Gemini Major Programmer's Guide

This portion of the manual provides guidelines for implementing GT6 and GV6 firmware features.

Before attempting to implement firmware features, make sure you have installed the Gemini drive according to the instructions provided in the drive's *Hardware Installation Guide*.

In this section:

Programming Fundamentals

Creating Programs

A *program* is a series of commands. These commands are executed in the order in which they are programmed. Immediate commands (commands that begin with an exclamation point [!]) cannot be stored in a program. Only buffered commands may be used in a program. As shown in the scenario below, the program is defined with the `DEF PROG` command, followed by the contents of the program, and completed by the `END` command. Up to 32 programs may be defined and stored in the Gemini drive.

A *subroutine* is defined the same as a program, but it is executed with an unconditional branch command, such as `GOSUB` or `JUMP`, from another program. Subroutines can be nested up to 16 levels deep. NOTE: The Gemini products do not support recursive calling of subroutines.

Compiled profiles are defined similar to programs, using the `DEF PROF` and `END` commands (the profile is automatically compiled when the Gemini drive executes the `END` command), and executed with the `PRUN PROF` command. Up to 16 compiled profiles may be defined and stored in the Gemini drive. Compiled profiles also affect a different part of the product's memory, called *compiled memory*. For information on compiled profiles, refer to page 49.

Programming Scenario

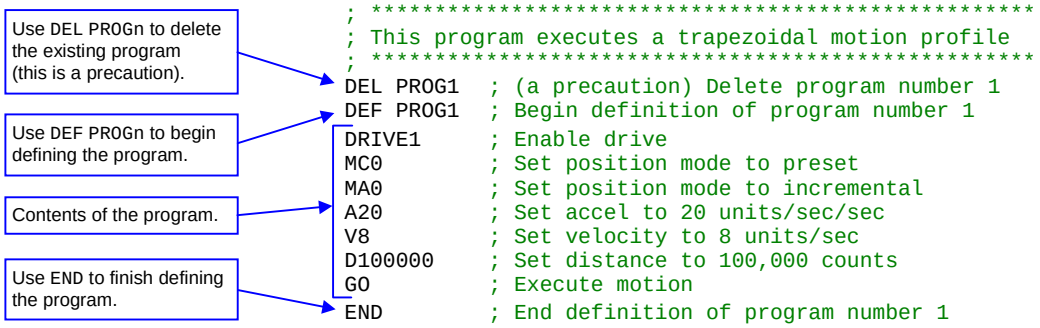
To best understand the process of developing Gemini programs, we invite you to follow along on a program development scenario. The scenario covers these programming tasks:

1. Create a simple motion program in the Motion Planner Editor.
2. Save the program file.
3. Download the program to the Gemini product.
4. Verify that the program is stored in the Gemini's memory (`TDIR`, `TMEM`)
5. Execute (run) the program from the Terminal.

FIRST: If you have not already done so, install Motion Planner (see page 5) and launch the Motion Planner application. Also, establish a serial connection between the Gemini drive and your computer and power up the Gemini drive.

1. Create a program.

Using Motion Planner's Editor, type in the commands as shown in the sample below. The sample also shows program comments to help you understand the purpose of each command and the implications of executing motion. Notice that the comments are placed after the comment delimiter (`;`).




```
; ************************************************
; This program executes a trapezoidal motion profile
; ************************************************
DEL PROG1   ; (a precaution) Delete program number 1
DEF PROG1   ; Begin definition of program number 1
DRIVE1      ; Enable drive
MC0         ; Set position mode to preset
MA0         ; Set position mode to incremental
A20         ; Set accel to 20 units/sec/sec
V8          ; Set velocity to 8 units/sec
D100000     ; Set distance to 100,000 counts
GO          ; Execute motion
END         ; End definition of program number 1
```

2. Save the program.

CAUTION: Programs and profiles stored in the Gemini drive's EEPROM can be deleted with the `DEL PROG` and `DEL PROF` commands, and by executing the `RFS` command. Therefore, to safeguard the contents of your programs, you should save your program files to your hard drive.

To save the program file, use the **File > Save** command or click the  button. Call it "example.prg".

3. Download the program.

To download the program to the Gemini drive, click the  button in the Editor window. When you are presented with a dialog asking if you wish to reset the drive after the download, click "No" (this sample program does not require a reset).

4. Verify the download.

Switch to Motion Planner's Terminal window, type `TDIR` and press ENTER. The drive should respond with "`*PROG1`". This verifies that the program you just downloaded actually resides in the Gemini drive's EEPROM.

TIP: To check the amount of memory (bytes) available for storing additional programs and profiles, use the `TMEM` command. To check the contents of a specific program, use the `TPROG PROGn` command ("n" is the program number); `TPROG` cannot be used to check the contents of defined profiles.

5. Run the program.

WARNING

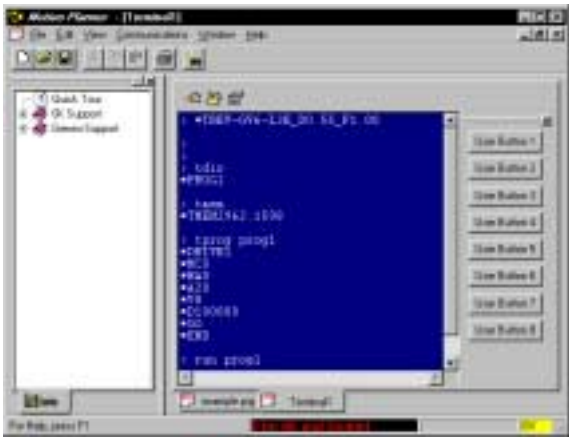
Executing the sample program will cause motion. Make sure it is safe to move the load without damaging equipment or injuring personnel.

In the Terminal, type `RUN PROG1` or `PROG1` to run the program. The motor should make one 100,000-count move in the positive direction.

Congratulations! You've just created and executed your first Gemini motion program. The illustrations below show the contents of Motion Planner's Editor window and Terminal window after completion of the scenario.

Motion Planner (viewing the Editor window)

Motion Planner (viewing the Terminal window)



Using a Setup Program

The intent of the Setup program is to place the Gemini drive in a ready state for subsequent motion control. The setup program typically contains elements such as drive, motor and feedback device configuration, tuning gain selections, programmable I/O definitions, homing configuration, etc.

The basic process of creating a setup program is:

1. Create a program to be used as the setup program. The easiest way to create the setup program is to use the setup wizard. If you are using Motion Planner, refer to page 6. If you are using Pocket Motion Planner, refer to page 11.
2. Save the program and download it to the Gemini drive.
3. There are two main options for executing the setup program:
   a. Execute the `STARTP` command to assign your new program as the "start-up" program (e.g., `STARTP PROG1` assigns program #1 as the start-up program). Thereafter, the assigned start-up program is automatically executed when the Gemini drive is powered up, when the `RESET` command is executed, or when the hardware Reset input (pin 3 on the DRIVE I/O connector) is activated.
   b. Call the setup program from the main program for your application. For example, if the setup program is program #1, at the appropriate location in the main program you could put a `JUMP PROG1` command or a `GOSUB PROG1` command to branch to program #1.

Executing Programs (options)

Following is a list of the primary options for executing programs stored in your controller:

Method	Description
Execute from a terminal emulator	In the Motion Planner terminal or the Pocket Motion Planner terminal, type in the `RUN PROGn` command or the `PROGn` command to execute program *#n*. This is demonstrated in the programming scenario beginning on page 20.
Execute as a subroutine from a "main" program	Use a branch (`GOSUB`, `RUN PROG` or `JUMP`) from the main program to execute another stored program.
Execute automatically when the controller is powered up	Assign a specific program as a startup program with the `STARTP PROGn` command. Thereafter, the assigned start-up program is automatically executed when the Gemini drive is powered up, when the `RESET` command is executed, or when the hardware Reset input (pin 3 on the DRIVE I/O connector) is activated.
Execute a specific program with BCD weighted inputs	Define programmable inputs to function as BCD select inputs, each with a BCD weight. A specific program (identified by its number) is executed based on the combination of active BCD inputs. Related commands: `INSELP` and `INFNCi-B`.
Execute from your own custom Windows program	Use a programming language (e.g., Visual Basic, Visual C++, etc.) and the Gemini Communications Server (provided on the Motion Planner CD) to create your own windows application to control the Gemini product. Refer to page 195 for details on the Communications Server.

Controlling the Execution of Programs and the Command Buffer

The Gemini drive allows you to determine the drive's handling of motion and program/command execution during normal operating events.

`COMEXC` (page 62) controls processing of motion commands received while motion is in progress.

`COMEXL` (page 63) controls whether the command buffer is saved upon encountering a hardware or software end-of-travel limit.

`COMEXR` (page 63) controls how the activation of a pause/continue input (`INFNCi-E`) impacts motion and program execution.

`COMEXS` (page 64) controls how the execution of a stop command (`S`) or the activation of a stop input (`INFNCi-D`) impacts motion and program execution and the command buffer.

Program Flow Control

Program flow refers to the order in which commands will be executed, and when or whether they will be executed at all. In general, commands are executed in the order in which they are received. However, certain commands can redirect the order in which commands will be processed.

You can affect program flow with:

Control Method	Related Commands
Unconditional Loop. Repeats the execution of a group of commands (located between the `L` and `LN` commands) for a predetermined number of iterations.	`L`.....pg. 125 `LN`.....pg. 128
Unconditional Branch. Flow of program execution ("control") passes to the program specified in the branch command. Using `GOSUB`, `RUN PROG`, or `PROG`, processing returns to the original ("calling") program. Using `JUMP`, processing does not return to the calling program.	`GOSUB` .....pg. 112 `RUN PROG` or `PROG` ....pg. 146 `JUMP` .....pg. 123
Wait Statement. Pauses program execution until the specified condition evaluates true.	`WAIT` .....pg. 184
"If" Statement. Executes a specific set of commands (between `IF` and `NIF`) only if a certain condition exists.	`IF`.....pg. 116 `NIF`.....pg. 132 `ELSE` .....pg. 100
Time Delay (Dwell). Impose a time delay between moves or other programmed events.	`T`.....pg. 157
Pause. Activate a "pause" input (an input configured with the `INFNCi-E` command) to pause program execution (and motion if in the `COMEXR1` mode). To resume, deactivate the input or send a `!C` command to the Gemini. Another way to pause program execution (not motion) is to place a `PS` command in the program – when the program is run, it will pause at the `PS` command. To resume, send a `!C` command.	`INFNC` .....pg. 118 `COMEXR` .....pg. 63 `C`.....pg. 61 `PS`.....pg. 140
Stop. Use a `S1` or `!S1` command to stop motion only. Use a `S` or `!S` command or use a "stop" input (an input configured with the `INFNCi-D` command) to stop motion and program execution. `COMEXS` controls whether motion and program can be resumed with `!C` or a resume input (`INFNCi-E`).	`S`.....pg. 147 `INFNC` .....pg. 118 `COMEXS` .....pg. 64 `C`.....pg. 61
Kill. Use a `K1` or `!K1` command to kill motion only Use a `!K` or `K` command, or a "kill" input (an input configured with the `INFNCi-C` command) to kill motion and terminate program execution.	`K`.....pg. 124 `INFNC` .....pg. 118

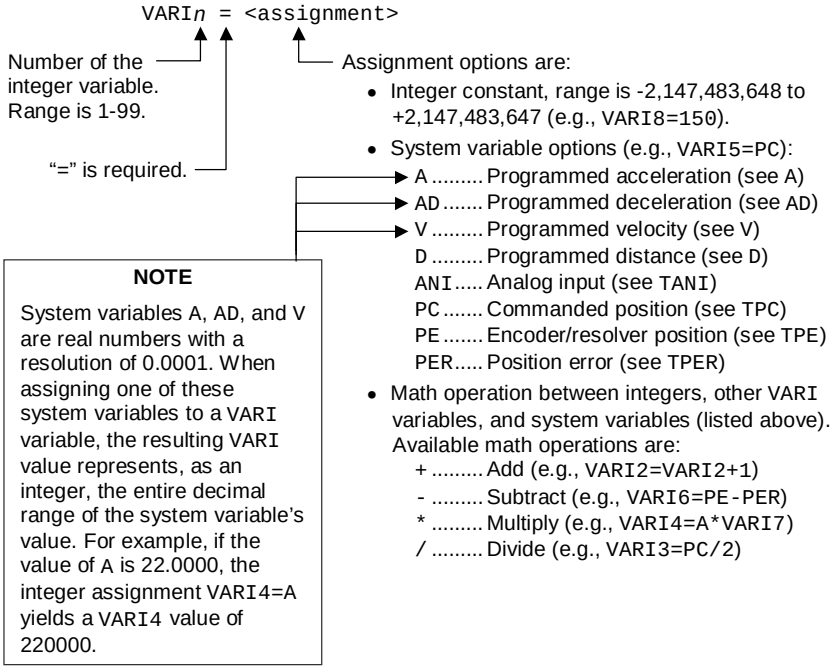
Using Variables

With the release of OS version 1.60, the GT6 and GV6 drives now allow you to define up to 99 user variables (integer variables). Integer variables are represented by the syntax VARI*n*, where "*n*" is the number of the variable (range is 1-99). Integer variables may be used for:

- Variable assignments and math operations
- Command value substitutions
- Variable comparisons in conditional expressions

Variable Assignments and Math Operations

Variable assignment allows you to specify the value of integer variables in terms of integer constants, system variables, or mathematical operations between integers, other VARI variables, and system variables.



To check the present value of an integer variable, execute the VARI*n* command with no other arguments. The drive responds with the present integer value.

NOTE

System variables A, AD, and V are real numbers with a resolution of 0.0001. When assigning one of these system variables to a VARI variable, the resulting VARI value represents, as an integer, the entire decimal range of the system variable's value. For example, if the value of A is 22.0000, the integer assignment VARI4=A yields a VARI4 value of 220000.

Command Value Substitutions

Variable substitution allows you to substitute the value of a VARI variable as the parameter value for certain commands. For example, if VARI9 is substituted for the L loop parameter L(VARI9), and the value of VARI9 is 7 when the L command is executed, the Gemini will initiate a 7-iteration loop.

The commands that allow parameter substitutions are:

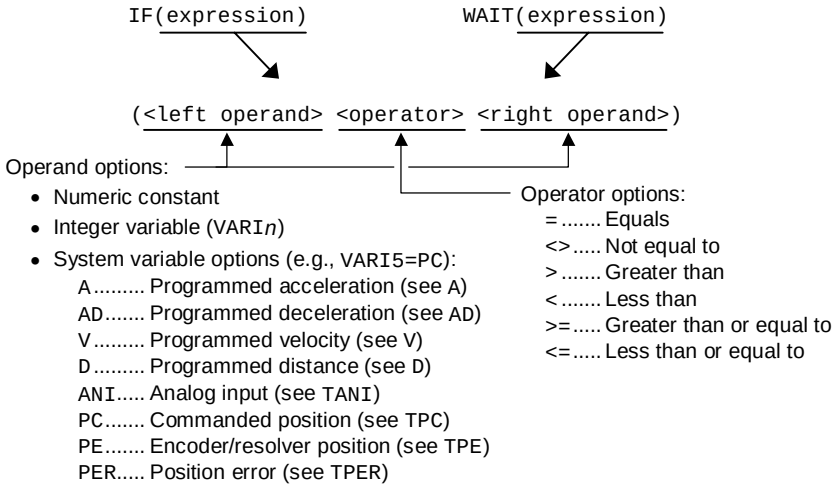
T ..... Time delay (real number, resolution is 0.001)
L ..... Loop
D ..... Distance
A ..... Acceleration (real number, resolution is 0.0001)
AD ..... Deceleration (real number, resolution is 0.0001)
V ..... Velocity (real number, resolution is 0.0001)
REG ..... Registration distance
PSET ..... Establish absolute position

NOTE

The command parameter values for A, AD, V, and T are real numbers. The resolution of A, AD, and V is 0.0001, for T it is 0.001. When substituting VARI variables in these commands, the VARI integer value is applied to the full decimal range. For example, if the value of VARI5 is 136298, the substitution A(VARI5) yields an acceleration value of A13.6298.

Variable Comparisons in Conditional Expressions

Variable comparison allows program flow to be affected by IF and WAIT conditional expressions, based on comparisons between integer variables and system variables or integer values. Relative to variable comparisons, the options for the comparison operands (left and right) and the comparison operator are:



Examples:

IF(VARI5<PE) .....Compares the value of integer variable #5 (VARI5) with the present encoder position (PE). The condition evaluates true when the value of VAR5 is less than the integer value of the encoder position.

WAIT(PE>=16000) .....Command execution pauses until the encoder or resolver position is greater than or equal to 16000.

For details on IF and WAIT conditional statements, refer to the IF command (page 116) and the WAIT command (page 184).

Programming Scenario

This program helps a manufacturer cut web stock a specific length in batches of ten. The Gemini drive controls the web feed motion and it uses one of its outputs to control the saw carriage to cut the stock.

```
DEL PROG1         ; Delete the existing PROG1 program (a precaution)
DEF PROG1         ; Begin definition of setup program
VARI1=10          ; Initialize a batch count variable (VARI1) to 10
VARI2=0           ; Initialize a loop count variable (VARI2) to zero
L                 ; Begin loop (infinite)
 IF(VARI1=VARI2)  ; If the batch is complete, stop the program
  OUTxxxxx1       ; Activate the "batch complete" output
  S               ; Stop
 ELSE             ; If the batch is not complete, cut more stock
 MA0              ; Select incremental positioning mode
 A10              ; Set acceleration and deceleration
 V5               ; Set velocity
 D249888          ; Set distance to increment the stock
 GO1              ; Initiate motion
 WAIT(AS.24=b1)   ; Wait until move complete (within "target zone")
 OUT1             ; Activate output #1 to engage the saw assembly
                  ; to make the cut
 WAIT(IN.5=b1)    ; Wait until the saw assembly returns to its
                  ; ready position (it turns on its "ready" output)
 VARI2=VARI2+1    ; Increment the loop counter
 NIF              ; End of IF code
LN                ; End of loop
END
```

Error Handling

The Gemini drive offers error-handling features that enable you to provide programmed responses to certain error conditions that may occur during the operation of your system. Programmed responses typically include actions such as shutting down the drive, activating or de-activating outputs, etc.

These are the primary elements of implementing error handling:

- **Define an "error program"**. This is a program that you wish to be invoked when an error condition exists. The program is defined like any other program (with `DEF PROGn` and `END`), and it is designated as the "error program" with the `ERRORP` command. For example, the `ERRORP PROG9` command assigns program #9 (an existing program stored in memory) as the error program. An example error program is provided below.
- **Tell the Gemini drive to check for certain error conditions**. Use the `ERROR` command to enable specific error-checking bits. For example, the `ERRORx1x1` command tells the Gemini drive to start checking After an error-checking bit is enabled, the Gemini drive will branch to the assigned error program when the condition occurs. To understand the type of branch and how to recover from the error condition, refer to the `ERRORP` command description (page 107).

  Error Conditions (see `ERROR` and `ERRORP` for details):

  Error Bit #1 ..... Stall detected (GT6 only).
  Error Bit #2 ..... Hardware end-of-travel limit hit.
  Error Bit #3 ..... Software end-of-travel limit hit.
  Error Bit #4 ..... Drive Fault detected.
  Error Bit #5 ..... Commanded stop or kill (`S`, `!S`, `K`, or `!K`).
  Error Bit #6 ..... Kill input (`INFNCi-C`) activated.
  Error Bit #7 ..... User fault input (`INFNCi-F`) activated.
  Error Bit #8 ..... Stop input (`INFNCi-D`) activated.
  Error Bit #9 ..... Enable input not grounded.
  Error Bit #10 ..... OTF or registration move not possible.
  Error Bit #11 ..... Target zone timeout (GV6 only).
  Error Bit #12 ..... Exceeded maximum allowable position error (GV6 only).

NOTE: Regardless of the error checking, the errors are still reported with the `TER` command, and can be used in a conditional statement. For example, `IF(ER.4=b1)` evaluates true if a drive fault (reported with `TER` bit #4) occurs.

Sample setup code and program:

```
DEF PROG8      ; Define program #8
               ; (which will be the error program)
IF(ER=bXX1)    ; If the error is caused by the occurrence of
               ; a software end-of-travel limit (bit #3 set
               ; to one), back off the software limit, reset
               ; the position, & continue.
D~             ; Change direction in preparation to back off
               ; the soft limit
D1             ; Set distance to 1 count (just far enough to
               ; back off the soft limit)
GO1            ; Initiate the 1-count move to back off the
               ; soft limit
PSET0          ; Reset the position to zero
NIF            ; End IF statement
END            ; End definition of program #8

ERRORP PROG8   ; Set program #8 to be the error program.
               ; Branch to program #8 upon encountering a
               ; software end-of-travel limit.

ERRORXX1       ; Set error condition bit #3 to check for
               ; a software end-of-travel limit
```

Basic Operation Setup

Use the Setup Wizard

Using a Setup Program:

The resulting code from the setup wizard, plus other application-specific commands as deemed necessary, is typically placed in a "setup program". The purpose of the setup program is to place the Gemini drive in a ready state for subsequent motion control. The setup program is usually executed on power up or reset (assigned as the start-up program – see STARTP on page 155), or it is executed from the primary or "main" program which controls the overall application.

The process for creating a setup program is presented on page 22.

For many setup parameters (see list below), you should use the setup wizard in Motion Planner (see page 6) or the configuration wizard in Pocket Motion Planner (see page 11). In each wizard, you have the option of performing an "express" setup (just enough to get up and running) or a "full" detailed setup.

If you need help understanding the items you are configuring, refer to the relevant command description.

Motor Setup:

Selection:

If you are using a Parker motor, select the motor series/frame size/part number from the pull-down lists. The wizard automatically fills in all of the motor parameters and many other drive/system parameters (denoted by **), based on the selected Parker motor.

If you are not using a Parker motor, or you are using a custom motor that is not listed, you will have to fill in all motor parameters. **CAUTION**: It is mandatory that all of these parameters be configured to avoid a motor configuration error, which prevents you from enabling the drive.

Parameters: (not necessary if using Parker motor)

Parameter	GT	GV	GT6	GV6	Command
** Rated Speed	–	X	–	X	DMTW
** Number of Pole Pairs	X	X	X	X	DPOLE
** Electrical Pitch (linear motors)	–	X	–	X	DMEPIT
** Rotor Inertia; Forcer Mass	X	X	X	X	DMTJ
** Damping	–	X	–	X	DMTD
** Thermal Time Constant	–	X	–	X	DMTTCM
** Winding Thermal Resistance	–	X	–	X	DMTRWC
Motor Ambient Temperature	–	X	–	X	DMTAMB
** Max. Motor Winding Temp.	–	X	–	X	DMTMAX
** Voltage Constant (Ke)	–	X	–	X	DMTKE
** Continuous Current	X	X	X	X	DMTIC
** Continuous Current Derating	–	X	–	X	DMTICD
** Peak Current	–	X	–	X	DMTIP
** Winding Time Constant	–	X	–	X	DMTTCW
** Winding Resistance	X	X	X	X	DMTRES
** Minimum Inductance	–	X	–	X	DMTLMN
** Maximum Inductance	–	X	–	X	DMTLMX
** Static Torque	X	–	X	–	DMTSTT
** Inductance	X	–	X	–	DMTIND
** Current Loop Gains	X	–	X	–	DIGNA/B/C/D

Drive, Load and Fault Settings:

Drive:

Drive Control Mode	X	X	X	X	DMODE
** Drive Resolution	X	X	X	–	DRES
Drive PWM Frequency	–	X	–	X	DPWM
** Feedback Source Selection	–	X	–	X	SFB
** Encoder Resolution	–	X	–	X	ERES
** Encoder Output Resolution	–	X	–	X	ORES
** Step & Dir. Output Resolution	X		X	–	ORES
** Linear Motor Electrical Pitch	–	X	–	X	DMEPIT
** Auto Current Standby	X		X	–	DAUTOS
** Torque Limit	–	X	–	X	DMTLIM
** Torque Scaling	–	X	–	X	DMTSCL
** Velocity Limit	X	X	X	X	DMVLIM
** Velocity Scaling	X	X	–	–	DMVSCL

(Continued)

Load:

Parameter	GT	GV	GT6	GV6	Command
Load/Rotor Inertia or Mass Ratio	X	X	X	X	LJRAT
Load Damping	–	X	–	X	LDAMP

Fault:

Fault on Startup Indexer Pulses	X	X	–	–	FLTSTP
Fault on Drive Disable	X	X	X	X	FLTDSB
Fault on Stall Detect	X	–	X	–	ESK
Stall Detect Sensitivity	X	–	X	–	DSTALL
Current Foldback	–	X	–	X	DIFOLD
Disable Drive on Kill	–		X	X	KDRIVE
Max Allowable Position Error	–	X	–	X	SMPER
Max Allowable Velocity Error	–	X	–	X	SMVER

I/O Setup (on DRIVE I/O connector):

Digital Inputs:

Input Function Assignment	–	–	X	X	INFNC
Active Level	X	X	X	X	INLVL
Enable End-of-Travel Limits	X	X	X	X	LH
End-of-Travel Limit Decel	–	–	X	X	LHAD
End-of-Travel Limit S-curve Decel	–	–	X	X	LHADA
Input Debounce	X	X	X	X	INDEB

Digital Outputs:

Output Function Assignment	–	–	X	X	OUTFNC
Active Level	X	X	X	X	OUTLVL

Analog Monitor Outputs:

Variables for Monitor A	X	X	X	X	DMONAV
Scaling for Monitor A	X	X	X	X	DMONAS
Variables for Monitor B	X	X	X	X	DMONBV
Scaling for Monitor B	X	X	X	X	DMONBS

Stepper Motor Matching and Damping:

Motor Matching:

Phase A Offset	X	–	X	–	DPHOFA
Phase B Offset	X	–	X	–	DPHOFB
Phase Balance	X	–	X	–	DPHBAL
Waveform	X	–	X	–	DWAVEF

Electronic Damping:

Active Damping Gain	X	–	X	–	DACTDP
Electronic Viscosity Gain	X	–	X	–	DELVIS
Damping During Acceleration	X	–	X	–	DDAMPA
ABS Damping	X	–	X	–	DABSD

Servo Tuning Gains:

Primary Gains:

** Current Loop Bandwidth	–	X	–	X	DIBW
** Velocity Loop Bandwidth	–	X	–	X	DVBW
** Position Loop Bandwidth	–	X	–	X	DPBW

Advanced Gains:

** Current Loop Bandwidth	–	X	–	X	DIBW
** Velocity Loop Bandwidth	–	X	–	X	DVBW
** Position Loop Bandwidth	–	X	–	X	DPBW
Velocity/Position Bandwidth Ratio	–	X	–	X	SGPSIG
Current Damping Ratio	–	X	–	X	SGIRAT
Velocity Damping Ratio	–	X	–	X	SGVRAT
Position Damping Ratio	–	X	–	X	SGPRAT
Notch Filter A Frequency	–	X	–	X	DNOTAF
Notch Filter A Quality Factor	–	X	–	X	DNOTAQ
Notch Filter A Depth	–	X	–	X	DNOTAD
Notch Filter B Frequency	–	X	–	X	DNOTBF
Notch Filter B Quality Factor	–	X	–	X	DNOTBQ
Notch Filter B Depth	–	X	–	X	DNOTBD
Lead Filter Break Frequency	–	X	–	X	DNOTLD
Lag Filter Break Frequency	–	X	–	X	DNOTLG
Integrator, Enable/Disable	–	X	–	X	SGINTE

Resetting the Gemini Drive

There are different ways to reset the Gemini drive, depending on what method and outcome you desire:

- Cycle power, execute the RESET command, or activate the hardware Reset input (pin 3 on the DRIVE I/O connector).

  What is saved in the Gemini drive's memory?
  - Existing programs (DEF PROG) and profiles (DEF PROF). NOTE: One of the programs can be assigned as the "Startup Program" (see STARTP command), which is automatically executed on power up or reset.
  - Some data-related commands, *but only if they are executed outside of a program*; these commands are denoted with "⊠" in the list on page 187. For those commands that are not automatically saved in EEPROM, they may be executed on power up or reset by placing them in a program (DEF PROG) and assigning the program as the "Startup Program" (see STARTP command).

- Execute the RFS command. This effectively returns the Gemini drive to factory default conditions (with the exception of the ERROK prompt and the TDHRS value). NOTE: All stored programs (DEF PROG) and profiles (DEF PROF) are deleted; therefore, you should make sure you save backup copies of original program/profile files on your hard drive (this is demonstrated in the programming scenario on page 20).

End-of-Travel Limits

The Gemini drive can respond to both hardware and software end-of-travel limits. The purpose of hardware and software end-of-travel limits is to prevent the motor's load from traveling past defined limits. Software and hardware limits are typically positioned in such a way that when the software limit is reached, the motor/load will start to decelerate toward the hardware limit, thus allowing for a much smoother stop at the hardware limit. Software limits can be used regardless of incremental or absolute positioning (MA). When a hardware or software end-of-travel limit is reached, the Gemini drive stops motion using the hardware deceleration rate (set with LHAD & LHADA) or software limit deceleration rate (set with LSAD & LSADA).

Related Commands:

LH.....Hardware end-of-travel limit enable
LHAD .....Hardware end-of-travel limit deceleration
LHADA .....Hardware end-of-travel limit deceleration (s-curve)
INFNC .....Limit input function assignments
INLVL .....Limit switch polarity
LS.....Software end-of-travel limit enable
LSAD .....Software end-of-travel limit deceleration
LSADA .....Software end-of-travel limit deceleration (s-curve)
LSNEG .....Software end-of-travel limit (negative direction)
LSPOS .....Software end-of-travel limit (positive direction)
TIN.....Limit input hardware status (factory default: see bit #1 and #2)
TAS.....Bit #15 indicates if a positive direction hardware limit was encountered.
Bit #16 indicates if a negative direction hardware limit was encountered.
Bit #17 indicates if a positive direction software limit was encountered.
Bit #18 indicates if a negative direction software limit was encountered.
TER.....Bit #2 indicates if a hardware end-of-travel limit was encountered;
Bit #3 indicates if a software end-of-travel limit was encountered
ERROR .....If bit #2 or #3 is enabled, the Gemini drive will branch to the ERRORP program if a hardware or software end-of-travel limit is encountered.

How to set up hardware end-of-travel limits:

1. Connect the end-of-travel limit inputs according to the instructions in your *Hardware Installation Guide*. To help assure safety, connect normally-closed switches and leave the active level at default "active low" setting (set with the `INLVL` command).
2. (Optional) Use the `INFNC` command (page 118) to define the inputs to be used as end-of-travel inputs for the respective axes. **NOTE**: When the Gemini drive is shipped from the factory, inputs #1 and #2 are configured with the `INFNC` command to function as the positive and negative end-of-travel limit inputs, respectively.
3. Set the hard limit deceleration rate (`LHAD` & `LHADA`) to be used when the limit switch is activated. The `LHADA` command allows you to define an s-curve deceleration (see page 53 for details on s-curve profiling).

NOTES ON HARDWARE LIMITS

- The Gemini drive is shipped from the factory with the hardware end-of-travel limits enabled (`LH3`), but not connected. Therefore, motion will not be allowed until you do one of the following:
  - Install limit switches or jumper the end-of-travel limit terminals to ground (*refer to your product's Installation Guide for wiring instructions*).
  - Disable the limits with the `LH0` command (only if the load is not coupled).
  - Reverse the active level of the limits by executing the `INLVL00` command.
- If a hardware limit is encountered while limits are enabled (`LH3`), motion must occur in the opposite direction before a move in the original direction is allowed.

How to set up software end-of-travel limits:

1. Use the `LS3` command to enable the software end-of-travel limits.
2. Define the positive-direction limit with the `LSPOS` command, and define the negative-direction limit with the `LSNEG` command.

   The `LSPOS` and `LSNEG` positions are specified as absolute positions, relative to the absolute zero position. The absolute zero position is established upon power-up and after a successful homing operation, and can be reset using the `PSET` command. Be sure to set the `LSPOS` value greater than the `LSNEG` value.

   For example, `LSPOS+80000` establishes the positive-direction limit at absolute position +80,000 and `LSNEG-60000` establishes the negative-direction limit at absolute position -60,000.

NOTES ON SOFTWARE LIMITS

- If a soft limit is encountered while limits are enabled (`LS3`), motion must occur in the opposite direction before a move in the original direction is allowed.
- To ensure proper motion when using software end-of-travel limits, be sure to set the `LSPOS` value to an absolute value greater than the `LSNEG` value.

Homing

The *homing operation* is a sequence of moves that position an axis using the Home Limit input and/or the Z Channel input of an incremental encoder. The goal of the homing operation is to return the load to a repeatable initial starting location.

Zero Reference After Homing: As soon as the homing operation is successfully completed, the absolute position register is reset to zero, thus establishing a zero reference position.

The homing operation has several potential homing functions you can customize to suit the needs of your application (illustrations of the effects of these commands are presented below):

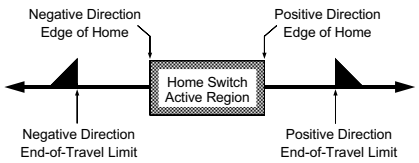
Command	Homing Function	Default
HOM	Initiate the homing move. To start the homing move in the positive direction, use HOMØ; to home in the negative direction, use HOM1.	(do not home)
HOMA	Acceleration and deceleration while homing.	HOMA1Ø (10 revs/sec^2)
HOMBAC	Back up to home. The load will decelerate to a stop after encountering the active edge of the home region, and then will move in the opposite direction at the HOMVF velocity until the active edge of the home region is encountered. Allows the use of HOMEDG and HOMDF.	HOMBAC0 (function disabled)
HOMDF	Final approach direction – during backup to home (HOMBAC) or during homing to the Z channel input of an incremental encoder (HOMZ).	HOMDF0 (positive direction)
HOMEDG	Specify the side of the home switch on which to stop (either the positive-travel side or the negative-travel side).	HOMEGD0 (positive side)
INLVL	Define the home limit input active level (i.e., the state, high or low, which is to be considered an activation of the input). The factory default setting (INLVL110) requires a normally-open switch.	INLVL110 (active-low, use a normally-open switch)
HOMV	Velocity while seeking the home position (see also HOMVF).	HOMV1 (1 rev/sec)
HOMVF	Velocity while in final approach to home position—during backup to home (HOMBAC) or during homing to the Z channel input of an incremental encoder (HOMZ).	HOMVF.1 (0.1 revs/sec)
HOMZ	(GV6 only) Home to the Z channel input from an incremental encoder. NOTE: The home limit input must be active prior to homing to the Z channel.	HOMZ0 (function disabled)

Homing Status:

Status of homing moves is stored in bit #5 of the axis status register (indicates whether or not the home operation was successful). To display the status, use the TAS command. To use the status in a conditional expression (e.g., for an IF statement), use the AS assignment/comparison operator.

NOTES ABOUT HOMING

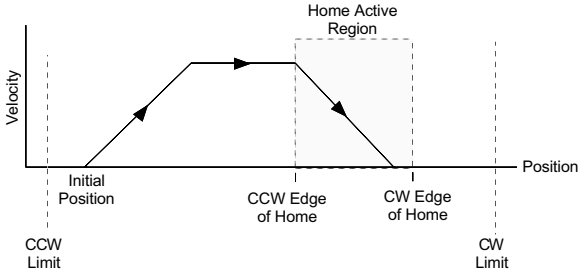
- Avoid using pause and resume functions during the homing operation. A pause command (PS or !PS) or pause/resume input (input configured with the INFNCi-E command) will pause the homing motion. However, when the subsequent resume command (C or !C) or pause/resume input occurs, motion will resume at the beginning of the homing motion sequence.
- "Positive" vs. "negative" direction:



- If an end-of-travel limit is encountered during the homing operation, the motion will be reversed and the home switch will be sought in the opposite direction. If a second limit is encountered, the homing operation will be terminated, stopping motion at the second limit.

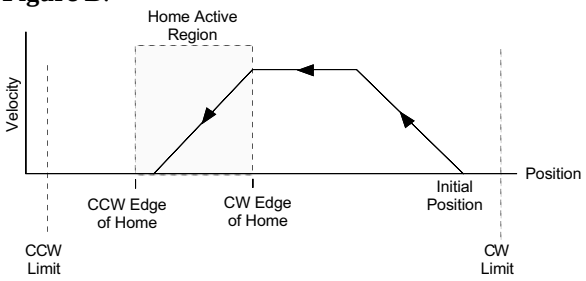
Figures A and B show the homing operation when HOMBAC is not enabled. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

Figure A:



Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home disabled (HOMBAC0)

Figure B:



Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home disabled (HOMBAC0)

Positive Homing, Backup to Home Enabled

The seven steps below describe a sample homing operation when HOMBAC is enabled (see Figure C). The final approach direction (HOMDF) is CW and the home edge (HOMEDG) is the CW edge. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

NOTE

To better illustrate the direction changes in the backup-to-home operation, the illustrations in the remainder of this section show the backup-to-home movements with varied velocities. In reality, the backup-to-home movements are performed at the same velocity (HOMVF value).

Step 1 A CW home move is started with the HOMØ command at the HOMA acceleration. Default HOMA is 10 counts/sec/sec.

Step 2 The HOMV velocity is reached (move continues at that velocity until home input goes active).

Step 3 The CCW edge of the home input is detected, this means the home input is active. At this time the move is decelerated at the HOMA command value. It does not matter if the home input becomes inactive during this deceleration.

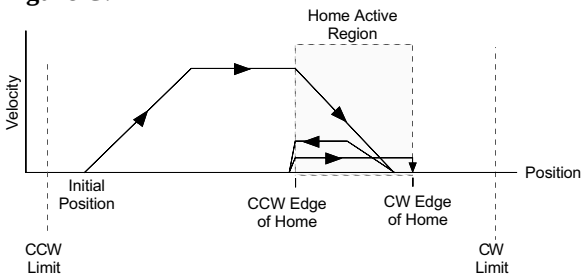
Step 4 After stopping, the direction is reversed and a second move with a peak velocity specified by the HOMVF value is started.

Step 5 This move continues until the CCW edge of the home input is reached.

Step 6 Upon reaching the CCW edge, the move is decelerated at the HOMA command value, the direction is reversed, and another move is started in the CW direction at the HOMVF velocity.

Step 7 As soon as the home input CW edge is reached, this last move is immediately terminated. The load is at home and the absolute position register is reset to zero.

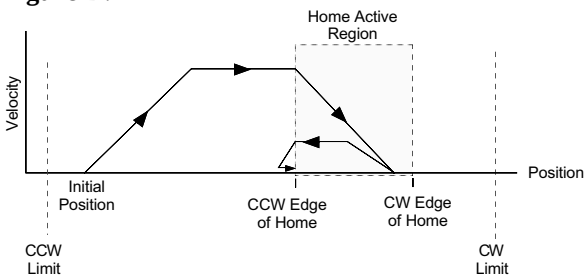
Figure C:



Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the home switch active region (HOMEDG0)

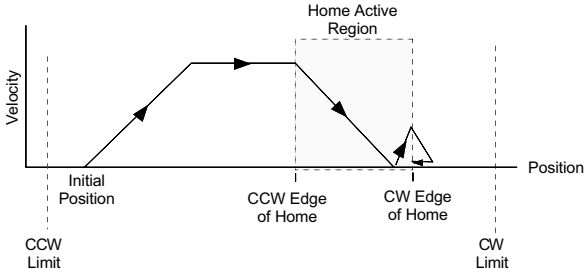
Figures D through F show the homing operation for different values of HOMDF and HOMEDG, when HOMBAC is enabled. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

Figure D:



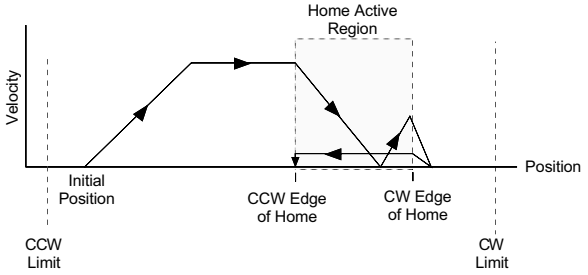
Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the negative-travel side of the home switch active region (HOMEDG1)

Figure E:



Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the positive-travel side of the home switch active region (HOMEDG0)

Figure F:

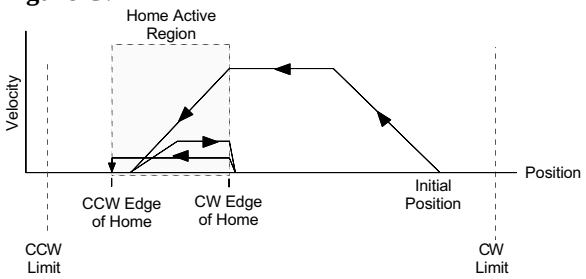


Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the negative-travel side of the home switch active region (HOMEDG1)

Negative Homing, Backup to Home Enabled

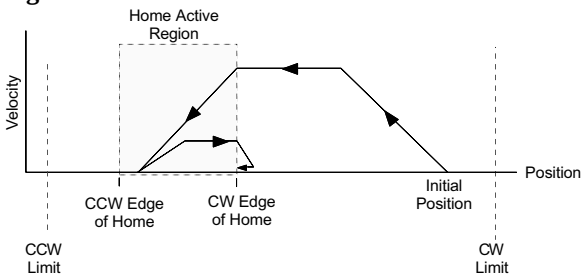
Figures G through J show the homing operation for different values of HOMDF and HOMEDG, when HOMBAC is enabled. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

Figure G:



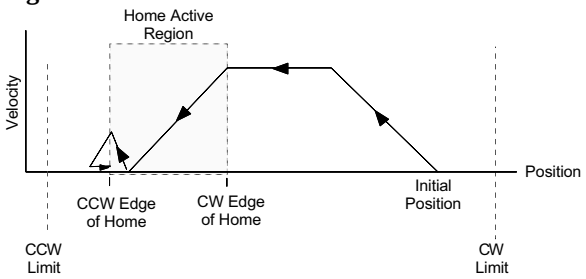
Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the negative-travel side of the home switch active region (HOMEDG1)

Figure H:



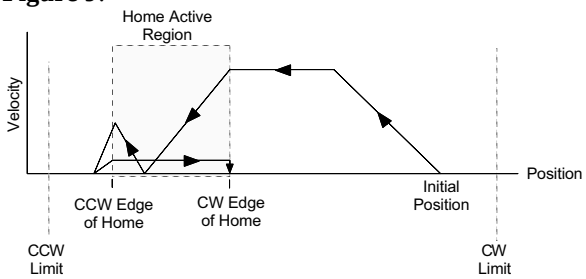
Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the positive-travel side of the home switch active region (HOMEDG0)

Figure I:



Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the negative-travel side of the home switch active region (HOMEDG1)

Figure J:

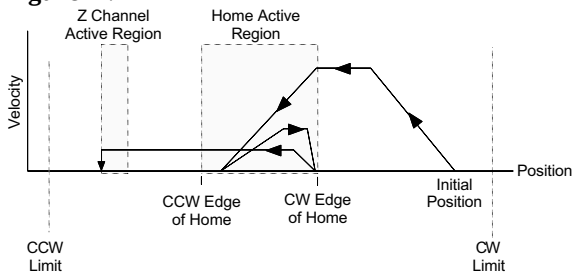


Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the home switch active region (HOMEDG0)

Homing Using The Z-Channel (GV6 only)

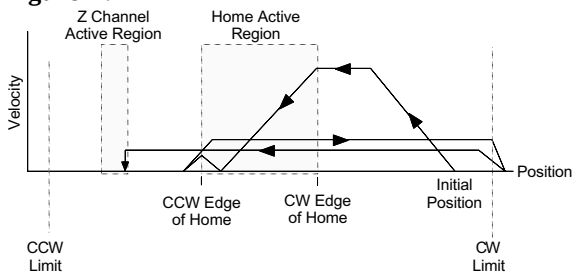
Figures K through O show the homing operation when homing to an encoder index pulse, or Z channel, is enabled (HOMZ1). The Z-channel will only be recognized after the home input is activated. It is desirable to position the Z channel within the home active region; this reduces the time required to search for the Z channel. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

Figure K:



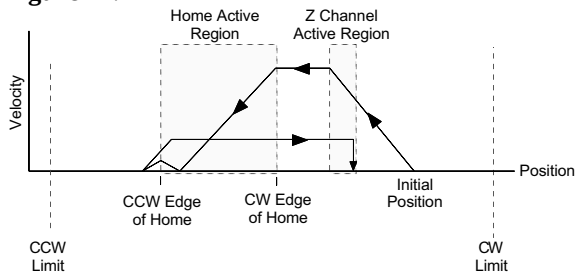
Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the negative-travel side of the z-channel active region (HOMEDG1)

Figure L:



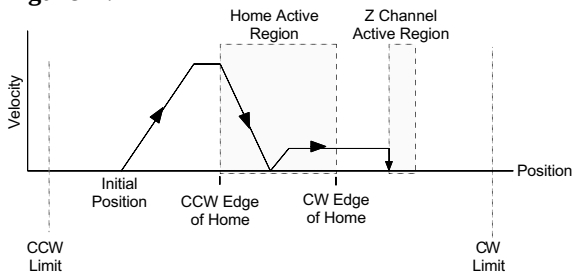
Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the z-channel active region (HOMEDG0)

Figure M:



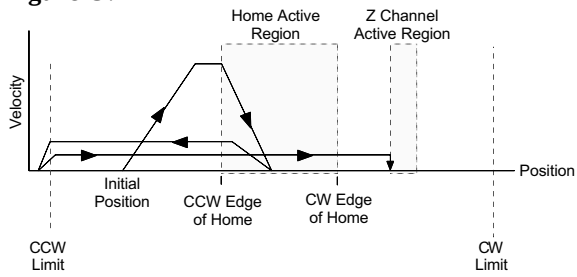
Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the z-channel active region (HOMEDG0)

Figure N:



Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in positive direction (HOM0)
- Backup To Home disabled (HOMBAC0)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the z-channel active region (HOMEDG0)

Figure O:



Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the z-channel active region (HOMEDG0)

Servo Tuning (GV6 only)

The goal of the tuning process is to identify the gain settings (see command list below) required to achieve optimum servo performance in your application. The tuning commands are typically placed into a setup program (see page 22).

Follow the tuning procedures provided in the *GV6 Hardware Installation Guide*. The tuning parameters that you send to the Gemini drive during the tuning procedure will be remembered (stored in the Gemini drive's EEPROM memory); however, for safe-keeping, you should also place these commands in a program file (see page 20) that you can later download to the Gemini drive in the event that the tuning values are changed or the `RFS` command is executed.

Tuning Related Commands

Below is a list of the tuning related commands. All of these commands, collectively, are regarded as a "gain set". You have the option of invoking different gain sets, to accommodate the dynamics of your servo system (see below).

`DIBW` ..... Current loop bandwidth
`DMTLIM`..... Torque/force limit
`DMVLIM`..... Velocity limit
`DNOTAD`..... Notch filter A depth
`DNOTAF`..... Notch filter A frequency
`DNOTAQ`..... Notch filter A quality factor
`DNOTBD`..... Notch filter B depth
`DNOTBF`..... Notch filter B frequency
`DNOTBQ`..... Notch filter A quality factor
`DNOTLD`..... Notch lead filter break frequency
`DNOTLG`..... Notch lag filter break frequency
`DPBW` ..... Position loop bandwidth
`DVBW` ..... Velocity loop bandwidth
`LDAMP` ..... Load damping
`LJRAT` ..... Load-to-rotor inertia ratio or load-to-force mass ratio
`SGAF` ..... Acceleration feedforward gain
`SGINTE`..... Integrator enable
`SGIRAT`..... Current damping ratio
`SGPRAT`..... Position loop ratio
`SGPSIG`..... Velocity/position bandwidth ratio
`SGVF` ..... Velocity feedforward gain
`SGVRAT`..... Velocity damping ratio

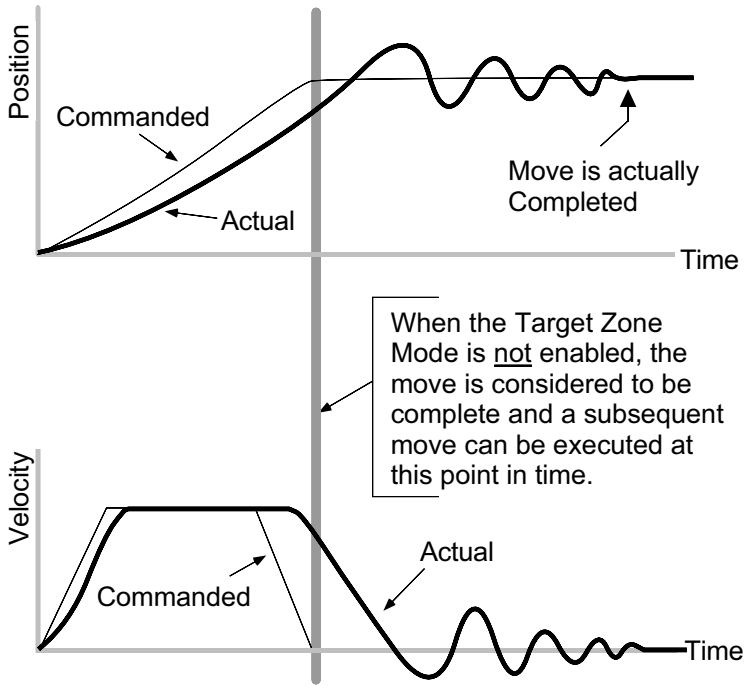
Using Gain Sets

The `SGSET` command saves the presently active gain values (see list above) as a set of gains. Up to three sets of gains can be saved. Any gain set can be displayed using the `TSGSET` command. To report the presently active gain values, enter the `TGAIN` command.

Any gain set can be enabled with the `SGENB` command during motion at any specified point in the profile, or when not in motion. For example, you could use one set of gain parameters for the constant velocity portion of the profile, and when you approach the target position a different set of gains can be enabled.

Target Zone (GV6 only)

Under default operation (Target Zone Mode not enabled), the Gemini drive's move completion criteria is simply derived from the move trajectory. The Gemini considers the current preset move to be complete when the commanded trajectory has reached the desired target position; after that, subsequent commands/ moves can be executed. Consequently, the next move or external operation can begin before the actual position has settled to the commanded position (see diagram below).



To prevent premature command execution before the actual position settles into the commanded position, use the *Target Zone Mode*. In this mode, enabled with the `STRGTE1` command, the move cannot be considered complete until the actual position and actual velocity are within the *target zone* (that is, within the distance zone defined by `STRGTD` and less than or equal to the velocity defined by `STRGTV`). If the load does not settle into the target zone before the timeout period set with the `STRGTT` command, the Gemini detects a *timeout error* (see illustration below).

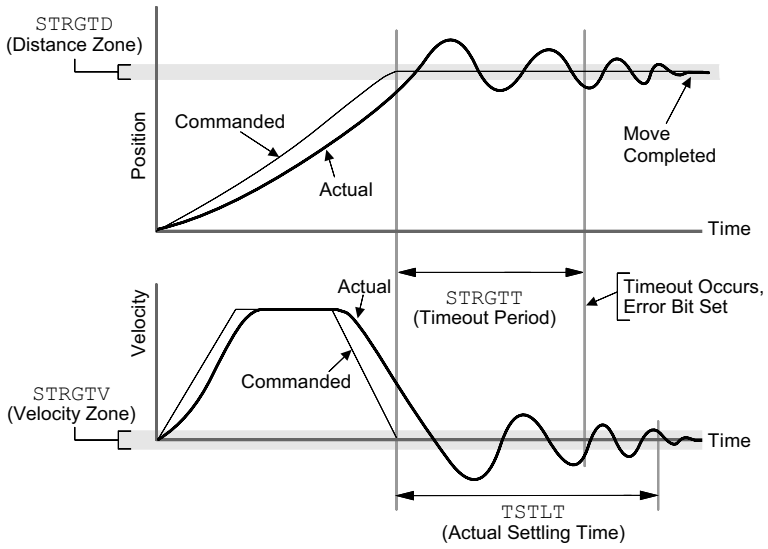
If the timeout error occurs, you can prevent subsequent command/move execution only if you enable the `ERROR` command to continually check for this error condition, and when it occurs to branch to a programmed response you can define in the `ERRORP` program.

As an example, setting the distance zone to ±5 counts (`STRGTD5`), the velocity zone to ≤0.5 revs/sec (`STRGTV0.5`), and the timeout period to 1/2 second (`STRGTT500`), a move with a distance of 8,000 counts (`D8000`) must end up between position 7,995 and 8,005 and settle down to ≤0.5 rps within 500 ms (1/2 second) after the commanded profile is complete.

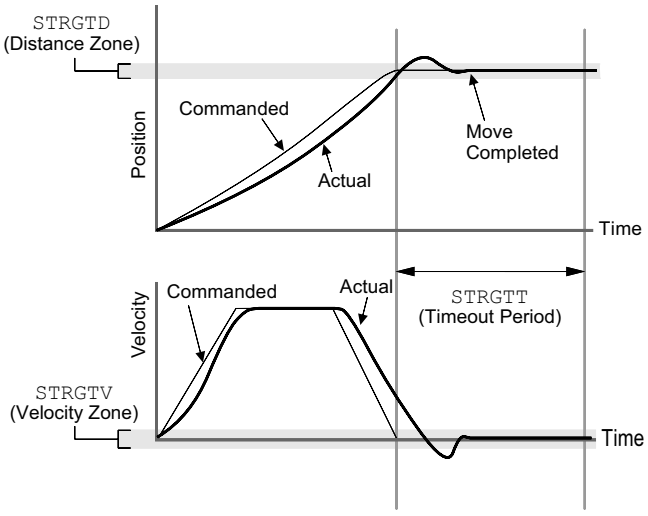
Damping is Critical

To ensure that a move settles within the distance zone, it must be damped to the point that it will not move out of the zone in an oscillatory manner. This helps ensure the actual velocity falls within the target velocity zone set with the STRGTV command (see illustration below).

Failed Move Completion



Successful Move Completion



Checking the Settling Time

Using the TSTLT command, you can display the actual time it took the last move to settle into the target zone (that is, within the distance zone defined by STRGTD and less than or equal to the velocity defined by STRGTV). The reported value represents milliseconds.

The TSTLT command is usable whether or not the Target Zone Settling Mode is enabled with the STRGTE command.

Programmable Inputs and Outputs

Programmable inputs and outputs allow the Gemini drive to detect and respond to the machine processes and program conditions (e.g., state of switches, thumbwheels, electronic sensors, and outputs of other equipment such as PLCs).

What to know about programmable inputs and outputs:

- Each input can be assigned a function that provides a programmed response to external conditions. See INFNC (page 118) for details on each function.

  INFNCi-A .....General-purpose
  INFNCi-B .....BCD program selection
  INFNCi-C .....Kill
  INFNCi-D .....Stop
  INFNCi-E .....Pause/Continue
  INFNCi-F .....User fault
  INFNCi-H .....Trigger interrupt (for registration or TRGFN functions)
  INFNCi-R .....End-of-travel limit, positive direction
  INFNCi-S .....End-of-travel limit, negative direction
  INFNCi-T .....Home limit

- Each output can be assigned a function that provides the ability to affect external conditions. See OUTFNC (page 134) for details on each function.

  OUTFNCi-A .....General-purpose (can be activated with OUT and POUTA)
  OUTFNCi-B .....Moving/not moving indicator
  OUTFNCi-C .....Program in progress indicator
  OUTFNCi-D .....End-of-travel limit encountered indicator
  OUTFNCi-E .....Stall indicator (GT6)
  OUTFNCi-F .....Fault indicator
  OUTFNCi-G .....Position exceeds the max. allowable SMPER limit (GV6)

- The status of each input can be used in a conditional IF statement to control program flow (see IF description on page 116).
- The programmable I/O conditions are updated once per millisecond.

The table below lists each programmable input, its pin number on the DRIVE I/O connector, and its factory default programmed conditions.

Input #	Pin #	Function Assignment (INFNC)	Debounce (INDEB)	Active Level (INLVL)	Status Bit (TIN)
1	28	INFNC1-R (Positive end-of-travel limit)	none	Active high	1
2	29	INFNC2-S (Negative end-of-travel limit)	none	Active high	2
3	31	INFNC3-T (Home limit)	none	Active low	3
4	34	INFNC4-H (Trigger interrupt)	50 ms	Active low	4
5	35	INFNC5-A (General-purpose)	50 ms	Active low	5
6	37	INFNC6-A (General-purpose)	50 ms	Active low	6
7	38	INFNC7-A (General-purpose)	50 ms	Active low	7
8	39	INFNC8-A (General-purpose)	50 ms	Active low	8

The table below lists each programmable output, its pin number on the DRIVE I/O connector (except the Relay), and its factory default programmed conditions.

Output #	Pin #	Function Assignment (OUTFNC)	Active Level (OUTLVL)	Status Bit (TOUT)
1	41	OUTFNC1-A (General-purpose; control with OUT, POUTA)	Active low	1
2	43	OUTFNC2-F (Fault)	Active low	2
3	45	OUTFNC3-D (End-of-travel limit encountered)	Active low	3
4	46	OUTFNC4-E (Stall – GT6) or OUTFNC4-G (Exceeded SMPER limit – GV6)	Active low	4
5	48	OUTFNC5-B (Moving/not moving)	Active low	5
6	49	OUTFNC6-A (General-purpose)	Active low	6
7	Relay	OUTFNC8-F (Fault)	Active low	7

Communication

The Gemini drive has several serial communication setup parameters that you can customize for your application. The list below identifies each parameter and the factory default condition. To explore optional settings, refer to the respective command description later in this manual. Refer to your drive's *Hardware Installation Guide* for RS-232 and RS-485 connection instructions.

<table>
<tr><th>Command</th><th>Description</th><th>Default Setting</th></tr>
<tr><td>E</td><td>Enables/disables serial communication.</td><td>E1 (enabled)</td></tr>
<tr><td>ECHO</td><td>Enables/disables command echo.
If using an RS-232 daisy-chain, use ECHO1.
If using an RS-485 multi-drop, use ECHO0.</td><td>ECHO1 (enabled)</td></tr>
<tr><td>ERRLVL *</td><td>Determines which characters are displayed as part of the response from the Gemini drive.

Characters transmitted when an error is not detected:
<table>
<tr><th></th><th>BOT/EOT/EOL</th><th>*</th><th>ERROK</th></tr>
<tr><td>ERRLVL4:</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>ERRLVL3:</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>ERRLVL2:</td><td>Yes</td><td>Yes</td><td>No</td></tr>
<tr><td>ERRLVL0:</td><td>Yes</td><td>No</td><td>No</td></tr>
</table>
Characters transmitted when an error is detected:
<table>
<tr><th></th><th>BOT/EOT/EOL</th><th>ERRBAD</th><th>Err Msg.</th></tr>
<tr><td>ERRLVL4:</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>ERRLVL3:</td><td>No</td><td>Yes</td><td>No</td></tr>
<tr><td>ERRLVL2:</td><td>No</td><td>No</td><td>No</td></tr>
<tr><td>ERRLVL0:</td><td>No</td><td>No</td><td>No</td></tr>
</table>
Recommendation for RS-485 multi-drop: Use ERRLVL2 to avoid having to address each command to the specific unit in the multi-drop. Be aware, however, that when you use ERRLVL2 there will be no ERROK or ERRBAD prompts and no error messages.</td><td>ERRLVL4</td></tr>
<tr><td>BOT *</td><td>Beginning-of-transmission ASCII characters, placed at the beginning of every response from the Gemini drive. (an ASCII table is provided on page 193)</td><td>BOT0,0,0
(no characters sent)</td></tr>
<tr><td>EOT *</td><td>End-of-transmission ASCII characters, placed at the end of every response from the Gemini drive.</td><td>EOT13,0,0
(carriage return only)</td></tr>
<tr><td>EOL *</td><td>End-of-line ASCII characters, placed at the end of each line of a multi-line response. The last line is terminated with the EOT characters.</td><td>EOL13,10,0
(carriage return and line feed)</td></tr>
<tr><td>ERRBAD *</td><td>ASCII characters sent after an erroneous command has been entered.</td><td>ERRBAD13,10,63,32
(carriage return, line feed, "?", and a space)</td></tr>
<tr><td>ERROK *</td><td>ASCII characters sent after a command has been entered correctly.</td><td>ERROK13,10,62,32
(carriage return, line feed, ">", and a space)</td></tr>
<tr><td>ADDR</td><td>Automatically configures unit addresses for a daisy-chain or multi-drop. Refer to the ADDR description (page 59) or to the RS-232 Daisy Chain or RS-485 Multi-Drop configuration procedures in your drive's *Hardware Installation Guide*.</td><td>ADDR0</td></tr>
<tr><td>XONOFF</td><td>Enables/disables XON/OFF ASCII handshaking with the Gemini drive. If you are using a RS-485 multi-drop, disable XON/XOFF handshaking (XONOFF0).</td><td>XONOFF1 (enabled)</td></tr>
</table>

* These commands are intended to be used only during live terminal communication with the drive. Do not download these commands to the drive, or place them in a program.

Motion Programming

Basic Motion Parameters

Accel, Decel, Velocity, Distance

The basic motion profile comprises acceleration, deceleration, velocity, and distance commands. Motion is initiated with the `GO` command. The table below identifies these commands and their units of measure.

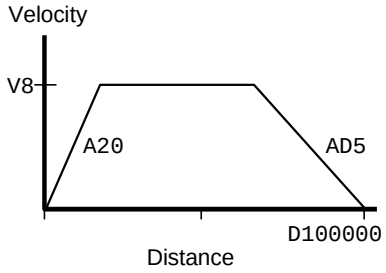
Profile Element	Command	Units *
Acceleration Deceleration	A AD	Revs/sec/sec (rps^2)
Velocity	V	Revs/sec (rps)
Distance	D	Counts

*** NOTE**: If you are using a linear motor, refer to page 44 for instructions on calculating rotary motion parameters to accommodate linear applications.

The following program produces a preset (incremental) 100,000-count move in a nominally trapezoidal profile.

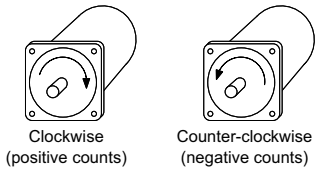
```
DEL PROG6       ; Delete program #6
DEF PROG6       ; Begin definition of program #6
MC0             ; Use the preset positioning mode
MA0             ; Use the incremental (preset) positioning mode
A20             ; Accelerate at 20 revs/sec/sec
AD5             ; Decelerate at 5 revs/sec/sec
V8              ; Set velocity to 8 revs/sec
D100000         ; Set distance to 100,000 counts
GO              ; Execute the move
END             ; End definition of program #6
```

The resulting profile from the above program is:



Direction of Motion for Rotary Motors

Positive distance values (e.g., `D20000`) represent clockwise motion, negative values (e.g., `D-20000`) represent counter-clockwise motion. This assumes you connected the motor (and feedback device for servo drives) according to the *Hardware Installation Guide* instructions.



Positioning Modes

As demonstrated in the program example above, the motion profile is affected by the "positioning mode". There are two main positioning modes:

- Preset Modes (`MC0`): Incremental (`MA0`) or Absolute (`MA1`) – see page 42
- Continuous Mode (`MC1`) – see page 43

Select the mode that is most appropriate for your application. For example, a repetitive cut-to-length application requires incremental positioning. X-Y positioning, on the other hand, is better served in the absolute mode. Continuous mode is useful for applications that require constant movement of the load based on internal conditions or inputs, not distance.

On-The-Fly (Pre-emptive Go) Motion Profiling

While motion is in progress (regardless of the positioning mode), you can change these motion parameters to affect a new profile:

- Acceleration (`A`)
  (s-curve acceleration is not allowed during on-the-fly changes)
- Deceleration (`AD`)
  (s-curve deceleration not allowed during on-the-fly changes)
- Velocity (`V`)
- Distance (`D`)
- Preset or Continuous Positioning Mode Selection (`MC`)
- Incremental or Absolute Positioning Mode Selection (`MA`)

The motion parameters can be changed by sending the respective command (e.g., `A`, `V`, `D`, `MC`, etc.) followed by the `GO` command. If the continuous command execution mode is enabled (`COMEXC1`), you can execute buffered commands; otherwise, you must prefix each command with an immediate command identifier (e.g., `!A`, `!V`, `!D`, `!MC`, etc., followed by `!GO`). The new `GO` command *pre-empts* the motion profile in progress with a new profile based on the new motion parameter(s).

For more information, see *On-The-Fly Motion Profiling* on page 44.

Preset Positioning Modes

A *preset* move is a point-to-point move of a specified distance. You can select preset moves by putting the Gemini drive into preset mode (canceling continuous mode) using the `MC0` command. Preset moves allow you to position the motor/load in relation to the previous stopped position (*incremental mode*—enabled with the `MA0` command) or in relation to a defined zero reference position (*absolute mode*—enabled with the `MA1` command).

Incremental Mode Moves

The incremental mode is the Gemini's default power-up mode. When using the Incremental Mode (`MA0`), a preset move moves the motor/load the specified distance from its starting position. For example, if you start at position *N*, executing the `D6000` command in the `MA0` mode will move the motor/load 6,000 counts in the positive direction from the *N* position. Executing the `D6000` command again will move the motor/load an additional 6,000 positive counts, ending the move 12,000 counts from position *N*.

You can specify the direction of the move by using the optional sign `+` or `-` (e.g., `D+6000` or `D-6000`). Whenever you do not specify the direction (e.g., `D6000`), the direction defaults to positive (`+`).

Example

```
MC0      ; Select Preset Positioning Mode
MA0      ; Select Incremental (Preset) Positioning Mode
A2       ; Set acceleration to 2 revs/sec/sec
V5       ; Set velocity to 5 revs/sec
D4000    ; Set distance to 4,000 positive counts
GO1      ; Initiate motion (move 4,000 positive counts)
GO1      ; Repeat the move (move 4,000 addition counts)
D-8000   ; Set distance to -8,000 counts (return to original position)
GO1      ; Initiate motion (move 8,000 counts in the negative
         ; direction and end at the original starting position)
```

Absolute Mode Moves

A preset move in the Absolute Mode (MA1) moves the motor/load the distance that you specify from the *absolute zero position*.

> **Establishing a *Zero Position***
>
> One way to establish the zero position is to issue the PSETØ command when the load is at the location you would like to reference as absolute position zero.
>
> The zero position is also established when the Go Home (HOM) command is issued, the absolute position register is automatically set to zero after reaching the home position, thus designating the home position as position zero.

The direction of an absolute preset move depends upon the motor's/load's position at the beginning of the move and the position you command it to move to. For example, if the motor/load is at absolute position +12,500, and you instruct it to move to position +5,000 (e.g., with the D5000 command), it will move in the negative direction a distance of 7,500 steps to reach the absolute position of +5,000.

The Gemini retains the absolute position, even while it is in the incremental mode. To ascertain the absolute position, use the TPC command.

Example

```
MC0      ; Select Preset Positioning Mode
MA1      ; Select Absolute (Preset) Positioning Mode
PSET0    ; Set the present absolute position to zero
A5       ; Set acceleration to 5 revs/sec/sec
V3       ; Set velocity to 3 revs/sec
D4000    ; Set move to absolute position +4,000 counts
GO1      ; Initiate motion (move to absolute position +4,000)
D8000    ; Set move to absolute position +8,000 counts
GO1      ; Move (starting from position +4000, move another 4,000
         ; counts in the positive direction to position +8000)
D0       ; Set move to absolute position zero
GO1      ; Move (starting at absolute position +8,000, move 8,000
         ; counts in the negative direction to position zero)
```

Continuous Positioning Mode

The Continuous Mode (MC1) is useful in these situations:

- Applications that require constant movement of the load
- Synchronize the motor to external events such as trigger input signals
- Changing the motion profile after a specified distance or after a specified time period (T command) has elapsed

You can manipulate the motor movement with either buffered or immediate commands. After you issue the GO command, buffered commands are not executed unless the continuous command execution mode is enabled (COMEXC1). Once COMEXC1 is enabled, buffered commands are executed in the order in which they were programmed (see page 62 for details).

The command can be specified as *immediate* by placing an exclamation mark (!) in front of the command. When a command is specified as immediate, it is placed at the front of the command queue and is executed immediately.

Example A

```
DEL PROG22    ; Delete program #22
DEF PROG22    ; Begin definition of program #22
COMEXC1       ; Enable continuous command processing mode
COMEXS1       ; Allow command execution to continue after stop
MC1           ; Select Continuous Positioning Mode
A10           ; Set acceleration to 10 revs/sec/sec
V1            ; Set velocity to 1 rev/sec
GO1           ; Initiates move (Go)
WAIT(AS.4=b1); Wait to reach commanded velocity
T5            ; Time delay of 5 seconds
S1            ; Stop the move
WAIT(AS.1=b0); Wait for no commanded motion
COMEXC0       ; Disable continuous command processing mode
END           ; End definition of program #22
; When the move is executed, the load will accelerate to 1 rev/sec,
; continue at 1 rps for 5 seconds, and then decelerate to a stop.
```

Example B

```
DEL PROG23    ; Delete program #23
DEF PROG23    ; Begin definition of program #23
COMEXC1       ; Enable continuous command processing mode
COMEXS1       ; Allow command execution to continue after stop
MC1           ; Select Continuous Positioning Mode
A10           ; Set acceleration to 10 revs/sec/sec
V1            ; Set velocity to 1 rev/sec
GO1           ; Initiate move (Go)
WAIT(AS.4=b1); Wait to reach commanded velocity
T3            ; Time delay of 3 seconds
A50           ; Set acceleration to 50
V10           ; Set velocity to 10
GO1           ; Initiate on-the-fly accel and velocity changes
T5            ; Time delay of 5 seconds
S1            ; Stop the move
WAIT(AS.1=b0); Wait for no commanded motion
COMEXC0       ; Disable continuous command processing mode
END           ; End definition of program #23
```

While in continuous mode, typical means to stop motion are:

- You issue an immediate Stop (`!S`) or Kill (`!K`) command.
- A hardware or software end-of-travel limit is encountered.
- The load trips a registration (`INFNCi-H`) input.
- The load or operator activates a kill input (`INFNCi-C`), a stop input (`INFNCi-D`), or a user fault input (`INFNCi-F`).

NOTE

While the axis is moving, you cannot change the parameters of some commands (such as `DRIVE` and `HOM` — see command list on page 62). This rule applies during the `COMEXC1` mode and even if you prefix the command with an immediate command identifier (`!`).

Linear Motion

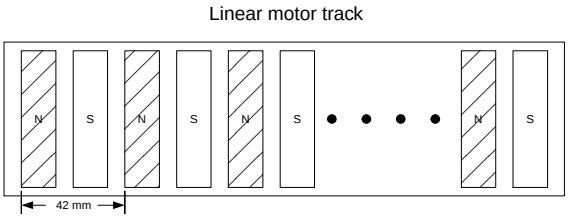
All Gemini drives operate internally in rotary units. If you use a linear motor, you must convert some of the motion parameters to their rotary equivalents:

`A`	Acceleration
`AA`	Acceleration (S-Curve)
`AD`	Deceleration
`ADA`	Deceleration (S-Curve)
`D`	Distance/Position
`HOMA`	Home Acceleration
`HOMV`	Home Velocity
`HOMVF`	Home Final Velocity
`LHAD`	Hardware EOT Limit Decel
`LHADA`	Hardware EOT Limit Decel (S-Curve)
`LSAD`	Software EOT Limit Decel
`LSADA`	Software EOT Limit Decel (S-Curve)
`PSET`	Absolute Position Reference
`REG`	Registration Distance
`REGLOD`	Registration Lockout Distance
`STRGTD`	Target Zone Distance
`STRGTV`	Target Zone Velocity
`V`	Velocity

NOTE: Motor setup is handled differently. If you use the Motion Planner setup wizard (page 6) or the Pocket Motion Planner configuration wizard (page 11), you may enter the motor data in linear units and the software converts them to rotary units before they are downloaded to the drive.

To make the conversion from linear to rotary and vice versa, the motor electrical pitch, or `DMEPIT`, must be known. The electrical pitch relates the linear distance required for the equivalent of one rotary motor revolution. Mechanically, the

definition of the electrical pitch is the linear distance between two magnets comprising a full magnetic cycle. The illustration (below) shows an example of an electrical pitch of 42mm (DMEPIT42). NOTE: Parker linear motors have an electrical pitch of 42mm.



Definition of DMEPIT (Electrical Pitch)

The feedback resolution (ERES) value must be set to the number of counts for the electrical pitch (1 rev).

$$ERES = \underbrace{\frac{\#\,counts}{1\,m}}_{\text{from encoder}} \times \underbrace{\frac{1\,m}{1000\,mm}}_{\text{conversion}} \times \underbrace{\frac{DMEPIT\,(mm)}{1\,(rev)}}_{\text{from motor}} = \frac{\#\,counts}{1\,(rev)}$$

Most linear encoders are metric, and most of those are 1 micron (1 µm):

$$1\ micron\ encoder = \frac{1\ count}{1\ \mu m} = \frac{1{,}000{,}000\ counts}{m}$$

Use the following formulas to convert from linear to rotary units. An sample conversion for position, acceleration, and velocity is provided on page 46.

Linear Position

- D (distance)
- PSET (absolute position reference)
- REG (registration distance)
- REGLOD (registration lockout distance)
- STRGTD (target zone distance)

Metric:

$$Position_{rotary}\,(counts) = \underbrace{\frac{\#\,counts}{1\,(m)}}_{\text{from encoder}} \times Position_{linear}\,(m)$$

English:

$$Position_{rotary}\,(counts) = \underbrace{\frac{\#\,counts}{1\,(m)}}_{\text{from encoder}} \times \frac{.0254\,(m)}{1\,(in)} \times Position_{linear}\,(in)$$

$$Position_{rotary}\,(counts) = \underbrace{\frac{\#\,counts}{1\,(m)}}_{\text{from encoder}} \times \frac{.0254\,(m)}{1\,(in)} \times \frac{12\,(in)}{1\,(ft)} \times Position_{linear}\,(ft)$$

Linear Velocity

- V (velocity)
- HOMV (home velocity)
- HOMVF (home final velocity)
- STRGTV (target velocity)

Metric:

$$Velocity_{rotary}\,(rps) = \frac{1}{DMEPIT\,(mm)} \times \frac{1000\,(mm)}{1\,(m)} \times Velocity_{linear}\,(m/s)$$

English:

$$Velocity_{rotary}\,(rps) = \frac{1}{DMEPIT\,(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times Velocity_{linear}\,(in/s)$$

$$Velocity_{rotary}\,(rps) = \frac{1}{DMEPIT\,(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times \frac{12\,(in)}{1\,(ft)} \times Velocity_{linear}\,(ft/s)$$

Linear Acceleration

- A (acceleration)
- AA (s-curve accel)
- AD (deceleration)
- AD (s-curve decel)
- HOMA (home accel)
- LHAD (hard limit decel)
- LHADA (hard limit s-curve decel)
- LSAD (soft limit decel)
- LSADA (soft limit s-curve decel)

Metric:

$$Accel_{rotary}(rpss) = \frac{1}{DMEPIT(mm)} \times \frac{1000\,(mm)}{1\,(m)} \times Accel_{linear}(m/s^2)$$

English:

$$Accel_{rotary}(rpss) = \frac{1}{DMEPIT(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times Accel_{linear}(in/s^2)$$

$$Accel_{rotary}(rpss) = \frac{1}{DMEPIT(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times \frac{12\,(in)}{1\,(ft)} \times Accel_{linear}(ft/s^2)$$

Conversion Example

Using a Parker 406-LXR-M-D15-E2 linear motor, the electrical pitch is 42mm (DMEPIT42) and the encoder is 1 micron. Here, we will convert an acceleration of 10 inches/sec^2, a velocity of 5 inches/sec, and a distance of 20 inches:

1. Calculate ERES in counts/rev (result is ERES42000):

$$ERES = \underbrace{\frac{1\,000\,000\ counts}{1\,(m)}}_{\text{from encoder}} \times \underbrace{\frac{1\,(m)}{1000\,(mm)}}_{\text{conversion}} \times \underbrace{\frac{42\,(mm)}{1\,(rev)}}_{\text{from motor}} = \frac{42\,000\ counts}{1\,(rev)}$$

2. Convert an acceleration of 10 inches/sec^2 to its rotary equivalent in revs/sec^2:

$$Accel_{rotary}(rpss) = \frac{1}{42\,(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times 10\,(in/s^2) = 6.0476\ rpss$$

3. Convert a velocity of 5 inches/sec to its rotary equivalent in revs/sec:

$$Velocity_{rotary}(rps) = \frac{1}{42\,(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times 5\,(in/s) = 3.0238\ rps$$

4. Convert a distance of 20 inches to its rotary equivalent in counts:

$$Position_{rotary}(counts) = \underbrace{\frac{1\,000\,000\ counts}{1\,(m)}}_{\text{from encoder}} \times \frac{.0254\,(m)}{1\,(in)} \times 20.0\,(in) = 508\,000$$

The program values resulting from these conversions are:

```
ERES42000     ; Set encoder resolution to 42000 counts/rev
A6.0476       ; Set acceleration to 6.0476 revs/sec/sec
              ; (10 inches/sec/sec)
V3.0238       ; Set velocity to 3.0238 revs/sec (5 inches/sec)
D508000       ; Set distance to 508,000 counts (20 inches)
```

On-The-Fly Motion Profiling

While motion is in progress, you can change these motion parameters to affect a new profile:

- Acceleration (A) — s-curve acceleration is not allowed
- Deceleration (AD) — s-curve deceleration is not allowed
- Velocity (V)
- Distance (D)
- Preset or Continuous Positioning Mode Selection (MC)
- Incremental or Absolute Positioning Mode Selection (MA)

The motion parameters can be changed by sending the respective command (e.g., A, V, D, MC) followed by the GO command. If the continuous command execution mode is enabled (COMEXC1), you can execute buffered commands; otherwise (COMEXC0), you must prefix each command with an immediate command identifier (e.g., !A, !V, !D, !MC, followed by !GO).

The new GO command pre-empts the motion profile in progress with a new profile based on the new motion parameter(s). On-the-fly motion changes are applicable only for motion started with the GO command, and not for motion started with HOM or PRUN.

On-the-fly motion changes are most likely to be used to change the velocity and/or goal position of a preset move already underway. In the event that the goal position is completely unknown before motion starts, a move may be started in continuous mode (MC1), with a switch to preset mode (MC0), a distance command (D), and a GO given later. In absolute positioning mode (MA1) the new goal position given with a pre-emptive GO is explicit in the D command. In incremental positioning (MA0) the distance given with a new pre-emptive GO is always measured from the at-rest position before the original GO. If a move is stopped (with the S command), and then resumed (with the C command), this resumed motion is considered to be part of the original GO. A subsequent distance given with a new pre-emptive GO is measured from the at rest position before the original GO, not the intermediate stopped position.

Programming Example: *This program creates a 2-tiered profile that changes velocity and deceleration when specific inputs are activated.*

```
DEL PROG17        ; Delete program #17
DEF PROG17        ; Begin definition of program #17
PSET0             ; Set position to zero
COMEXC1           ; Enable continuous command processing mode
MC0               ; Select preset positioning
MA0               ; Select incremental positioning
A20               ; Set accel to 20 revs/sec/sec
AD20              ; Set decel to 20 revs/sec/sec
V9                ; Set velocity to 9 revs/sec
D500000           ; Set distance to 500,000 counts
GO1               ; Initiate motion
WAIT(IN.5=b1)     ; Wait until input #5 is activated
V4                ; Slow down for machine operation
GO1               ; Initiate new profile with new velocity
WAIT(IN.6=b1)     ; Wait until input #6 is activated
AD5               ; Set decel for gentle stop
V1                ; Slow down for gentle stop
GO1               ; Initiate new profile with new velocity
END               ; End program definition
```

OTF Error Conditions

Further instructions about handling error conditions are provided on page 26 and in the ERROR and ERRORP command descriptions.

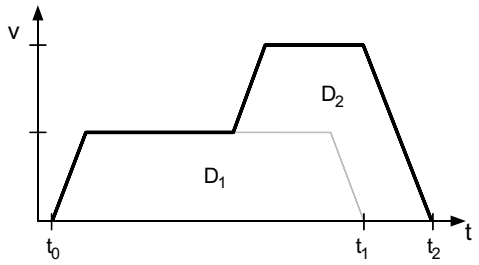
The ability to change the goal position on the fly raises the possibility that the new position goal of an on-the-fly GO cannot be reached with the present direction, velocity, and deceleration. If this happens, an error condition is flagged in axis status (TAS) bit #30 and in error status (TER) bit #10.

If the direction of the new goal position is opposite that of present travel direction, the Gemini will kill motion (decelerating at the LHAD value) and set TAS bit #30 and TER bit #10. Refer to scenario #2 below.

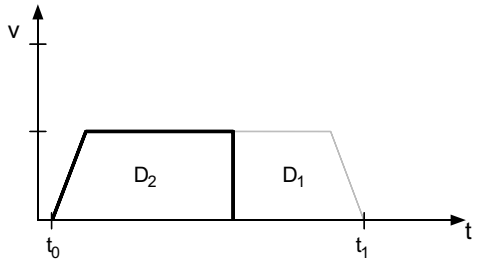
If there has not yet been an overshoot, but it is not possible to decelerate to the new distance from the present velocity using the specified AD value, this case is considered an overshoot — the Gemini with kill the move and set TAS bit #30 and TER bit #10.

Scenarios

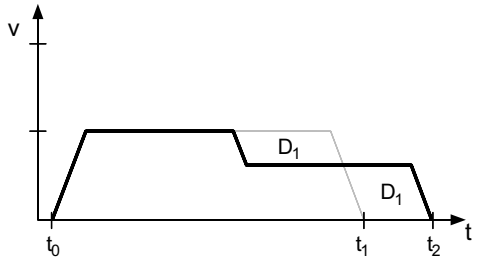
Scenario #1: OTF change of velocity and distance, where new commanded distance (D_2) is greater than the original distance (D_1) that was pre-empted [$D_2 > D_1$]. The distances are the areas under the profiles, starting at t_0 for both. If the original move had continued, D_1 would have been reached at time t_1. D_2 is reached at time t_2.



Scenario #2: OTF change of distance, where new commanded distance (D_2) is less than the original distance (D_1) that was pre-empted [$D_2 < D_1$]. In this example, the position where the OTF change was entered is already beyond D_2 (or D_2 can not be reached with the commanded deceleration). The result is an error and motion is killed (decel at the LHAD value) and TAS bit #30 and TER bit #10 are set.



Scenario #3: OTF change of velocity. Note that motion must continue for a longer time at the reduced velocity to reach the original commanded distance than if it had continued at the original velocity ($t_2 > t_1$).



Registration

The Gemini drive offers a “Registration” feature, which allows you to interrupt motion in progress with a predefined registration profile. The interrupt is triggered by activating a “Registration Input” (an input configured with the INFNCi-H command.

When a *registration input* is activated, the motion profile currently being executed is replaced by the registration profile with its own distance (REG), acceleration (A & AA), deceleration (AD & ADA), and velocity (V) values. The registration move may interrupt any preset (MC0) or continuous (MC1) move in progress. However, a registration move in progress cannot be interrupted by a secondary registration move.

The registration move does not alter the rest of the program being executed when registration occurs, nor does it affect commands being executed in the background if the Gemini is operating in the continuous command execution mode (COMEXC1).

For further details and programming samples, refer to the RE description on page 141.

Compiled Motion Profiling

The Gemini drive allows you to design compiled motion profiles. Because the motion and functions are *pre-compiled*, delays associated with command processing are eliminated during profile execution, allowing more rapid sequencing of actions than would be possible with programs which are not compiled. Command processing is then free to monitor other activities such as I/O and communications.

Compiled profiles are defined similar to programs, using the DEF PROF and END commands (the profile is automatically compiled when the Gemini drive executes the END command), and executed with the PRUN PROF command. The commands that can be used in a compiled profile are:

A.....Acceleration.
AD.....Deceleration.
D.....Distance.
MC.....Continuous or preset (incremental) positioning mode.
V.....Velocity.
VF.....Final velocity of the segment or profile.
GOBUF .....Store a compiled motion segment. A profile is constructed by sequentially appending motion segments using GOBUF commands. Each GOBUF motion segment may have its own distance to travel, velocity, acceleration, and deceleration.
GOWHEN ....Delay execution of the subsequent GOBUF statement until the specified time delay (in milliseconds) has been satisfied. During the time delay, the profile in progress continues at constant velocity. For example, when progress through the profile reaches the GOWHEN(T=500) command, execution of the subsequent GOBUF is paused for ½ second.
TRGFN .....Suspend execution of a subsequent GOBUF until a specified trigger interrupt input is activated. For example, TRGFND1 suspends execution of the GOBUF until you activate input #4.
PLOOP .....Beginning of loop.
PLN.....End of loop.
POUTA .....Turn on/off a “general-purpose” output. If you attempt to change an output that is not defined as a “general-purpose” output (OUTFNCi-A), the POUTA command will be ignored. By factory default, outputs #1 and #6 can be controlled. For example, POUTA.1-1 turns on output #1.
SGENB .....Enable a servo gain set.

GOWHEN & TRGFN Status: ➔
Axis Status (TAS) bit #26 is set when a GOBUF is suspended, pending activation of a TRGFN trigger input, or pending a GOWHEN time delay.

Up to 16 compiled profiles may be defined and stored in the Gemini drive. Compiled profiles are stored in the *compiled* portion of the Gemini's EEPROM memory. To check the amount of memory available, use the `TMEM` command (the right-hand value indicates the number of bytes remaining in compiled memory). To check which profiles are stored in the Gemini, use the `TDIR` command.

Constructing a Compiled Profile

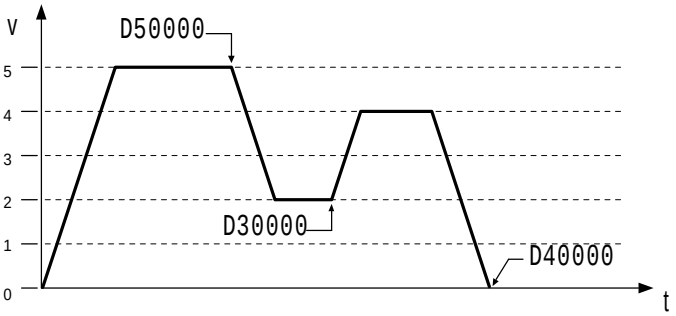
A compiled profile is constructed by sequentially appending motion segments using `GOBUF` commands. Each `GOBUF` motion segment may have its own distance to travel, velocity, acceleration, and deceleration – this is demonstrated in the following program.

The end of a `GOBUF` motion segment in preset mode (`MC0`) is determined by the specified distance (`D`). The end of a `GOBUF` motion segment in continuous mode (`MC1`) is determined by the specified goal velocity (`V` or `VF`). In both cases, the final velocity and position achieved by a segment will be the starting velocity and position for the next segment. If either type of segment is followed by a `GOWHEN` command, the segment's final velocity will be maintained until the `GOWHEN` condition evaluates true.

Example:

```
DEL PROF1 ; Delete profile #1
DEF PROF1 ; Begin definition of profile #1
MC0       ; Use the incremental preset positioning mode
D50000    ; Distance is 50000
A10       ; Acceleration is 10
AD10      ; Deceleration is 10
V5        ; Velocity is 5
GOBUF1    ; Store the first motion segment.
          ; Attributes are: MC0, A10, AD10, V5, D50000
D30000    ; Distance is 30000
V2        ; Velocity is 2
GOBUF1    ; Store the second motion segment.
          ; Attributes are: MC0, A10, AD10, V2, D30000
D40000    ; Distance is 40000
V4        ; Velocity is 4
GOBUF1    ; Store the third motion segment.
          ; Attributes are: MC0, A10, AD10, V4, D40000
          ; Because this is the last segment in a preset
          ; profile, the velocity will automatically end at zero.
END       ; End profile definition
```

The resulting profile from the above program when you execute `PRUN PROF1`:

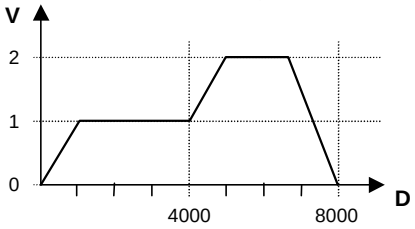


Rules for Using Velocity in Preset Compiled Motion

When defining preset mode (MC0) compiled profiles there are several rules that govern the velocity.

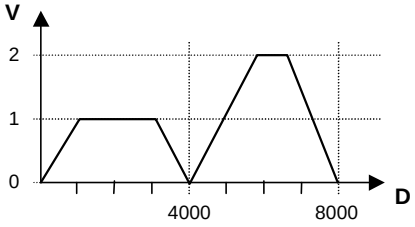
Rule #1: The last segment in the compiled profile will automatically end at zero velocity (only if not in a PLOOP/PLN loop).

```
DEF PROF6   ; Begin definition of profile #6
MC0         ; Select preset positioning
V1          ; Set velocity to 1 rev/sec
D4000       ; Set distance to 4000
GOBUF1      ; First motion segment (V1, D4000)
V2          ; Set velocity to 2 (second segment)
GOBUF1      ; Second motion segment (V2, D4000)
END         ; End definition of profile #6
; When you execute PRUN PROF5, the resulting profile is:
```



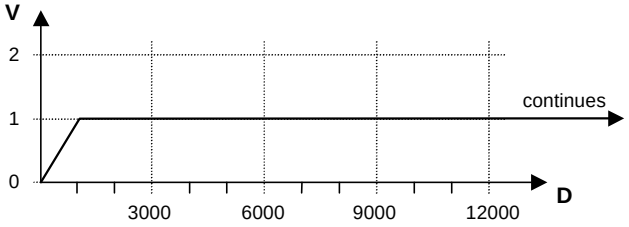
Rule #2: If you wish intermediate segments to end in zero velocity, use the VF0 command in the respective GOBUF segment.

```
DEF PROF5   ; Begin definition of profile #5
MC0         ; Select preset positioning
V1          ; Set velocity to 1 rev/sec
VF0         ; End this segment at zero velocity
D4000       ; Set distance to 4000
GOBUF1      ; First motion segment (V1, D4000, VF0)
V2          ; Set velocity to 2 (second segment)
VF0         ; End this segment at zero velocity
GOBUF1      ; Second motion segment (V2, D4000, VF0)
END         ; End definition of profile #5
; When you execute PRUN PROF6, the resulting profile is:
```



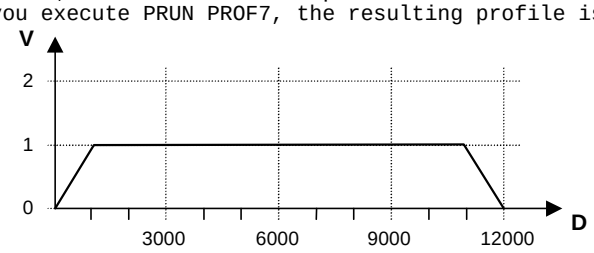
Rule #3: **WARNING**: With compiled loops (PLOOP and PLN), the last segment within the loop must end at zero velocity or there must be a final segment placed outside the loop. Otherwise, after the final segment is completed, the motor will continue moving at the last segment's velocity.

```
DEF PROF7   ; Begin definition of profile #7
MC0         ; Select preset positioning
D3000       ; Set distance to 3000
PLOOP4      ; Loop (between PLOOP & PLN) 4 times
V1          ; Set velocity to 1 rev/sec
GOBUF1      ; First motion segment
PLN1        ; End loop
END         ; End definition of profile #7
; When you execute PRUN PROF7, the resulting profile is:
```



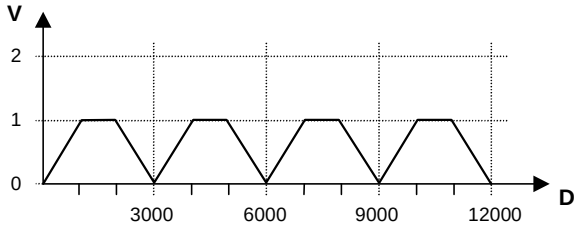
To fix the profile, reduce the PLOOP count by one and add a GOBUF statement after the PLN command:

```
DEF PROF7   ; Begin definition of profile #7
MC0         ; Select preset positioning
D3000       ; Set distance to 3000
PLOOP3      ; Loop (between PLOOP & PLN) 3 times
V1          ; Set velocity to 1 rev/sec
GOBUF1      ; Looped motion segment
PLN1        ; End loop
GOBUF1      ; Last motion segment (end at zero velocity)
END         ; End definition of profile #7
; When you execute PRUN PROF7, the resulting profile is:
```



Rule #4: With compiled loops (PLOOP and PLN), if you wish the velocity at the end of each loop to end at zero, use a VF0 command.

```
DEF PROF8   ; Begin definition of profile #8
MC0         ; Select preset positioning
D3000       ; Set distance to 3000
PLOOP4      ; Loop (between PLOOP & PLN) 4 times
V1          ; Set velocity to 1 rev/sec
VF0         ; End each segment at zero velocity
GOBUF1      ; Looped motion segment
PLN1        ; End loop
END         ; End definition of profile #8
; When you execute PRUN PROF8, the resulting profile is:
```



Dwells and Direction Changes

Compiled profiles may incorporate changes in direction only if the preceding motion segment has come to rest. This may be achieved by creating a continuous segment with a goal velocity of zero, or by preceding a preset segment with VF0. Motion within the profile comes to rest, although the profile is not yet complete. Even though the motor is not moving, axis status (TAS) bit 1 will remain set, indicating a profile is still underway. Only then can you change direction using the D+, D-, or D~ command within a profile. An attempt to incorporate changes in direction if the preceding motion segment has not come to rest will cause a fault.

In many applications, it may be useful to create a time delay between moves. For example, a machine cycle may require a move out, dwell for 2 seconds, and move back. To create this dwell, a compiled GOWHEN may be used between the two moves. The code within a compiled program may look like:

```
MC0             ; Preset positioning used
D26000          ; Target distance is 26000 counts
VF0             ; Motion comes to rest at end of move
GOBUF1          ; Create the "move out" segment
GOWHEN(T=2000)  ; Profile delays for 2 seconds
D-              ; Return to starting position (direction reversed)
VF0             ; Motion comes to rest at end of move
GOBUF1          ; Create the "move back" segment
```

Compiled Motion vs. On-the-Fly Motion

The two basic ways of creating a complex profile are with compiled motion or with on-the-fly pre-emptive GO commands (see page 44). With compiled motion, portions of a profile are built piece by piece, and stored for later execution. Compiled motion is appropriate for profiles with motion segments of pre-determined velocity, acceleration and distance. Compiled motion profiles allow for shorter motion segments, which results in faster cycle times because there is no command processing and execution delay, thus freeing program flow for other tasks, such as I/O, machine control, and host requests. The disadvantages to pre-defined compiled motion profiles are the amount of memory use and limited run-time decision making and I/O processing.

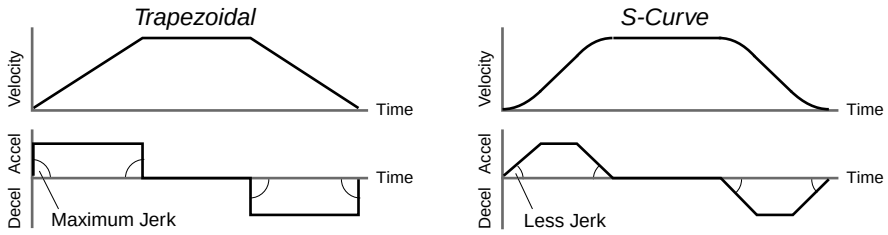
With pre-emptive GO moves, the motion profile underway is pre-empted with a new profile when a new GO command is issued. The new GO command constructs and launches the pre-empting profile. Pre-emptive GOs are appropriate when the desired motion parameters are not known until motion is already underway.

The table below summarizes the differences between the use of compiled motion and on-the-fly motion.

Command/Issue	Compiled Motion	On-The-Fly Motion
GOBUF	Constructs motion segment and appends to previously constructed segment.	N/A
PRUN PROF	Used to launch compiled motion profiles.	N/A
GO	Ignored while executing a compiled profile with PRUN PROF.	Constructs & launches profile, even if moving.
Direction changes	Only if previous motion segment comes to rest; otherwise, a fault may result (GT6: TAS bit 12 and TER bit 1; GV6: TAS bit 23 and TER bit 12).	Not allowed during motion; a fault will result (TAS bit 30 and TER bit 10 are set).
Insufficient room for AD (decel) value	Same as on-the-fly.	Motion is killed, TAS bit 30 and TER bit 10 are set.

S-Curve Accel/Decel Profiling

Gemini drives allow you to perform *S-curve* move profiles, in addition to the usual trapezoidal profiles. S-curve profiling provides smoother motion control by reducing the *jerk* (rate of change) in acceleration and deceleration portions of the move profile (see drawing below). Because S-curve profiling reduces jerk, it improves position tracking performance.



S-Curve Programming Requirements

To program an S-curve profile, you must use the *average accel/decel* commands provided in the Gemini programming language. For every maximum accel/decel command (A, AD, LHAD, and LSAD) there is an *average* command for S-curve profiling (see table below).

Maximum Accel/Decel Commands:		**"S-Curve" Accel/Decel Commands:**	
Command	**Function**	**Command**	**Function**
A	Acceleration	AA	Average Acceleration
AD	Deceleration	ADA	Average Deceleration
LHAD	Hard Limit Deceleration	LHADA	Average Hard Limit Deceleration
LSAD	Soft Limit Deceleration	LSADA	Average Soft Limit Deceleration

Determining the S-Curve Characteristics

The command values for average accel/decel (AA, ADA, etc.) and maximum accel/decel (A, AD, etc.) determine the characteristics of the S-curve. To smooth the accel/decel ramps, you must enter average accel/decel command values that satisfy the equation $\boxed{\frac{1}{2}A \leq AA < A}$, where A represents maximum accel/decel and AA represents average accel/decel. Given this requirement, the following conditions are possible:

Acceleration Setting	**Profiling Condition**
AA > ½ A, but AA < A	S-curve profile with a variable period of constant acceleration. Increasing the AA value above the pure S-curve level (AA > ½ A), the time required to reach the target velocity and the target distance is decreased. However, increasing AA also increases jerk.
AA = ½ A	*Pure S-curve* (no period of constant acceleration—smoothest motion).
AA = A	Trapezoidal profile (but can be changed to an S-curve by specifying a new AA value less than A).
AA < ½ A; or AA > A	When you issue the GO command, the move will not be executed and you will receive the "INVALID_DATA" error.
AA = zero	S-curve profiling is disabled. Trapezoidal profiling is enabled. AA tracks A. (*Track* means the command's value will match the other command's value and will continue to match whatever the other command's value is set to.) However, if you enter an average deceleration command equal to zero, you will receive the "INVALID_DATA" error.
AA ≠ zero and AA ≠ A	S-curve profiling is enabled **only for standard moves** (e.g., not for compiled motion, or on-the-fly motion changes). All subsequent standard moves must comply with this equation: $\frac{1}{2}A \leq AA < A$.
AA > ½ A	Average accel/decel is raised above the pure S-curve level; this decreases the time required to reach the target velocity and distance. However, increasing AA also increases jerk. After increasing AA, you can reduce jerk by increasing A, but be aware that increasing A requires a greater torque to achieve the commanded velocity at the mid-point of the acceleration profile.
No AA value ever entered	Profile will default to trapezoidal. AA tracks A.

If you never change the A or AA commands, ADA will track AA acceleration. However, once you change AD deceleration, ADA deceleration will no longer track changes in AA acceleration.

The calculation for determining S-curve average accel and decel move times is as follows (*calculation method identical for S-curve and trapezoidal moves*):

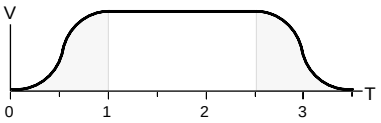
$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \quad \text{or} \quad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

Programming Example

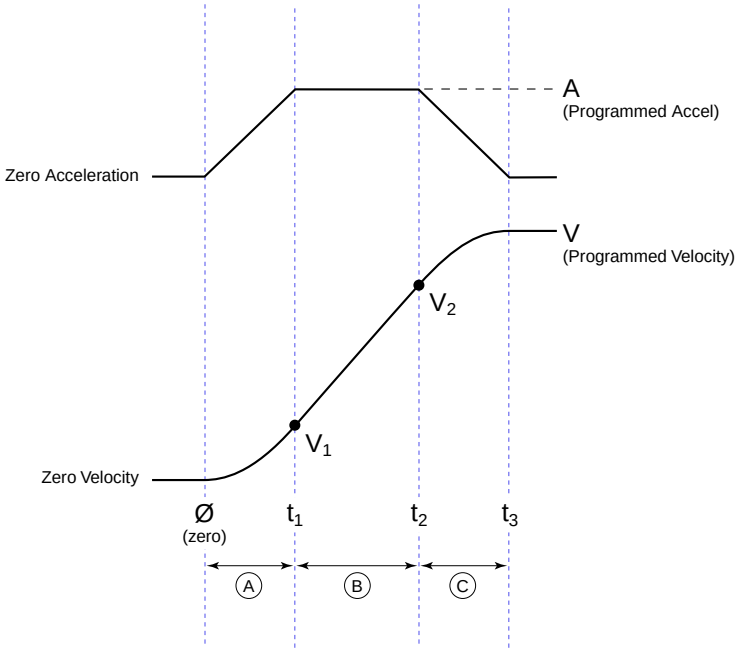
```
; In this example, the Gemini executes a pure S-curve and takes
; 1 second to reach a velocity of 5 rps.

DEF PROG16 ; Define program #16
MC0        ; Select preset positioning mode
MA0        ; Select incremental positioning mode
D40000     ; Set distance to 40,000 positive-direction counts
A10        ; Set max. accel to 10 revs/sec/sec
AA5        ; Set avg. accel to 5 revs/sec/sec
AD10       ; Set max. decel to 10 revs/sec/sec
ADA5       ; Set avg. decel to 5 revs/sec/sec
V5         ; Set velocity to 5 revs/sec
GO1        ; Execute motion
END        ; End definition of program #16
```

Move profile:



Calculating Jerk



Rules of Motion:

$$\text{Jerk} = \frac{da}{dt}$$

$$a = \frac{dv}{dt}$$

$$v = \frac{dx}{dt} \quad (x = \text{distance})$$

Assuming the accel profile starts when the load is at zero velocity and the ramp to the programmed velocity is not compromised:

$$\textbf{Jerk} = J_A = \frac{A^2 * AA}{V\ (A\text{-}AA)}$$

A = programmed acceleration (A, AD, LHAD, LSAD)
AA = average acceleration (AA, ADA, LHADA, LSADA)
V = programmed velocity (V)

(continued on next page)

$$t_1 = \frac{A}{J_A}$$

$$t_2 = \frac{V}{AA} - \frac{A}{J_A}$$

$$t_3 = \frac{V}{AA}$$

NOTE: $t_3 - t_2 = t_1$

$$V_1 = \frac{J_A * t_1^2}{2} = \frac{A^2}{2 * J_A}$$

$$V_2 = V - \frac{A^2}{2 * J_A}$$

Ⓐ $t_1 \geq t \geq \emptyset$

$$a(t) = J_A * t$$

$$v(t) = \frac{J_A * t^2}{2}$$

$$d(t) = \frac{J_A * t^3}{6}$$

a (t) = acceleration at time t
v (t) = velocity at time t
d (t) = distance at time t

Ⓑ $t_2 \geq t > t_1$

$$a(t) = A$$

$$v(t) = \frac{A^2}{2J_A} + A * (t - t_1)$$

$$d(t) = \frac{J_A * t_1^3}{6} + \left(\frac{A * (t - t_1)^2}{2}\right) + \left(V_1 * (t - t_1)\right)$$

Ⓒ $t_3 \geq t > t_2$

$$a(t) = A - (J_A * (t - t_2))$$

$$v(t) = V - \left(\frac{J_A * (t_3 - t)^2}{2}\right)$$

$$d(t) = \frac{V^2}{2AA} + \left(\frac{J_A (t_3 - t)^3}{6}\right) - \left(V * (t_3 - t)\right)$$

Starting at a Non-Zero Velocity: If starting the acceleration profile with a non-zero initial velocity, the move comprises two components: a constant velocity component, and an s-curve component. Typically, the change of velocity should be used in the S-curve calculations. Thus, in the calculations above, you would substitute "$(V_F - V_O)$" for "V" (V_F = final velocity, V_O = initial velocity). For example, the jerk equation would be:

$$\textbf{Jerk} = J_A = \frac{A^2 * AA}{(V_F - V_O)\ (A - AA)}$$