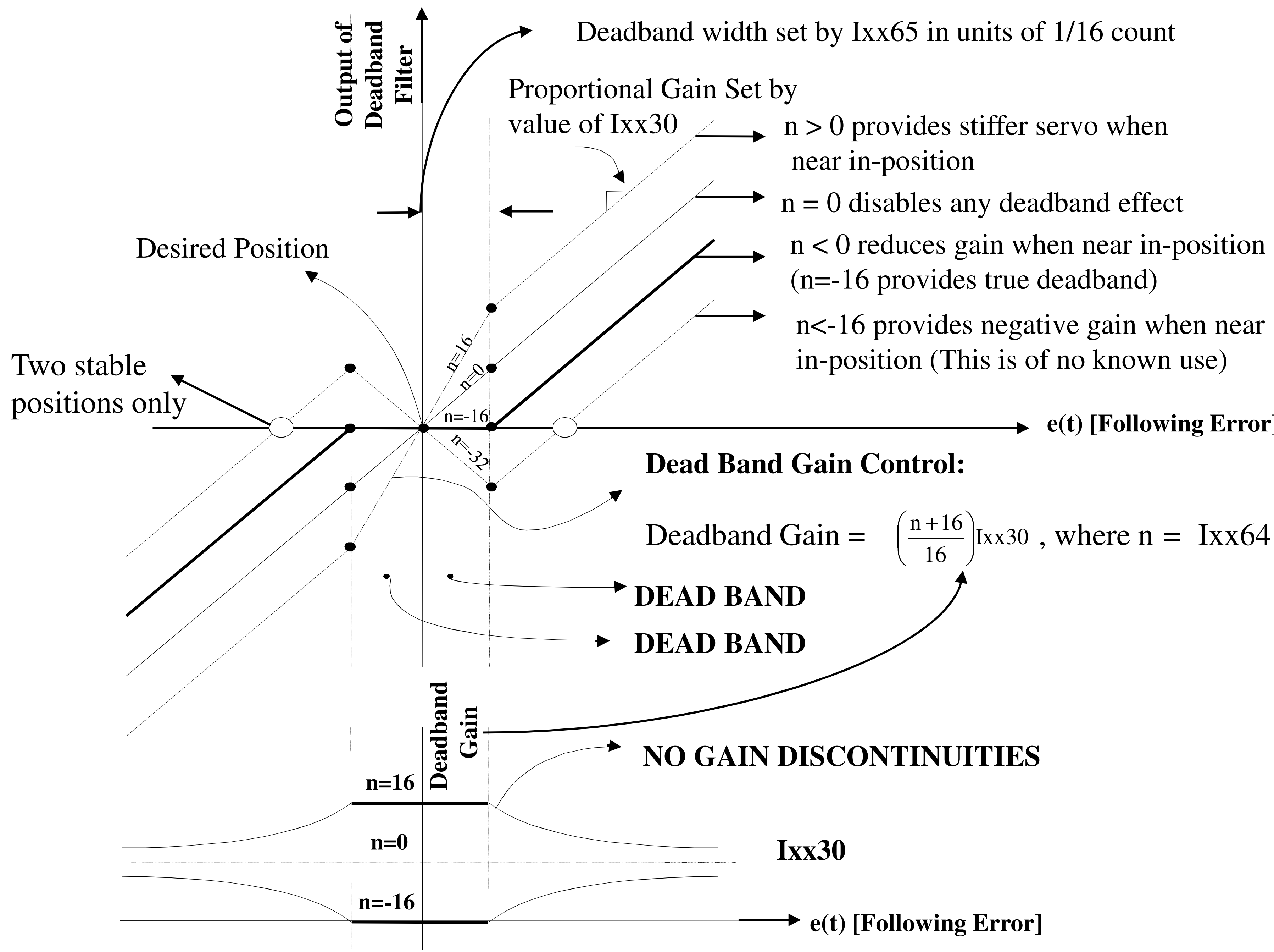
Ixx64: Motor xx Deadband Gain Factor

Controls the effective gain within the deadband zone

Ixx65: Motor xx Deadband Size [1/16 count]

Defines the size of the position error band, measured from zero error, within which there will be changed or no control effort, for the PMAC feature known as deadband compensation.

Deadband Compensation



Servo Loop Modifier I-Variables (cont.)

Ixx67: **Motor xx Position Error Limit [1/16 Count]**

Limits error that filter "sees"

Ixx68: **Motor xx Friction Feedforward [16-bit DAC Bits]**

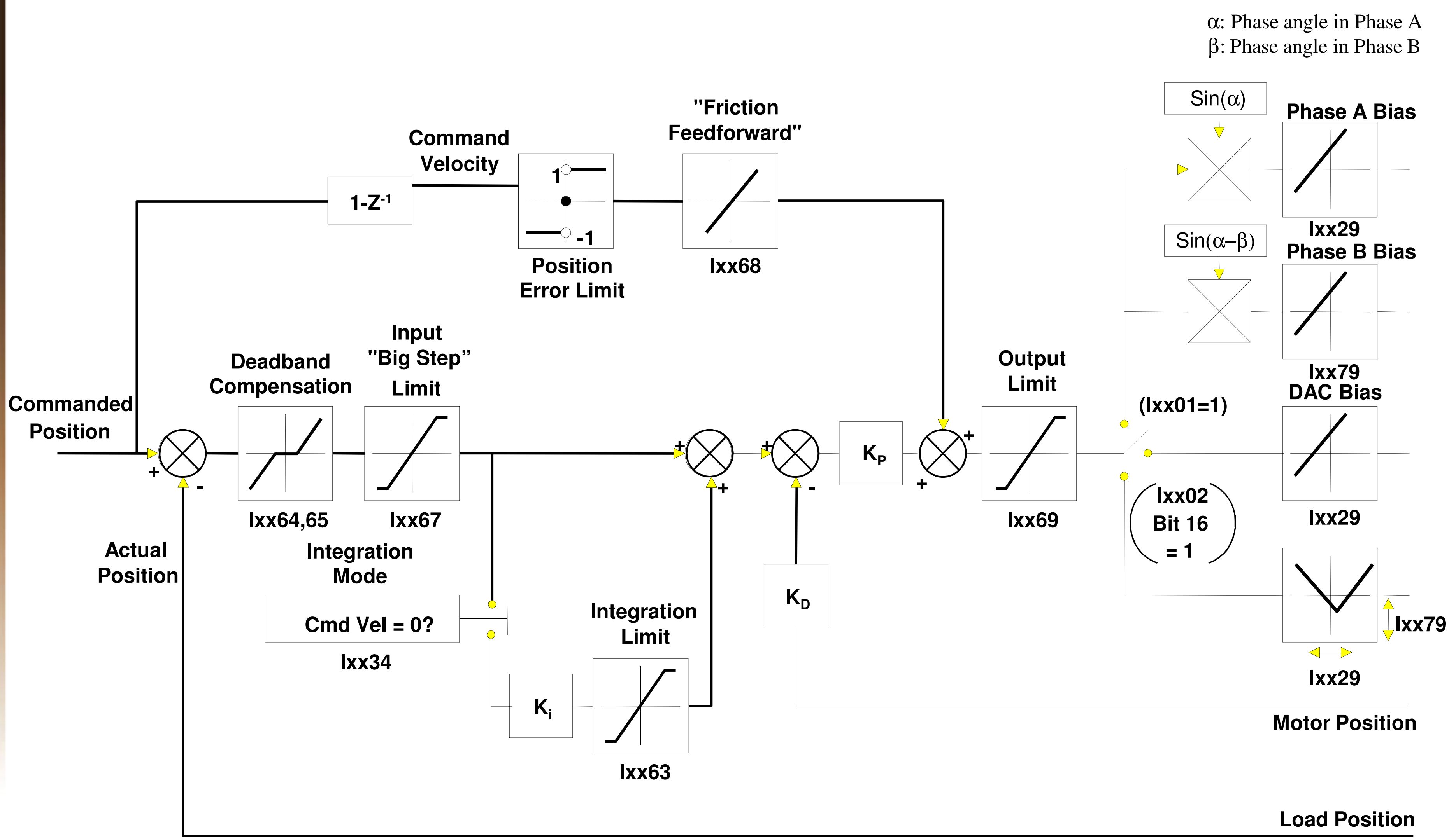
Compensates for dry (Coulumb) friction

Ixx69: **Output Command (DAC) Limit [16-bit DAC Bits]**

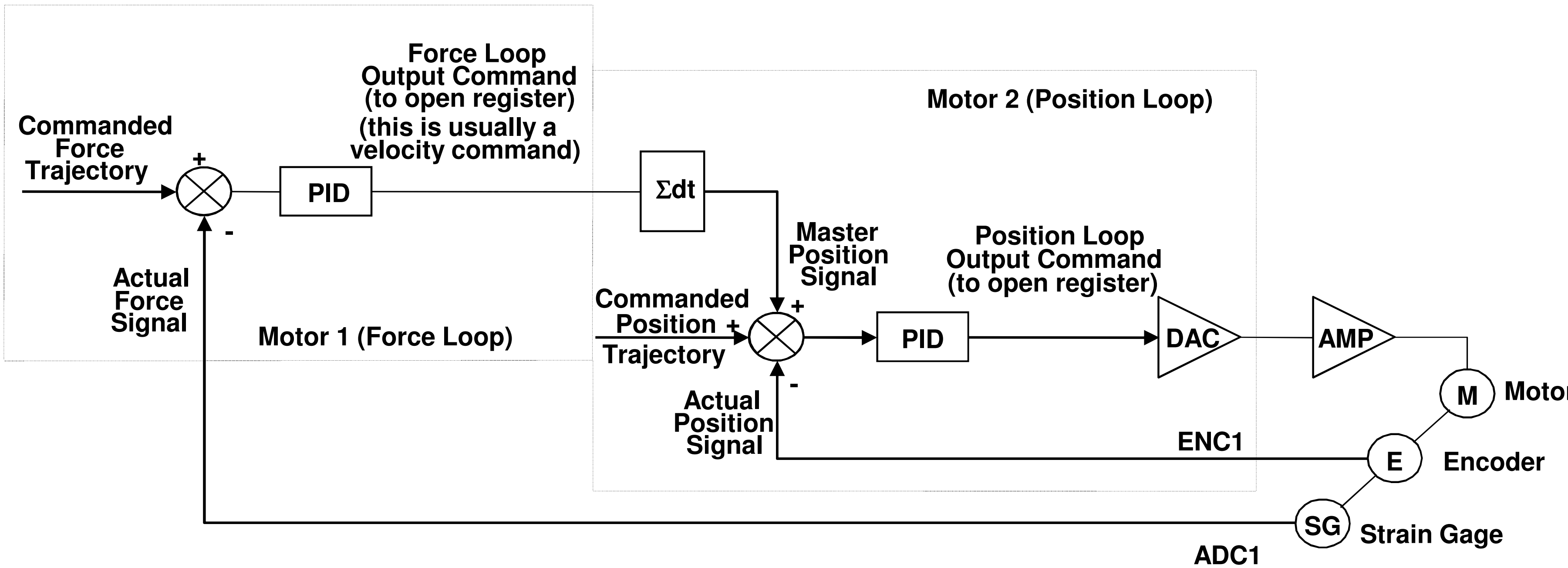
Limits output of filter

Acts as torque (current) limit if commutating

Servo Loop Modifiers



Closing a Force Loop around a Position Loop



Note

See the "Cascading Servo Loops" section of the Turbo PMAC User Manual for detailed information on how to configure this type of hybrid control.