

# Introduction

## Purpose of this Document

This document is designed as a reference for all the 6K Series commands. To gain a full understanding of how the 6K Series commands are used together to implement specific features, refer to the *6K Series Programmer's Guide* (p/n 88-017137-01). For hardware-related information (e.g., electrical wiring connections, specifications, tuning, etc.), refer to the *6K Series Hardware Installation Guide*.

## Table of Contents

Pages 1-20	<i>Introduction:</i> <ul style="list-style-type: none"><li>Command Description Format</li><li>Syntax -- Letters and Symbols</li><li>Syntax -- General Guidelines</li><li>Syntax -- Command Value Substitutions</li><li>Programmable I/O Bit Patterns</li><li>Programming Error Messages</li><li>S-Curve Accel/Decel Profiling</li><li>Units of Measure and Scaling</li></ul>
Pages 21-302	<i>Command Descriptions:</i> Operator symbols are described first, followed by the rest of the 6K Series commands in alphabetical order.
Pages 303-306	<i>Appendix A: 6K Series Command List:</i> Alphabetical list of all 6K Series commands.
Pages 307-308	<i>Appendix B: ASCII Table</i>
Pages 309-312	<i>Appendix C: 6K vs. 6000 Programming Differences</i>
Pages 313-320	<i>Index</i>

# Description of Format

1.	2.	3.
<b>INEN</b>	<b>Input Enable</b>	
4. Type	Inputs or Program Debug Tools	<b>Product</b>
5. Syntax	<!><B>INEN<d><d><d>...<d>	6K
6. Units	d = 0, 1, E, or X	<b>Rev</b>
7. Range	0 = off, 1 = on, E = enable, X = don't care	5.0
8. Default	E	
9. Response	INEN: *INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE	
10. See Also	[IN], INFNC, INLVL, INPLC, INSTW, TIN, TIO	

Item Number	Description
1.	<b>Mnemonic Code:</b> This field contains the command's mnemonic code. If the command is in brackets (e.g., [ IN ]), it is an operator that must be used within the syntax of another command (e.g., IN may be used in a conditional expression like IF( IN.3=b1)).
2.	<b>Full Name:</b> This field contains the command's full name.
3.	<b>Valid Product &amp; Revision:</b> This field lists the 6K Series products and the revision of each product when this command was incorporated or modified per the description. If the command does not apply to that particular product, the <b>Rev</b> is specified as "n/a".  You can use the TREV command to determine which product revision you are using. For example, if the TREV response is *TREV92-012222-01-5.0, the product revision is 5.0.
4.	<b>Type:</b> This field contains the command's type. Inside the back cover you will find a list of all 6K Series commands organized by command type.
5.	<b>Syntax:</b> The proper syntax for the command is shown here. The specific parameters associated with the command are also shown. Definitions of the parameters are described in the <i>Syntax</i> sections below.
6.	<b>Units:</b> This field describes what unit of measurement the parameter (b, d, i, r, or t) in the command syntax represents.
7.	<b>Range:</b> This is the range of valid values that you can specify for an argument (or any other parameter specified).
8.	<b>Default:</b> The default setting for the command is shown in this field. A command will perform its function with the default setting if you do not provide a value.
9.	<b>Response:</b> Some commands allow you to check the status of the command. In the example above, entering the INEN command by itself, you will receive the response *INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE (response indicates all inputs are enabled). The example responses provided are based on the default error level, Error Level 4, established with the ERRLVL4 command.
10.	<b>See Also:</b> Commands related or similar to the command described are listed here.

# Syntax -- Letters and Symbols

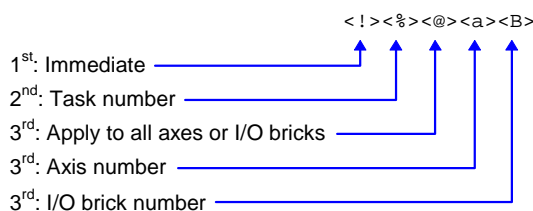
The command descriptions provided within this manual use alphabetic letters and ASCII symbols within the **Syntax** description (see example below) to represent different parameter requirements.

<b>INEN</b>		<b>Input Enable</b>	
Type	Inputs or Program Debug Tools	<b>Product</b>	<b>Rev</b>
→ Syntax	<!><%><B>INEN<d><d><d>...<d>	6K	5.0
Units	d = 0, 1, E, or X		
Range	0 = off, 1 = on, E = enable, X = don't care		
Default	E		
Response	INEN: *INENEEEE EEEE EEEE EEEE EEEE EEEE EEEE		
See Also	[IN], INFNC, INLVL, INPLC, INSTW, TIN, TIO		

<b>Letter/Symbol</b>	<b>Description</b>
a .....	Represents an axis specifier, numeric value from 1 to 8.
B .....	Represents the number of the product's I/O brick. External I/O bricks are represented by numbers 1 through n (to connect external I/O bricks, refer to your product's <i>Installation Guide</i> ). On-board I/O are address at brick location zero (Ø). If the brick identifier is omitted from the command, the controller assumes the command is supposed to affect the onboard I/O.
b * .....	Represents the values 1, 0, X or x; does not require field separator between values.
c .....	Represents a character (A to Z, or a to z)
d .....	Represents the values 1, 0, X or x, E or e ; does not require field separator between values. E or e enables a specific command field. X or x leaves the specific command field unchanged or ignored. In the ANIEN command, the "d" symbol may also represent a real numeric value.
i .....	Represents a numeric value that cannot contain a decimal point (integer values only). The numeric range varies by command. Field separator required.
r .....	Represents a numeric value that may contain a decimal point, but is not required to have a decimal point. The numeric range varies by command. Field separator required.
t .....	Represents a string of alpha numeric characters from 1 to 6 characters in length. The string must start with a alpha character.
! .....	Represents an immediate command. Changes a buffered command to an immediate command. Immediate commands are processed immediately, even before previously entered buffered commands.
% .....	(Multitasking Only) Represents a task identifier. To address the command to a specific task, prefix the command with "i%", where "i" is the task number. For example, the 4% <i>CUT</i> command uses task #4 to execute the program called " <i>CUT</i> ".
, .....	Represents a field separator. Commands with the symbol r or i in their Syntax description require field separators. Commands with the symbol b or d in their Syntax description <b>do not</b> require field separators (but they may be included). See <i>General Guidelines</i> table below.
@ .....	Represents a global specifier, where only one field need be entered. Applicable to all commands with multiple command fields. (e.g., @V1 sets velocity on all axes to 1 rps).
< > .....	Indicates that the item contained within the < > is optional, not required by that command. NOTE: Do not confuse with <cr>, <sp>, and <lf>, which refer to the ASCII characters corresponding to a carriage return, space, and line feed, respectively.
[ ] .....	Indicates that the command between the [ ] must be used in conjunction with another command, and cannot be used by itself.

\* The ASCII character b can also be used within a command to precede a binary number. When the b is used in this context, it is not to be replaced with a 0, 1, X, or x. Examples are assignments such as VARB1=b10001, and comparisons such as IF(3IN=b1001X1).

## Order of Precedence for Command Prefix Characters (from left to right):



# Syntax -- General Guidelines

Guideline Topic	Guideline	Examples
Command Delimiters (<cr>, <lf>, and :)	All commands must be separated by a delimiter. A carriage return is the most commonly used. The colon (:) allows you to place multiple commands on one line of code.	Set acceleration on axis 2 to 10 rev/sec/sec: A,10,,<cr> A,10,,<lf> A,10,,:V,25,,:D,25000,,:@GO<cr>
Neutral Characters (<sp> and <tab>)	Using neutral characters anywhere within a command will not affect the command.	Set velocity on axis 1 to 10 rps, axis 2 to 25 rps: V<sp>10,<sp>25,,<cr>  Add a comment to the command: V 10, 25,,<tab> ;set accel.<cr>
Case Sensitivity	There is no case sensitivity. Use upper or lower case letters within commands.	Initiate motion on axes 1, 3 and 4: GO1011 go1011
Comment Delimiter (:)	All text between a comment delimiter and a command delimiter is considered <i>program comments</i> .	Add a comment to the command: V10<tab> ;set velocity
Field Separator (,)	Commands with the symbol <i>r</i> or <i>i</i> in their Syntax description require field separators.  Commands with the symbol <i>b</i> or <i>d</i> in their Syntax description <b>do not</b> require field separators (but they may be included).  Axes not participating in the command need not be specified; however, field separators that are normally required must be specified (unless the axis prefix is used).	Set velocity on axes 1 - 4 to 10 rps, 25 rps, 5 rps and 10 rps, respectively: V10,25,5,10  Initiate motion on axes 1, 3 and 4: GO1011<cr> GO1,0,1,1  Set velocity on axes 4 and 6 to 5 rps: V,,,5,,5  Alternative is to use the axis prefix: 4V5,,5
Global Command Identifier (@)	When you wish to set the command value equal on all axes, add the @ symbol at the beginning of the command (enter only the value for one command field).  The @ symbol is also useful for checking the status of all axes, or all inputs or outputs on all I/O bricks.	Set velocity on all axes to 10 rps: @V10  Check the status of all digital outputs (onboard, and on external I/O bricks): @OUT
Bit Select Operator (.)	The bit select operator allows you to affect one or more binary bits without having to enter all the preceding bits in the command.  Syntax for setup commands: [command name].[bit #]-[binary value]  Syntax for conditional expressions: [command name].[bit #]=[binary value]	Enable error-checking bit #9: ERROR.9-1  Enable error-check bits #9-12: ERROR.9-1,1,1,1  IF statement based on value of axis status bit #12 for axis #1: IF(1AS.12=b1)
Left-to-right Math	All mathematical operations assume left-to-right precedence.	VAR1=5+3*2 Result: Variable 1 is assigned the value of 16 (8*2), not 11 (5+6).
Binary and Hexadecimal Values	When making assignments with or comparisons against binary or hexadecimal values, you must precede the binary value with the letter "b" or "B", and the hex value with "h" or "H". In the binary syntax, an "x" simply means the status of that bit is ignored.	Binary: IF(IN=b1x01) ↑ Hexadecimal: IF(IN=h7F) ↑
Multi-tasking Task Identifier (%)	Use the % command prefix to identify the command with a specific task.	Launch the "move1" program in Task 1: 1%move1  Check the error status for Task 3: 3%TER  Check the system status for Task 3: 3%TSS

**NOTE:** The command line is limited to 80 characters (excluding spaces).

# Syntax -- Command Value Substitutions

Many commands can substitute one or more of its command field values with one of these substitution items (demonstrated in the programming example below):

VAR.....Places current value of the numeric variable in the corresponding field of the command.  
VARB ..... Uses the value of the binary variable to establish all the fields in the command.  
VARI .....Places current value of the integer variable in the corresponding field of the command.  
READ .....Information is requested at the time the command is executed.  
DREAD.....Reads the RP240's numeric keypad into the corresponding field of the command.  
DREADF .....Reads the RP240's function keypad into the corresponding field of the command.  
TW.....Places the current value set on the thumbwheels in the corresponding field of the command.  
DAT.....Places the current value of the data program (DATP) in the corresponding field of the command.

**Programming Example:** (NOTE: The substitution item must be enclosed in parentheses.)

```
VAR1=15          ; Set variable 1 to 15
A5,(VAR1),4,4    ; Set acceleration to 5,15,4,4 for axes 1-4, respectively
VARB1=b1101XX1  ; Set binary variable 1 to 1101XX1 (bits 5 & 6 not affected)
GO(VARB1)        ; Initiate motion on axes 1, 2 & 4 (value of binary
                 ; variable 1 makes it equivalent to the G01101 command)
OUT(VARB1)       ; Turn on outputs 1, 2, 4, and 7
VAR$1="Enter Velocity" ; Set string variable 1 to the message "Enter Velocity"
V2,(READ1)      ; Set the velocity to 2 on axis 1. Read in the velocity for
                 ; axis 2 , output variable string 1 as the prompting message
                 ; 1. Operator sees "ENTER VELOCITY" displayed on the screen.
                 ; 2. Operator enters velocity prefixed by !' (e.g., !'20).
HOMV2,1,(TW1)   ; Set the home velocity to 2 and 1 on axes 1 and 2, respectively.
                 ; Read in the home velocity for axis 3 from thumbwheel set 1
HOMV2,1,(DAT1)  ; Set the home velocity to 2 and 1 on axes 1 and 2, respectively.
                 ; Read home velocity for axis 3 from data program 1.
VARI1=2*3       ; Set integer variable 1 to 6 (2 multiplied by 3)
D(VARI2),,(VARI3) ; Set the distance of axis 1 equal to the value of
                 ; integer variable 2, and the distance of axis 3 equal to
                 ; the value of integer variable 3.
```

## Rule of Thumb

Not all of the commands allow command field substitutions. In general, commands with a binary command field (<b> in the syntax) will accept the VARB substitution. Commands with a real or integer command field (<r> or <i> in the syntax) will accept VAR, VARI, READ, DREAD, DREADF, TW or DAT.

# Programmable I/O Bit Patterns

The 6K product has programmable inputs and outputs. The total number of onboard inputs and outputs (trigger inputs, limit inputs, digital outputs) depends on the product. The total number of expansion inputs and outputs (analog inputs, digital inputs and digital outputs) depends on your configuration of expansion I/O bricks.

These programmable I/O are represented by binary bit patterns, and it is the bit pattern that you reference when programming and checking the status of specific inputs and outputs. The bit pattern is referenced 1 to *n*, from left to right.

- **Onboard I/O.** For example, the status command to check all onboard trigger inputs is `TIN`.  
An example response for the 6K8 is: `*TIN0100_0001_0000_0011_0`.  

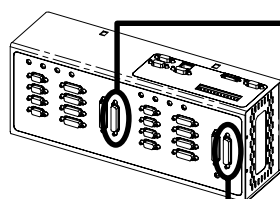
Bit 1 →
← Bit 17
- **Expansion I/O.** For example, the status command to check all digital inputs on I/O brick 2 is `2TIN`.  
An example response for the 6K8 is: `*2TIN0010_0110_1100_0000_XXXX_XXXX_XXXX_XXXX`.  

I/O Brick 2 →
← Bit 1
← Bit 32

## Onboard I/O

I/O	Location	Programming	Status Report, Assignment
Limit Inputs	"LIMITS/HOME" connectors	LIMFNC, LIMEN, LIMLVL	TLIM, LIM
Trigger Inputs	"TRIGGERS/OUTPUTS" connectors (pins 9, 11, 13, 15, 17, 19, 21 & 23). Master Trigger is "MASTER TRIG" on connector on top of the 6K chassis	INFNC, INLVL, INEN, ONIN, INPLC, INSTW	TIN, IN
Outputs (digital)	"TRIGGERS/OUTPUTS" connectors (pins 1, 3, 5 & 7).	OUT, OUTFNC, OUTLVL, OUTEN, OUTALL, OUTPLC, OUTTW, POUT	TOUT, [OUT]

### Limit Inputs ("LIMITS/HOME" connectors)



Input bit pattern for LIM, TLIM, LIMEN, LIMFNC, and LIMLVL:

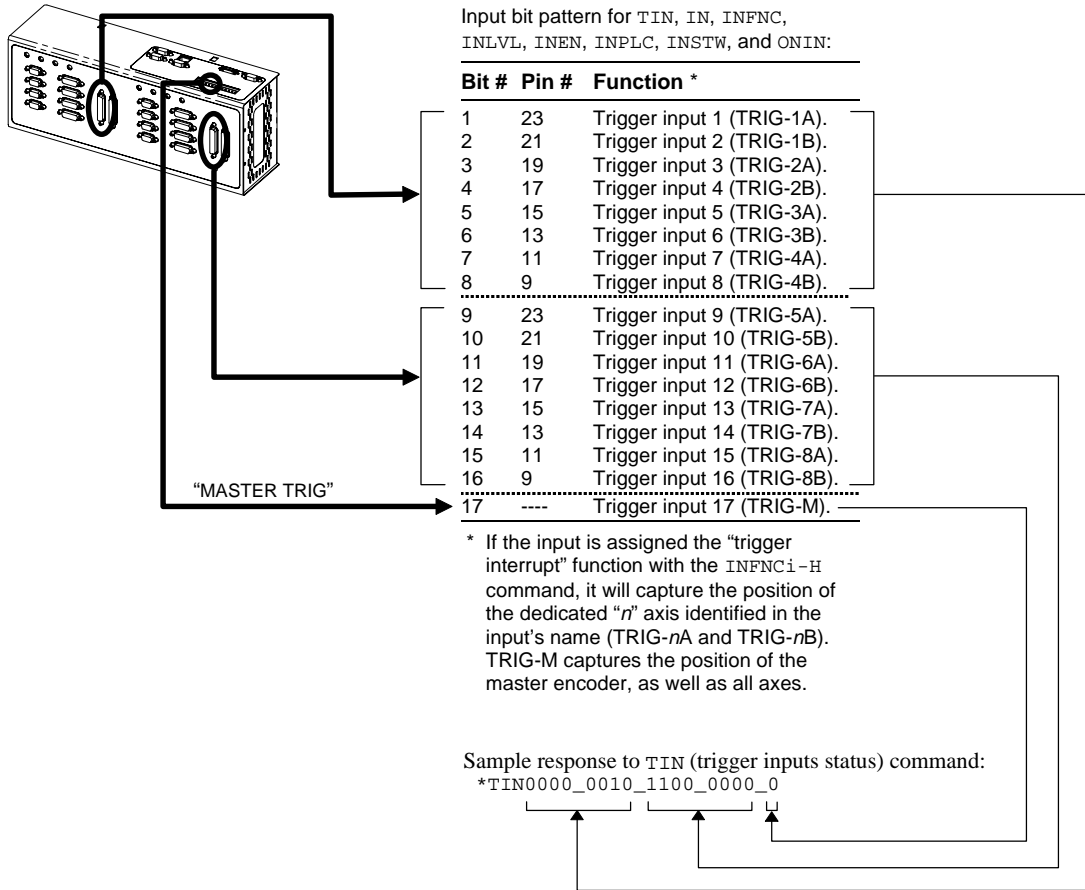
Bit #	Pin #	Function *
1	23	Positive end-of-travel limit, axis 1.
2	21	Negative end-of-travel limit, axis 1.
3	19	Home limit, axis 1.
4	17	Positive end-of-travel limit, axis 2.
5	15	Negative end-of-travel limit, axis 2.
6	13	Home limit, axis 2.
7	11	Positive end-of-travel limit, axis 3.
8	9	Negative end-of-travel limit, axis 3.
9	7	Home limit, axis 3.
10	5	Positive end-of-travel limit, axis 4.
11	3	Negative end-of-travel limit, axis 4.
12	1	Home limit, axis 4.
13	23	Positive end-of-travel limit, axis 5.
14	21	Negative end-of-travel limit, axis 5.
15	19	Home limit, axis 5.
16	17	Positive end-of-travel limit, axis 6.
17	15	Negative end-of-travel limit, axis 6.
18	13	Home limit, axis 6.
19	11	Positive end-of-travel limit, axis 7.
20	9	Negative end-of-travel limit, axis 7.
21	7	Home limit, axis 7.
22	5	Positive end-of-travel limit, axis 8.
23	3	Negative end-of-travel limit, axis 8.
24	1	Home limit, axis 8.

\* The functions listed are the factory default functions; other functions may be assigned with the `LIMFNC` command.

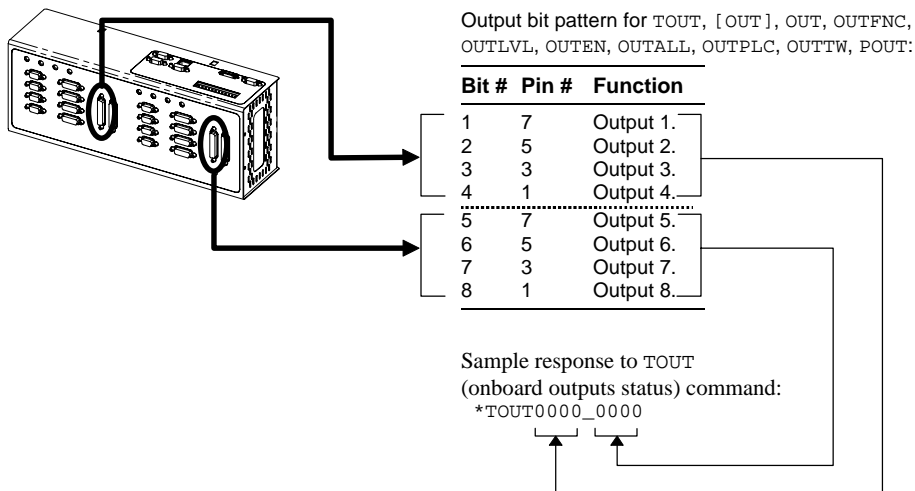
Sample response to `TLIM` (limit inputs status) command:

`*TLIM001_001_001_001_001_001_001_001`

### Trigger Inputs ("TRIGGERS/OUTPUTS" connectors)



### Outputs ("TRIGGERS/OUTPUTS" connectors)



## Expansion I/O Bricks

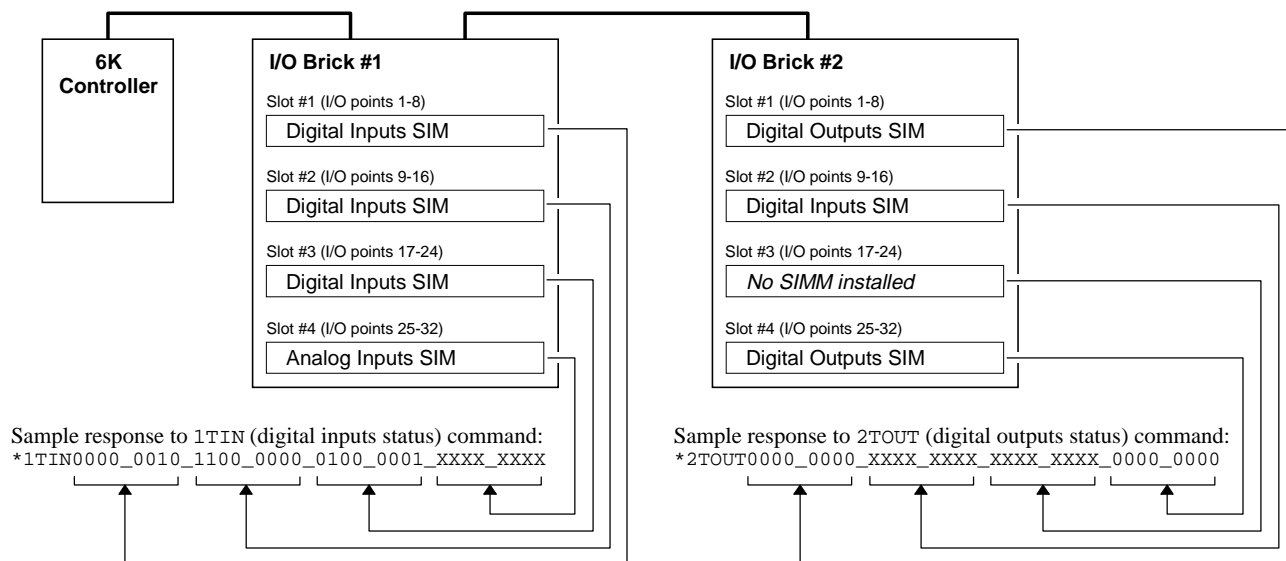
The 6K product allows you to expand your system I/O by connecting up to 8 I/O bricks (see *Installation Guide* for connections). Expansion I/O bricks may be ordered separately (referred to as the “EVM32”). Each I/O brick can hold from 1 to 4 of these I/O SIM modules in any combination:

SIM Type	Programming	Status Report, Assignment
Digital Inputs SIM (8 inputs)	INFNC, INLVL, INEN, ONIN, INPLC, INSTW	TIN, IN
Digital Outputs SIM (8 outputs)	OUT, OUTFNC, OUTLVL, OUTEN, OUTALL, OUTPLC, OUTTW, POUT	TOUT, [OUT]
Analog Inputs SIM (8 inputs)	<ul style="list-style-type: none"> <li>• Enable/Disable: ANIEN.</li> <li>• Joystick setup: JOYAXH, JOYAXL, JOYCDB, JOYCTR, JOYEDB, JOYZ.</li> <li>• Servo feedback: ANIFB, SFB</li> <li>• Following master source: ANIMAS, FOLMAS</li> </ul>	<ul style="list-style-type: none"> <li>• Voltage: TANI, ANI</li> <li>• Servo position: TPANI, PANI, FB, TFB</li> </ul>

Each I/O brick has a unique “brick address”, denoted with the “<B>” symbol in the command syntax. The I/O bricks are connected in series to the “EXPANSION I/O” connector on the 6K. The 1<sup>st</sup> I/O brick has address #1, the next brick has address #2, and so on. (**NOTE:** If you leave out the brick address in the command, the 6K product will assume you are addressing the command to the onboard I/O.) Each I/O brick has 32 I/O addresses, referenced as absolute I/O point locations:

- SIM slot 1 = I/O points 1-8
- SIM slot 2 = I/O points 9-16
- SIM slot 3 = I/O points 17-24
- SIM slot 4 = I/O points 25-32

### Example:



The TIO command identifies the connected I/O bricks (and installed SIMs), including the status of each I/O point:

```
*BRICK 1: SIM Type      Status      Function
          1-8: DIGITAL INPUTS  0000_0000  AAAA_AAAA
          9-16: DIGITAL INPUTS  0000_0000  AAAA_AAAA
          17-24: DIGITAL INPUTS  0000_0000  AAAA_AAAA
          25-32: ANALOG INPUTS   0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000

*BRICK 2: SIM Type      Status      Function
          1-8: DIGITAL OUTPUTS  0000_0000  AAAA_AAAA -- SINKING
          9-16: DIGITAL INPUTS  0000_0000  AAAA_AAAA
          17-24: NO SIM PRESENT
          25-32: DIGITAL OUTPUTS  0000_0000  AAAA_AAAA -- SOURCING
```

# Programming Error Messages

Depending on the error level setting (set with the `ERRLVL` command), when a programming error is created, the 6K controller will respond with an error message and/or an error prompt. A list of all possible error messages is provided in a table below. The default error prompt is a question mark (?), but you can change it with the `ERRBAD` command if you wish.

At error level 4 (`ERRLVL4`—the factory default setting) the 6K controller responds with both the error message and the error prompt. At error level 3 (`ERRLVL3`), the 6K controller responds with only the error prompt.

Error Response	Possible Cause
ACCESS DENIED	Program security feature enabled, but the <i>program access input</i> ( <code>INFNCi-Q</code> or <code>LIMFNCi-Q</code> ) is not activated.
ALREADY DEFINED FOR THUMBWHEELS	Attempting to assign an I/O function to an I/O that is already defined as a thumbwheel I/O.
ALTERNATIVE TASK NOT ALLOWED	Attempting to execute a <code>LOCK</code> command directed to another task.
AXES NOT READY	Compiled Profile path compilation error.
COMMAND NOT IMPLEMENTED	Command is not applicable to the 6K Series product.
COMMAND NOT ALLOWED IN PROGRAM	Command is not allowed inside a program definition (between <code>DEF</code> and <code>END</code> ).
COMMAND/DRIVE MISMATCH	The command (or $\geq$ one field in the command) is not appropriate to the <code>AXSDEF</code> configuration (e.g., attempting to execute a servo tuning command on a stepper axis)
ERROR: MOTION ENDS IN NON-ZERO VELOCITY - AXIS N	Compiled Motion: The last <code>GOBUF</code> segment within a <code>PLOOP/PLN</code> loop does not end at zero velocity, or there is no final <code>GOBUF</code> segment placed outside the loop.
EXCESSIVE PATH RADIUS DIFFERENCE	Contouring path compilation error.
FOLMAS NOT SPECIFIED	No <code>FOLMAS</code> for the axis is currently specified. It will occur if <code>FMCNEW</code> , <code>FSHFC</code> , or <code>FSHFD</code> commands are executed and no <code>FOLMAS0</code> command was executed, or <code>FOLMAS0</code> was executed.
INCORRECT AXIS	Axis specified is incorrect.
INCORRECT BRICK NUMBER	Attempted to execute a command that addresses an I/O brick that is not connected to your 6K controller.
INCORRECT DATA	Incorrect command syntax. Following: Velocity ( <code>v</code> ), acceleration ( <code>A</code> ) or deceleration ( <code>AD</code> ) command is zero (used by <code>FSHFC</code> & <code>FSHFD</code> ).
INPUT(S) NOT DEFINED AS JOYSTICK INPUT	Attempted to execute <code>JOYCDB</code> , <code>JOYCTR</code> , <code>JOYEDB</code> , or <code>JOYZ</code> before executing <code>JOYAXH</code> or <code>JOYAXL</code> to assign the analog input to an axis.
INSUFFICIENT MEMORY	Not enough memory for the user program or compiled profile segments. This may be remedied by reallocating memory (see <code>MEMORY</code> command description).
INVALID COMMAND	Command is invalid because of existing conditions

## Programming Error Messages *(continued)*

Error Response	Possible Cause
INVALID CONDITIONS FOR COMMAND	<p>System not ready for command (e.g., LN command issued before the L command).</p> <p>Following (these conditions can cause an error during Following):</p> <ul style="list-style-type: none"> <li>The FOLMD value is too small to achieve the preset distance and still remain within the FOLRN/FOLRD ratio.</li> <li>A phase shift cannot be performed: <ul style="list-style-type: none"> <li>FSHFD .... Error if already shifting or performing other time based move.</li> <li>FSHFC .... Error if currently executing a FSHFD move, or if currently executing another FSHFC move in the opposite direction.</li> </ul> </li> <li>The FOLEN1 command was given while a profile was suspended by a GOWHEN.</li> </ul>
INVALID CONDITIONS FOR S_CURVE ACCELERATION—FIELD <i>n</i>	Average (AA) acceleration or deceleration command (e.g., AA, ADA, HOMAA, HOMADA, etc.) with a range that violates the equation $\frac{1}{2}A \leq AA \leq A$ (A is the max. accel or decel command—e.g., A, AD, HOMA, HOMAD, etc.)
INVALID DATA	<p>Data for a command is out of range.</p> <p>Following (these conditions can cause an error during Following):</p> <ul style="list-style-type: none"> <li>The parameter supplied with the command is valid. <ul style="list-style-type: none"> <li>FFILT .... Error if: smooth number is not 0-4</li> <li>FMCLEN.. Error if: master steps &gt; 999999999 or negative</li> <li>FMCP ..... Error if: master steps &gt; 999999999 or &lt; -999999999</li> <li>FOLMD .... Error if: master steps &gt; 999999999 or negative</li> <li>FOLRD .... Error if: master steps &gt; 999999999 or negative</li> <li>FOLRN .... Error if: follower steps &gt; 999999999 or negative</li> <li>FSHFC .... Error if: number is not 0-3</li> <li>FSHFD .... Error if: follower steps &gt; 999999999 or &lt; -999999999</li> <li>GOWHEN.. Error if: position &gt; 999999999 or &lt; -999999999</li> <li>WAIT ..... Error if: position &gt; 999999999 or &lt; -999999999</li> </ul> </li> <li>Error if a GO command is given in the preset positioning mode (MCØ) and: <ul style="list-style-type: none"> <li>FOLRN = zero</li> <li>FOLMD = zero, or too small</li> </ul> <p>(see Following chapter in the <i>Programmer's Guide</i>)</p> </li> </ul>
INVALID FOLMAS SPECIFIED	Following: An illegal master was specified in FOLMAS. A follower may never use its own commanded position or feedback source as its master.
INVALID RATIO	Following: Error if the FOLRN:FOLRD ratio after scaling is > 127 when a GO is executed
INVALID TASK IDENTIFIER	Attempting to launch a PEXE or EXE command into the supervisor task (task 0).
LABEL ALREADY DEFINED	Defining a program or label with an existing program name or label name
MAXIMUM COMMAND LENGTH EXCEEDED	Command exceeds the maximum number of characters
MAXIMUM COUNTS PER SECOND EXCEEDED	Velocity value is greater than 1,600,000 counts/sec
MOTION IN PROGRESS	<p>Attempting to execute a command not allowed during motion (see Restricted Commands During Motion section in the <i>Programmer's Guide</i>.)</p> <p>Following: The FOLEN1 command was given while that follower was moving in a non-Following mode.</p>

## Programming Error Messages *(continued)*

Error Response	Possible Cause
NEST LEVEL TOO DEEP	IFs, REPEATs, WHILEs, or GOSUBs nested greater than 16 levels (for each type)
NO MOTION IN PROGRESS	Attempting to execute a command that requires motion, but motion is not in progress
NO PATH SEGMENTS DEFINED	Compiled Profile compilation error
NO PROGRAM BEING DEFINED	END command issued before a DEF command
NOT ALLOWED IF SFBØ	Changes to tuning commands (SGILIM, SGAF, SGI, SGP, SGV, and SGVF) and SMPER are not allowed if SFBØ is selected
NOT ALLOWED IN PATH	Compiled Profile path compilation error
NOT DEFINING A PATH	Executing a compiled profile or contouring path command while not in a path
NOT VALID DURING FOLLOWING MOTION	A GO command was given while moving in the Following mode (FOLEN1) and while in the preset positioning mode (MCØ).
NOT VALID DURING RAMP	A GO command was given while moving in a Following ramp and while in the continuous positioning mode (MC1). Following status (FS) bit #3 will be set to 1. A FOLEN command was given during one of these conditions: <ul style="list-style-type: none"> <li>• During a shift (FSHFC or FSHFD)</li> <li>• During a change in ratio (FOLRN/FOLRD)</li> <li>• During deceleration to a stop</li> </ul>
PATH ALREADY MOVING	Compiled Profile path compilation error
PATH NOT COMPILED	Attempting to execute a individual axis profile or a multiple axis contouring path that has not been compiled
PATH RADIUS TOO SMALL	Contouring path compilation error
PATH RADIUS ZERO	Contouring path compilation error
PATH VELOCITY ZERO	Contouring path compilation error
STRING ALREADY DEFINED	A string (program name or label) with the specified name already exists
STRING IS A COMMAND	Defining a program or label that is a command or a variant of a command
UNDEFINED LABEL	Command issued to product is not a command or program name
WARNING: POINTER HAS WRAPPED AROUND TO DATA POINT 1	During the process of writing data (DATTC) or recalling data (DAT), the pointer reached the last data element in the program and automatically wrapped around to the first datum in the program
WARNING: ENABLE INPUT INACTIVE	<b>ENABLE</b> input is no longer connected to ground ( <b>GND</b> )
WARNING: DEFINED WITH ANOTHER TW/PLC	Duplicate I/O in multiple thumbwheel definitions

## Identifying Bad Commands

To facilitate program debugging, the Transfer Command Error (TCMDER) command allows you to transfer the first command that the controller detects as an error. This is especially useful if you receive an error message when running or downloading a program, because it catches and remembers the command that caused the error.

### Using Motion Planner:

If you are typing the command in a live terminal emulator session, the controller will detect the bad command and respond with an error message, followed by the `ERRBAD` error prompt (?). If the bad command was detected on download, the bad command is reported automatically (see example below).

**NOTE:** If you are not using Motion Planner, you'll have to type in the `TCMDER` command at the error prompt to display the bad command.

Once a command error has occurred, the command and its fields are stored and system status bit #11 (reported in the `TSSF`, `TSS` and `SS` commands) is set to 1. The status bit remains set until the `TCMDER` command is issued.

### Example Error Scenario:

1. In Motion Planner's program editor, create and save a program with a programming error:

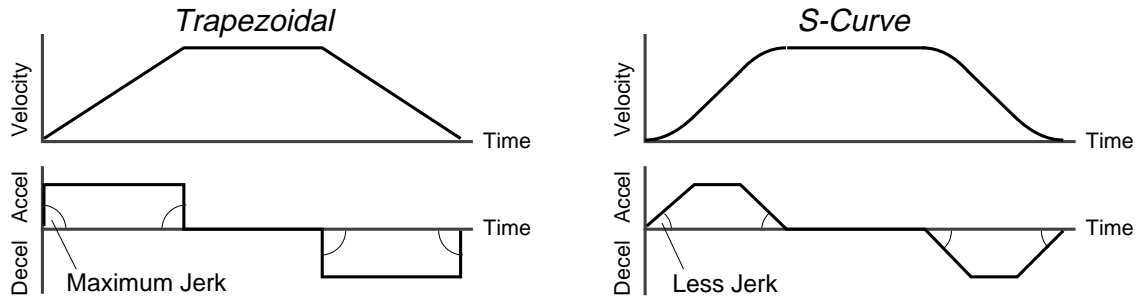
```
DEL badprg ; Delete a program before defining and downloading
DEF badprg ; Begin definition of program called badprg
MA11      ; Select the absolute preset positioning mode
A25,40    ; Set acceleration
AD11,26   ; Set deceleration
V5,8      ; Set velocity
VAR1=0    ; Set variable #1 equal to zero
GO11     ; Initiate move on both axes
IF(VAR1<)16 ; MISTYPED IF STATEMENT - should be typed as "IF(VAR1<16)"
VAR1=VAR1+1 ; If variable #1 is less than 16, increment the counter by 1
NIF      ; End IF statement
END      ; End programming of program called badprg
```

2. Using Motion Planner's terminal emulator, download the program to the 6K Series product. Notice that an error response identifies the bad command as an "INCORRECT DATA" item and displays it:

```
> *NO ERRORS
*INCORRECT DATA
> *IF(VAR1<)16
>
```

# S-Curve Acceleration/Deceleration Profiling

6K controllers 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, especially in linear interpolation applications (not contouring).



## S-Curve Programming Requirements

To program an S-curve profile, you must use the *average accel/decel* commands provided in the 6K Series programming language. For every maximum accel/decel command (e.g., A, AD, HOMA, HOMAD, JOGA, JOGAD, etc.) there is an *average* command for S-curve profiling (see table below).

Maximum Accel/Decel Commands:		Average ("S-Curve") Accel/Decel Commands:	
Command	Function	Command	Function
A	Acceleration	AA	Average Acceleration
AD	Deceleration	ADA	Average Deceleration
HOMA	Home Acceleration	HOMAA	Average Home Acceleration
HOMAD	Home Deceleration	HOMADA	Average Home Deceleration
JOGA	Jog Acceleration	JOGAA	Average Jog Acceleration
JOGAD	Jog Deceleration	JOGADA	Average Jog Deceleration
JOYA	Joystick Acceleration	JOYAA	Average Joystick Acceleration
JOYAD	Joystick Deceleration	JOYADA	Average Joystick Deceleration
LHAD	Hard Limit Deceleration	LHADA	Average Hard Limit Deceleration
LSAD	Soft Limit Deceleration	LSADA	Average Soft Limit Deceleration
PA	Path Acceleration	PAA	Average Path Acceleration
PAD	Path Deceleration	PADA	Average Path 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  $\frac{1}{2} A \leq AA < A$ , 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 an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION-FIELD n, will be displayed.
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.)
AA ≠ zero and AA ≠ A	S-curve profiling is enabled <b>only for standard moves</b> (e.g., not for contouring, which requires the PADA and/or PAA commands). All subsequent standard moves for that axis 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 deceleration commands, AA deceleration will track AA acceleration. However, once you change A deceleration, AA 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*):

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

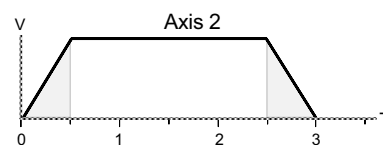
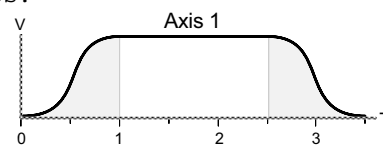
**Scaling** affects the AA average acceleration (AA, ADA, etc.) the same as it does for the A maximum acceleration (A, AD, etc.). See page 16 for details on scaling.

**NOTE:** Equations for calculating jerk are provided on page 15.

### Programming Example (see move profile drawings below)

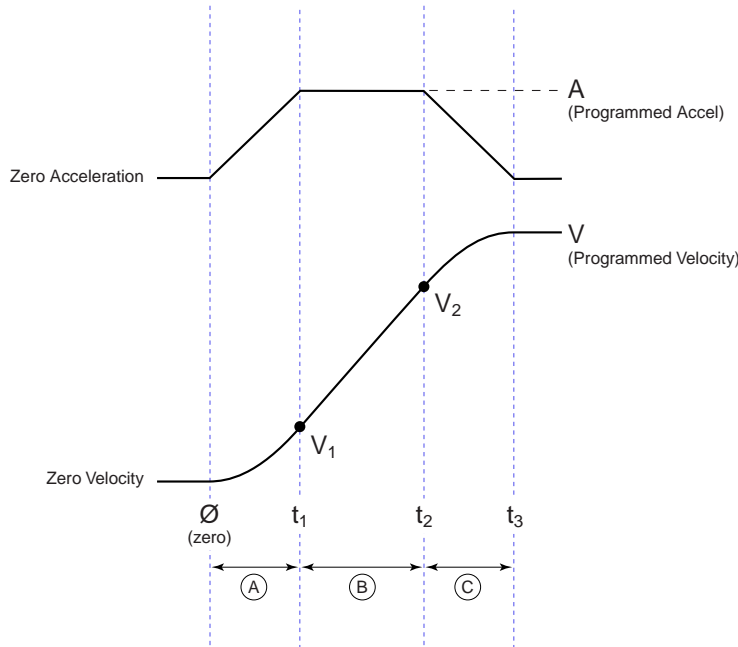
```
; In this example, axis 1 executes a pure S-curve and takes 1 second
; to reach a velocity of 5 rps; axis 2 executes a trapezoidal profile
; and takes 0.5 seconds to reach a velocity of 5 rps.
```

```
SCALE0      ; Disable scaling
DEF SCURV   ; Begin definition of program SCURV
@MA0       ; Select incremental positioning mode
@D40000    ; Set distances to 40,000 positive-
           ; direction steps
A10,10     ; Set max. accel to 10 rev/sec/sec
           ; on axes 1 and 2
AA5,10     ; Set avg. accel to 5 rev/sec/sec on
           ; axis 1, & 10 rev/sec/sec on axis 2
AD10,10    ; Set max. decel to 10 rev/sec/sec
           ; on axes 1 and 2
ADA5,10    ; Set avg. decel to 5 rev/sec/sec on
           ; axis 1, & 10 rev/sec/sec on axis 2
V5,5       ; Set velocity to 5 rps on axes 1 & 2
GO11      ; Execute motion on axes 1 and 2
END        ; End definition of program
```



Move profiles

## 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:

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

A = programmed acceleration  
(A, AD, HOMAD, etc.)

AA = average acceleration  
(AA, ADA, HOMAA, etc.)

V = programmed velocity  
(V, HOMV, etc.)

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

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

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

NOTE:  $t_3 - t_2 = t_1$

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

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

Ⓐ  $t_1 \geq t \geq 0$

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

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

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

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:

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

# Units of Measure and Scaling

## Units of Measure without Scaling

Scaling is disabled (SCALEØ) as the factory default condition:

- Stepper axes: All distance values entered are in commanded counts (sometimes referred to as *motor steps*), and all acceleration, deceleration and velocity values entered are internally multiplied by the DRES command value.

Motion Attribute	Units of Measure (per feedback source)	
	Encoder	Analog Input
Accel/Decel	Revs/sec/sec *	volts/sec/sec
Velocity	Revs/sec *	volts/sec
Distance	Counts **	Counts **

\* All accel/decel & velocity values are internally multiplied by the ERES command value.

\*\* Distance is measured in the counts received from the feedback device.

**Contouring & Linear Interpolated Motion:** Path acceleration, velocity, and distance are based on the resolution (DRES for steppers, ERES for servos) of axis 1. If multi-tasking is used, path motion units are based on the resolution of the first (lowest number) axis associated with the task (TSKAX).

## What is Scaling?

Scaling allows you to program acceleration, deceleration, velocity, and position values in units of measure that are appropriate for your application. The SCALE command is used to enable or disable scaling (SCALE1 to enable, SCALEØ to disable). The motion type(s) you are using in your application determines which scale factor commands you need to configure:

Type of Motion	Accel/Decel Scaling	Velocity Scaling	Distance Scaling
Standard Point-to-Point Motion	SCLA	SCLV	SCLD
Contouring, Linear Interpolation	SCLD	SCLD	SCLD
Following	SCLA	SCLV	SCLD for follower distances SCLMAS for master distances

## When Should I Define Scaling Factors?

Scaling calculations are performed when a program is defined or downloaded. Consequently, you must enable scaling (SCALE1) and define the scaling factors (SCLD, SCLA, SCLV, SCLMAS) *prior* to defining (DEF), uploading (TPROG), or running (RUN) the program.

**RECOMMENDATION:** Place the scaling commands at the beginning of your program file, *before* the location of any defined programs. This ensures that the motion parameters in subsequent programs in your program file are scaled correctly. When you use Motion Planner's Setup Generator wizard, the scaling commands are automatically placed in the appropriate location in your program file.

**ALTERNATIVE:** Scaling factors could be defined via a terminal emulator *just before* defining or downloading a program. Because scaling command values are saved in battery-backed RAM (remembered until you issue a RESET command), all subsequent program definitions and downloads will be scaled correctly.

### NOTES

- Scaling commands are not allowed in a program. If there are scaling commands in a program, the controller will report an error message ("COMMAND NOT ALLOWED IN PROGRAM") when the program is downloaded.
- If you intend to upload a program with scaled motion parameters, be sure to use Motion Planner. Motion Planner automatically uploads the scaling parameters and places them at the beginning of the program file containing the uploaded program from the controller. This ensures correct scaling when the program file is later downloaded.

## Servo Axes

Scaling can be used with encoder or analog input feedback sources. When the scaling commands (SCLA, SCLD, etc.) are executed, they are specific only to the current feedback source selected with the last SFB command.

If your application requires switching between feedback sources for the same axis, then for each feedback source, you must select the feedback source with the appropriate SFB command and issue the scaling factors specific to operating with that feedback source.

For example, if you have two axes and will be switching between encoder and ANI feedback, you should include code similar to the following in your setup program:

```
SFB1,1           ; Select encoder feedback (subsequent scaling
                  ; parameters are specific to encoder feedback)
SCLA4000,4000    ; Program accel/decel in revs/sec/sec
SCLV4000,4000    ; Program velocity in revs/sec
SCLD4000,4000    ; Program distances in revs
SFB2,2           ; Select ANI feedback (subsequent scaling
                  ; parameters are specific to ANI feedback)
SCLA205,205      ; Program accel/decel in volts/sec/sec
SCLV205,205      ; Program velocity in volts/sec
SCLD205,205      ; Program distances in volts
```

## Acceleration & Deceleration Scaling (SCLA)

**Stepper Axes:** If scaling is enabled (SCALE1), all accel/decel values entered are internally multiplied by the acceleration scaling factor to convert user units/sec/sec to commanded counts/sec/sec. The scaled values are always in reference in commanded counts, regardless of the existence of an encoder.

**Servo Axes:** If scaling is enabled (SCALE1), all accel/decel values entered are internally multiplied by the acceleration scaling factor to convert user units/sec/sec to encoder or analog input counts/sec/sec.

All accel/decel commands (e.g., A, AA, AD, HOMA, HOMAD, JOGA, etc.) are multiplied by the SCLA command value. **NOTE:** Path accel/decel commands (PA, PAD, etc.) are multiplied by the SCLD value.

As the accel/decel scaling factor (SCLA) changes, the resolution of the accel and decel values and the number of positions to the right of the decimal point also change (see table at right). An accel/decel value with greater resolution than allowed will be truncated (e.g., if scaling is set to SCLA10, the A9.9999 command would be truncated to A9.9).

SCLA value (counts/unit/unit)	Decimal Places
1 - 9 .....	0
10 - 99 .....	1
100 - 999 .....	2
1000 - 9999 .....	3
10000 - 99999 .....	4
100000 - 999999 .....	5

The following equations can help you determine the range of acceleration and deceleration values.

Axis Type	Min. Accel or Decel (resolution)	Max. Accel or Decel
Stepper	$\frac{0.001 * DRES}{SCLA}$	$\frac{999.9999 * DRES}{SCLA}$
Servo	Encoder Feedback: $\frac{0.001 * ERES}{SCLA}$	Encoder Feedback: $\frac{999.9999 * ERES}{SCLA}$
	ANI Feedback: * $\frac{0.205}{SCLA}$	ANI Feedback: * $\frac{204799.9795}{SCLA}$

\* This calculation assumes the analog input range (ANIRNG value) is left in its default setting (range is -10V to +10V).

## Velocity Scaling (SCLV)

**Stepper Axes:** If scaling is enabled (SCALE1), all velocity values entered are internally multiplied by the velocity scaling factor to convert user units/sec to commanded counts/sec. The scaled values are always in reference to commanded counts (sometimes referred to as “motor steps”).

**Servo Axes:** If scaling is enabled (SCALE1), all velocity values entered are internally multiplied by the velocity scaling factor to convert user units/sec to encoder or analog input counts/sec.

All velocity commands (e.g., V, HOMV, HOMVF, JOGVH, JOGVL, etc.) are multiplied by the SCLV command value. **NOTE:** Path velocity (PV) is multiplied by the SCLD value.

As the velocity scaling factor (SCLV) changes, the velocity command's range and its decimal places also change (see table below). A velocity value with greater resolution than allowed will be truncated. For example, if scaling is set to SCLV10, the V9.9999 command would be truncated to V9.9.

SCLV Value (counts/unit)	Velocity Resolution (units/sec)	Decimal Places
1 - 9	1	0
10 - 99	0.1	1
100 - 999	0.01	2
1000 - 9999	0.001	3
10000 - 99999	0.0001	4
100000 - 999999	0.00001	5

Use the following equations to determine the maximum velocity range for your product type.

Max. Velocity for Stepper Axes		Max. Velocity for Servo Axes (determined by feedback source selected for axis #1)	
$\frac{n}{SCLV}$	$n$ = maximum velocity is determined by the PULSE command setting.	Encoder Feedback:	$\frac{6,500,000}{SCLV}$
		ANI Feedback: *	$\frac{1000 * 205}{SCLV}$

\* This calculation assumes the analog input range (ANIRNG value) is left in its default setting (range is -10V to +10V).

## Distance Scaling (SCLD and SCLMAS)

**Stepper Axes:** If scaling is enabled (SCALE1), all distance values entered are internally multiplied by the distance scaling factor to convert user units to commanded counts (“motor steps”).

**Servo Axes:** If scaling is enabled (SCALE1), all distance values entered are internally multiplied by the distance scaling factor to convert user units to encoder or analog input counts.

All distance commands (e.g., D, PSET, REG, SMPER) are multiplied by the SCLD command value. The only exception is for master distance values (see table below)

**Scaling for Following Motion:** The SCLD command defines the follower axis distance scale factor, and the SCLMAS command defines the master's distance scale factor. The Following-related commands that are affected by SCLD and SCLMAS are listed in the table below.

### Commands Affected by Master Scaling (SCLMAS)

FMCLN: *Master Cycle Length*  
 FMCP: *Master Cycle Position Offset*  
 FOLMD: *Master Distance*  
 FOLRD: *Follower-to-Master Ratio (Denominator)*  
 GOWHEN: *Conditional GO (left-hand variable is PMAS)*  
 TPMAS & [PMAS]: *Position of Master Axis*  
 TVMAS & [VMAS]: *Velocity of Master Axis*

### Commands Affected by Follower Scaling (SCLD)

FOLRN: *Follower-to-Master Ratio (Numerator)*  
 FGADV: *Geared Advance*  
 FSHFD: *Preset Phase Shift*  
 GOWHEN: *Conditional GO (left-hand variable ≠ PMAS)*  
 TPSHF & [PSHF]: *Net Position Shift of Follower*  
 TPSLV & [PSLV]: *Position of Follower Axis*

As the SCLD or SCLMAS scaling factor changes, the distance command's range and its decimal places also change (see table below). A distance value with greater resolution than allowed will be truncated. For example, if scaling is set to SCLD4000, the D105.2776 command would be truncated to D105.277.

SCLD or SCLMAS Value (counts/unit)	Distance Resolution (units)	Distance Range * (units)	Decimal Places
1 - 9	1.0	0 - ±999999999	0
10 - 99	0.10	0.0 - ±99999999.9	1
100 - 999	0.010	0.00 - ±9999999.99	2
1000 - 9999	0.0010	0.000 - ±999999.999	3
10000 - 99999	0.00010	0.0000 - ±99999.9999	4
100000 - 999999	0.00001	0.00000 - ±9999.99999	5

**NOTE                      FRACTIONAL STEP TRUNCATION                      NOTE**

If you are operating in the incremental mode (MAØ), or specifying master distance values with FOLMD, when the distance scaling factor (SCLD or SCLMAS) and the distance value are multiplied, a fraction of one step may be left over. This fraction is truncated when the distance value is used in the move algorithm. This truncation error can accumulate over a period of time, when performing incremental moves continuously in the same direction. To eliminate this truncation problem, set SCLD or SCLMAS to 1, or a multiple of 10.

### Scaling Example — Stepper Axes

Axis #1 and axis #2 control 25,000 step/rev motor/drives attached to 5-pitch leadscrews. The user wants to program motion parameters in inches; therefore the scale factor calculation is: 25,000 steps/rev x 5 revs/inch = 125,000 steps/inch. For instance, with a scale factor of 125,000, the operator could enter a move distance value of 2.000 and the controller would send out 250,000 pulses, corresponding to two inches of travel.

```
SCALE1           ; Enable scaling
DRES25000,25000  ; Set drive resolution to 25,000 steps/rev on both axes
SCLD125000,125000 ; Allow user to enter distance in inches (both axes)
SCLV125000,125000 ; Allow user to enter velocity in inches/sec (both axes)
SCLA125000,125000 ; Allow entering accel/decel in inches/sec/sec (both axes)
```

### Scaling Example — Servo Axes

Axis #1 controls a 4,000 count/rev servo motor/drive system (using a 1000-line encoder) attached to a 5-pitch leadscrew. The user wants to position in inches; therefore, the scale factor calculation is 4,000 counts/rev x 5 revs/inch = 20,000 counts/inch. Half way through the motion process, axis #1 must switch to ANI feedback for the purpose of positioning to a voltage (scale factor is 205 counts/volt).

Axis #2 controls a 4,000 count/rev servo motor/drive system (using a 1000-line encoder) attached to a 10-pitch leadscrew. The user wants to position in inches (scale factor calculation: 4,000 counts/rev x 10 revs/inch = 40,000 counts/inch).

```
SFB1,1           ; Select encoder feedback for both axes
ERES4000,4000    ; Set encoder res to 4000 steps/rev (post quadrature)
SCALE1           ; Enable scaling
SCLD20000,40000  ; Allow user to enter distance values in inches
SCLV20000,40000  ; Allow user to enter velocity values in inches/sec
SCLA20000,40000  ; Allow user to enter accel/decel values in inches/sec/sec
SFB2             ; Select ANI feedback for axis #1
SCALE1           ; Enable scaling
SCLD205          ; Allow user to enter distance values in volts
SCLV205          ; Allow user to enter velocity values in volts/sec
SCLA205          ; Allow user to enter accel/decel values in volts/sec/sec
SFB1,1           ; Select encoder feedback for both axes (prepare for motion)
```

## Scaling Example — Following

Typically, the master and follower scale factors are programmed so that master and follower units are the same, but this is not required. Consider the scenario below as an example.

The master is a 1000-line encoder (4000 counts/rev post-quadrature) mounted to a 50 teeth/rev pulley attached to a 10 teeth/inch conveyor belt, resulting in 80 counts/tooth (4000 counts/50 teeth = 80 counts/tooth). To program in inches, you would set up the master scaling factor with the SCLMAS800 command (80 counts/tooth \* 10 teeth/inch = 800 counts/inch).

The follower axis is a servo motor with position feedback from a 1000-line encoder (4000 counts/rev). The motor is mounted to a 4-pitch (4 revs/inch) leadscrew. Thus, to program in inches, you would set up the follower scaling factor with the SCLD16000 command (4000 counts/rev \* 4 revs/inch = 16000 counts/inch).

```
SCALE1      ; Enable scaling
SCLMAS800   ; Master scaling (80 counts/tooth * 10 teeth/inch = 800 counts/inch)
SCLD16000   ; Follower scaling (4000 counts/rev * 4 revs/inch = 16000 counts/inch)
```

## Scaling Example — Contouring & Linear Interpolation

This simple example uses 2 servo axes (axes 1 and 2) for contouring. Both axes use encoder feedback with a resolution (ERES) of 4000 counts/rev, axis 1 uses a 10-pitch (10 revs per inch) leadscrew and axis 2 uses a 5-pitch (5 revs per inch) lead screw, and you want to program in inches. For this application you would use the SCLD40000, 20000 command to establish path motion units in inches: distance is inches, acceleration is inches/sec/sec, and velocity is inches/sec. Note that all path motion attributes are scaled by the SCLD value.

```
SCALE1      ; Enable scaling
SCLD40000,20000 ; Set scaling to program in inches:
                ; Axis 1: 4000 counts/rev * 10 revs/inch = 40000 counts/inch
                ; Axis 2: 4000 counts/rev * 5 revs/inch = 20000 counts/inch
PV5         ; Set path velocity to 5 inches/sec
PA50       ; Set path acceleration to 50 inches/sec/sec
PAD100     ; Set path deceleration to 100 inches/sec/sec
DEF prog1  ; Begin definition of path named prog1
PAXES1,2   ; Set axes 1 and 2 as the X and Y contouring axes
PAB0       ; Set to incremental coordinates
PLIN1,1    ; Specify X-Y endpoint position to create a 45 degree
                ; angle line segment
END        ; End definition of path prog1
PCOMP prog1 ; Compile path prog1
PRUN prog1 ; Execute path prog1
```