CHAPTER ④

Tuning

Section Objectives

The information in this section will enable you to:

- ❑ Understand the role and interaction of each gain parameter
- ❑ Tune your system for optimal performance
- ❑ Configure an In Position Window to determine move completion

Tuning and Performance

The OEM070 uses a digital *Proportional Integral Derivative* (PID) filter to compensate the control loop. For best performance, you must tune the filter's parameters. A properly tuned system will exhibit smooth motor rotation, accurate tracking, and fast settling time.

All tuning is performed via RS232-C communications.

PID Tuning

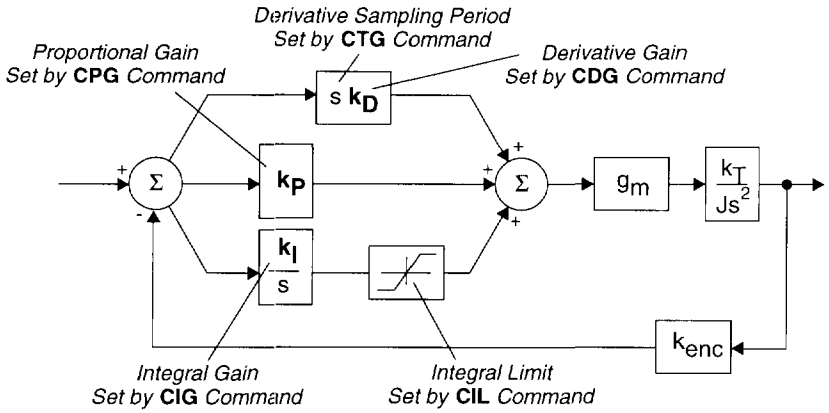
In the procedure described below, you will systematically vary the tuning parameters until you achieve a move that meets your requirements for accuracy and response time.

The OEM070 generates a move profile based upon the user supplied acceleration, velocity, and distance commands (**A**, **V**, and **D**). At each servo sampling period (every 266 microseconds), the OEM070 calculates the position the motor should reach as it follows the move profile. This is called *commanded position*, and is one of two inputs to a summing node. Position

information from the encoder, which is called *actual position*, is the other input to the summing node. During a typical move, actual position will differ from commanded position by at least a few encoder counts. When actual position is subtracted from commanded position at the summing node, an error signal is produced. The error signal is the input to the PID filter.

The position specified by the distance command (**D**) is called the *target position*. During the move, commanded position is not the same as target position. The commanded position is incremented each sampling period. When it finally matches the target position, the move is over.

The servo block diagram is shown below.



As the figure shows, you can adjust five different parameters to tune the PID filter. The relevant commands are:

CPG	Configure Proportional Gain
CDG	Configure Derivative Gain
CTG	Configure Derivative Sampling Period
CIG	Configure Integral Gain
CIL	Configure Integral Limit

To tune the system, you will iteratively increase **CDG** and **CTG** to their optimum values, then increase **CPG** to its optimum value. If necessary, you will also increase **CIG** and **CIL**.

In general, you will set **CDG** and **CPG** as high as possible, and **CIG** as low as possible. Trade-offs between response time, stability, and final position error will dictate the values you

select. For loads that vary during operations, you can download new parameters by using buffered versions of the five tuning commands (**BCPG, BCDG, BCTG, BCIG, BCIL**).

WARNING

During servo tuning, the system can undergo accidental and violent movement due to improper gain settings and programming errors. Please use extreme caution while prototyping

Each tuning parameter is described in the following sections.

CPG – Proportional Gain

Proportional gain provides a torque that is directly proportional to the *magnitude* of the error signal. Proportional gain is similar to a spring—the larger the error, the larger the restoring force. It determines the stiffness of the system and affects the following error. High proportional gain gives a stiff, responsive system, but can result in overshoot and oscillation. Damping—provided by derivative gain—can reduce this overshoot and oscillation.

CDG – Derivative Gain; CTG – Derivative Sampling Period

Derivative gain provides a torque that is directly proportional to the *rate of change* of the error signal. The previous error is subtracted from the present error each sampling period. The difference represents the error's instantaneous rate of change, or *derivative*. The difference is multiplied by the value set by the **CDG** command, and the product contributes to the motor control output.

Derivative gain opposes rapid changes in velocity. It will dampen the resonance effects of proportional gain. With higher derivative gain, you can use higher proportional gain.

You can use the **CTG** command to make the derivative sampling period longer than the system's sampling period. The system sampling period—266 µsec—is the period between updates of position error, and cannot be changed. The derivative sampling period is an integer multiple of the system sampling period. It can range from 266 µsec to 68 msec, in increments of 266µsec (for example: CTG0 = 266 µs, CTG1 = 532 µs, CTG2 = 798 µs, etc.).

With a longer derivative sampling period, more time elapses between derivative error measurements. The difference be-

tween previous and present error is still multiplied by the CDG value. The product contributes to the motor control output every *system* sampling period, but is only updated every *derivative* sampling period. This gives a more constant derivative term and improves stability. Low velocity systems in particular can benefit from a longer sampling period.

Because of stability considerations, however, the derivative sampling period should be no longer than one tenth of the system mechanical time constant. This means many systems must have low values of **CTG**.

CIG – INTEGRAL GAIN; CIL – INTEGRAL LIMIT

Integral gain provides a torque that is directly proportional to the sum, over time, of the error values—the *integral* of the error. The controller reads the error value every sampling period, and adds it to the sum of all previous error values. The sum is multiplied by the value set by the **CIG** command, and the product contributes to the motor control output every system sampling period.

Integral gain can remove steady state errors that are due to gravity or a constant static torque. Integral gain can also correct velocity lag that can occur in a constant velocity system.

If error persists during a move, the sum of the error values may be quite high at the end of the move. In this case, the torque provided by the integral gain can also be very high, and can cause an overshoot. This effect is called *integral windup*. You can use **CIL**, the integral limit command, to set a maximum value for integral gain. The integral limit constrains the integral term to values less than or equal to **CIL**, which will reduce the overshoot caused by integral windup.

Tuning Procedure

You can manually tune the OEM070 by varying the tuning parameters while you empirically evaluate the system response. This manual method works well in most applications.

TUNING PROCEDURE

You will achieve best results by making a consistent, repetitive move that is representative of your application.

1 ***Issue a RETURN TO FACTORY SETTINGS command (RFS)***
The RFS command will reset the gains to their default values (CDG240, CTG0, CPG16, CIG2, CIL2, CPE4000)

2 ***Decrease CPG***
Decrease **CPG** to zero (**CPG0**). If your system has very little friction, internal offsets in the drive may cause the motor to run away (spin faster and faster) with **CPG** set to zero. If the motor runs away, issue an **OFF** command. When the motor stops, increase **CPG** by one unit (**CPG1**), and issue an **ON** command. If the motor continues to run away, repeat this procedure—incrementing **CPG** by one unit—until the motor remains stopped.

3 ***Decrease CIG***
Set the integral gain to zero (**CIG0**).

4 ***Increase Derivative Gain (CDG) and Derivative Sampling Period (CTG)***
Determine **CDG** and **CTG** iteratively. Increase **CDG** until the shaft begins high frequency oscillations, then increase **CTG** by one. With a higher **CTG**, the oscillations should be damped. Again increase **CDG** until oscillations occur, then increase **CTG** by one. Repeat this process until **CTG** reaches a value appropriate for the system.

In general, you will want values for **CDG** and **CTG** that are as large as possible, without producing unacceptably high motor vibrations. However, many systems will require a low **CTG** value, to ensure that the derivative sampling period is shorter than one tenth of the system mechanical time constant. Therefore, start with a low **CTG** and gradually increase it, rather than immediately trying a large **CTG** value.

5 ***Increase Proportional Gain (CPG)***
Determine the **CPG** value iteratively. Increase **CPG**, and evaluate the system damping. Repeat until the system is critically damped. You should increase **CPG** to the largest value that does not cause overshoot or ringing. Because proportional gain and derivative gain affect each other, you may need to repeat step 4 and step 5 several times to arrive at optimum values for **CPG** and **CDG**.

6 ***Determine Integral Gain (CIG) and Integral Limit (CIL) Values***
High values for **CIG** will make the system respond quickly, but can cause other problems. In general, you should set **CIG** to the *lowest* value that will correct following errors and static position errors, but not increase overshoot or settling time. In a system without static torque loading, a **CIG** of zero may be appropriate.

CIL limits **CIG**—therefore, before you increase **CIG** to a particular value, you must first increase **CIL** to an equal or higher value.

TUNING PARAMETERS FOR APEX SERIES MOTORS

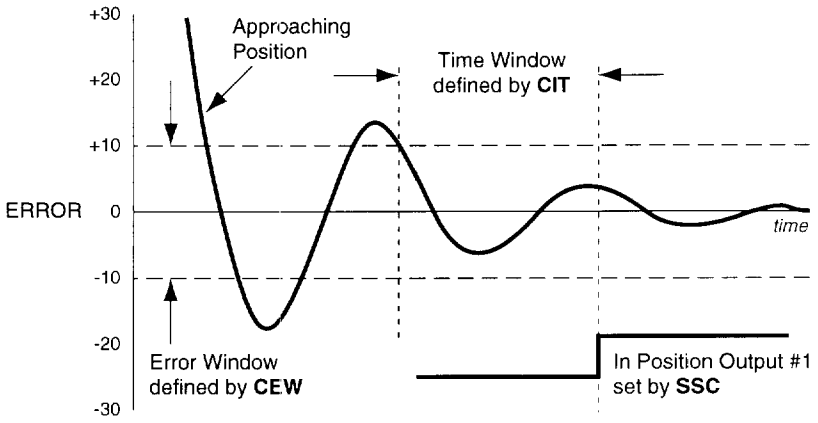
If you use Compumotor's APEX Series Motors, refer to the following table. Set the initial tuning parameters as indicated for your specific motor, then follow the above tuning procedure to fine tune your system.

INITIAL TUNING PARAMETERS FOR APEX MOTORS

MOTOR	CPG	CDG	CTG	CIG	CIL
APEX604	CPG30	CDG220	CTG2	CIG20	CIL40
APEX605	CPG40	CDG250	CTG3	CIG20	CIL40
APEX606	CPG60	CDG300	CTG5	CIG20	CIL40
APEX610	CPG100	CDG320	CTG4	CIG20	CIL40
APEX620	CPG110	CDG350	CTG3	CIG20	CIL40
APEX630	CPG120	CDG380	CTG4	CIG20	CIL40
APEX635	CPG150	CDG390	CTG5	CIG20	CIL40
APEX640	CPG150	CDG390	CTG5	CIG20	CIL40

Configuring an In Position Window

You can define an In Position Window, and use it to indicate that the preceding move is done. Two commands—**CEW** and **CIT**—determine the height and width of the window. A third command—**SSC**—can turn on output #1 when the In Positon criteria are met.



As the drawing shows, **CEW** defines the position error window at the end of a move. **CIT** specifies the length of time the motor must be within the error window. The motor is In Position when three conditions are satisfied:

① The controller algorithm is finished (no input position command)

② Position error is less than that specified by the **CEW** command

③ Condition ② above has been true for the length of time specified by the **CIT** command

If **SSC** has been set to 1, output #1 will turn on when these three conditions have been met. You can use output #1 to trigger external hardware from the In Positon condition. The output will stay on until the next move command is issued, such as **GO** or **GO HOME**.

(**NOTE**: If the motor is held (mechanically, or against an end stop), and **CPE** is greater than **CEW**, the motor may become "trapped" between **CPE** and **CEW**: it will not execute the next

move. In this rare situation, two things are happening: 1.) **CPE** is not violated, and therefore no position error fault occurs; 2.) in position criteria are not met.

If you were to execute a **1R**, the response would be ***B**, which means the drive is "busy" waiting for the move to be over. Why doesn't the drive force the motor to finish the move? The motor is somehow held. To correct this situation, try touching the motor; this may complete the move, and the drive may execute the next move. Or, execute a **DPA** to read actual position, and verify that the move is not complete. You can also execute a **KILL** to reset the positions, and then do the next series of moves.)