
HALT Terminate Program Execution

Type	Program Flow Control	Product	Rev
Syntax	<!>HALT	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	BP, BREAK, C, ELSE, IF, K, NIF, NWHILE, PS, REPEAT, S, T, UNTIL, WAIT, WHILE		

The Terminate Program Execution (HALT) command terminates program execution when processed. This command allows the user to terminate command processing at any point in a program. The programmer may want processing to stop because of an error condition, an input, a variable, or just after a specific motion has been accomplished. This command is useful when debugging a program.

Example:

```
DEF prog1      ; Define a program called prog1
GO1000        ; Initiate motion on axis 1
GOSUB prog2    ; Gosub to subroutine named prog2
GO0100        ; Initiate motion on axis 2
END           ; End program definition
DEF prog2      ; Define a program called prog2
GO1110        ; Initiate motion on axes 1, 2, and 3
IF(IN=b1X0)    ; If onboard input 1 is active (1), and input 3 is inactive (0)
HALT          ; If condition is true break out of program
ELSE          ; Else part of if condition
TPE           ; If condition does not come true transfer position of all
              ; encoders to PC
NIF           ; End If statement
END           ; End program definition
RUN prog1      ; Execute program prog2
;
; Upon completion of motion on axis 1, subroutine prog2 is called.
; If inputs 1 and 3 are in the correct state after the motion is complete,
; program processing will be terminated. In other words, all commands waiting
; to be parsed in the program buffer will be eliminated.
; **** Note: There will not be a return to prog1.
```

HELP Technical Support

Type	Program Debug Tool	Product	Rev
Syntax	<!>HELP	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	See description below		
See Also	None		

The (HELP) command provides the telephone numbers for technical support.

HOM

Go Home

Type	Homing	Product	Rev
Syntax	<!><@>HOM	6K	5.0
Units	n/a		
Range	b = 0 (home in positive direction), 1 (home in negative direction), or X (do not home)		
Default	X		
Response	n/a		
See Also	[AS], HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ, [LIM], LIMEN, LIMLVL, PSET, TAS, TLIM		

The Go Home (HOM) command instructs the controller to search for the home position in the direction, and on the axes, specified by the command. If an end-of-travel limit is activated while searching for the home limit, the controller will reverse direction and search for home in the opposite direction. However, if a second end-of-travel limit is encountered, after the change of direction, the homing operation will be aborted.

The status of the homing operation is provided by bit 5 of each axis status register (refer to the TAS or AS command). *When the homing operation is successfully completed, the absolute position register is set to zero (equivalent to PSETØ).*

NOTE

Pause and resume functions are not recommended during the homing operation. A Pause command or input will pause the homing motion; however, when the subsequent Resume command or input occurs, motion will resume at the beginning of the homing motion sequence.

The homing operation has several parameters that determine the homing algorithm:

- Home acceleration (HOMA and HOMAA)
- Home deceleration (HOMAD and HOMADA)
- Home velocity (HOMV)
- Final home velocity (HOMVF)
- Home reference edge (HOMEDG)
- Backup to home (HOMBAC)
- Final home direction (HOMDF)
- Active state of home input (LIMLVL)
- Home to encoder Z-channel (HOMZ)

For more information on homing refer to the *Homing* section of the *Programmer's Guide*.

Example:

```
SCALE1          ; Enable scaling
SCLA25000,25000,1,1 ; Set accel. scaling: axes 1 & 2 = 25000 steps/unit/unit;
                  ; axes 3 & 4 = 1 step/unit/unit
SCLV25000,25000,1,1 ; Set vel. scaling: axes 1 & 2 = 25000 steps/unit;
                  ; axes 3 & 4 = 1 step/unit
@SCLD1          ; Set distance scaling factor for all axes to 1 step/unit
DEL Homrdy      ; Delete program called Homrdy
DEF Homrdy      ; Begin definition of program called Homrdy
@MA0            ; Incremental index mode for all axes
@MC0            ; Preset index mode for all axes
HOMA10,12,1,2   ; Set home acceleration to 10, 12, 1, & 2 units/sec/sec for
                  ; axes 1, 2, 3 & 4
@HOMAD20        ; Set home deceleration to 20 units/sec/sec for all axes
HOMBAC1100      ; Enable backup to home switch on axes 1 and 2 only
HOMEDG0011      ; Axes 1 & 2 stop on the positive-direction edge of the home
                  ; switch, axes 3 and 4 are to stop on negative-direction side
@HOMDF0         ; Set final home direction to positive on all axes.
@HOMZ0          ; Disable homing to encoder Z-channel on all axes
LIMLVLxx0xx0xx0 ; Set home active level to low on axes 1-4
HOMV1,1,1,2     ; Set home velocity to 1, 1, 1, and 2 units/sec for
                  ; axes 1, 2, 3 & 4
@HOMVF.1        ; Sets home final velocity to 0.1 units/sec for all axes
HOM01XX         ; Execute go home in positive-direction on axis 1,
                  ; negative-direction on axis 2. Do not home on axes 3 and 4.
END              ; End definition of Homrdy
```

HOMA Home Acceleration

Type	Homing	Product	Rev
Syntax	<!><@><a>HOMA<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>	6K	5.0
Units	r = units/sec/sec		
Range	0.00001 - 39,999,998 (depending on the scaling factor)		
Default	10.0000		
Response	HOMA: *HOMA10.0000,10.0000,10.0000,10.0000 ... 1HOMA: *1HOMA10.0000		
See Also	HOM, HOMAD, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ, [LIM], LIMEN, LIMLVL, SCALE, SCLA		

The Home Acceleration (HOMA) command specifies the acceleration rate to be used upon executing the next go home (HOM) command.

UNITS OF MEASURE and SCALING: refer to page 16.

The homing acceleration remains set until you change it with a subsequent homing acceleration command. Homing accelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid homing acceleration is entered the previous homing acceleration value is retained.

If the home deceleration (HOMAD) command has not been entered, the home acceleration (HOMA) command will set the home deceleration rate. Once the home deceleration (HOMAD) command has been entered, the home acceleration (HOMA) command no longer affects home deceleration.

Example: Refer to the go home (HOM) command example.

HOMAA Homing Average Acceleration

Type	Motion (S-Curve)	Product	Rev
Syntax	<!><@><a>HOMAA<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>	6K	5.0
Units	r = units/sec/sec		
Range	0.00001 - 39,999,998 (depending on the scaling factor)		
Default	10.00 (trapezoidal profiling is default, where HOMAA tracks HOMA)		
Response	HOMAA: *HOMAA10.0000,10.0000,10.0000,10.0000 ... 1HOMAA: *1HOMAA10.0000		
See Also	A, AD, ADA, HOM, HOMA, HOMAD, HOMADA, HOMBAC, SCALE, SCLA		

The Homing Average Acceleration (HOMAA) command allows you to specify the average acceleration for an S-curve homing profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*. Refer to page 13 for details on S-curve profiling.

Scaling (SCLA) affects HOMAA the same as it does for HOMA. Refer to page 16 for details on scaling.

Example:

```
SCALE0           ; Disable scaling
DEL proge        ; Delete program called proge
DEF proge        ; Begin definition of program called proge
@MA0             ; Select incremental positioning mode
HOMA10,10        ; Set homing max. accel to 10 rev/sec/sec (axes 1 and 2)
HOMAA5,10        ; Set homing avg. accel to 5 rev/sec/sec on axis 1,
                  ; and 10 rev/sec/sec on axis 2
HOMAD10,10       ; Set homing max. decel to 10 rev/sec/sec (axes 1 and 2)
HOMADA5,10       ; Set homing avg. decel to 5 rev/sec/sec on axis 1,
                  ; and 10 rev/sec/sec on axis 2
HOM11XX          ; Execute negative-direction homing moves on axes 1 and 2
; Axis 1 executes a pure S-curve; axis 2 executes a trapezoidal profile.
END              ; End definition of program
```

HOMAD Home Deceleration

Type	Homing	Product	Rev
Syntax	<!><@><a>HOMAD<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>	6K	5.0
Units	r = units/sec/sec		
Range	0.00001 - 39,999,998 (depending on the scaling factor)		
Default	10.0000 (HOMAD tracks HOMA)		
Response	HOMAD: *HOMAD10.0000,10.0000,10.0000,10.0000 ... 1HOMAD: *1HOMAD10.0000		
See Also	HOM, HOMA, HOMAA, HOMADA, HOMBAC, HOMEDG, HOMDF, HOMV, HOMVF, HOMZ, [LIM], LIMEN, LIMLVL, SCALE, SCLA		

The Home Deceleration (HOMAD) command specifies the deceleration rate to be used upon executing the next go home (HOM) command.

UNITS OF MEASURE and SCALING: refer to page 16.
--

The home deceleration remains set until you change it with a subsequent home deceleration command. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid deceleration is entered the previous deceleration value is retained.

If the home deceleration (HOMAD) command has not been entered, the home acceleration (HOMA) command will set the deceleration rate. Once the home deceleration (HOMAD) command has been entered, the home acceleration (HOMA) command no longer affects home deceleration. If the HOMAD command is set to zero (HOMADØ), then the homing deceleration will once again track whatever the HOMA command is set to.

Example: Refer to the go home (HOM) command example.

HOMADA Homing Average Deceleration

Type	Motion (S-Curve)	Product	Rev
Syntax	<!><@><a>HOMADA<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>	6K	5.0
Units	r = units/sec/sec		
Range	0.00001 - 39,999,998 (depending on the scaling factor)		
Default	10.00 (HOMADA tracks HOMAA)		
Response	HOMADA: *HOMADA10.0000,10.0000,10.0000,10.0000 ... 1HOMADA: *1HOMADA10.0000		
See Also	A, AD, HOM, HOMA, HOMAA, HOMAD, SCALE, SCLA		

The Homing Average Deceleration (HOMADA) command allows you to specify the average deceleration for an S-curve homing profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*. Refer to page 13 for details on S-curve profiling.

Scaling (SCLA) affects HOMADA the same as it does for HOMAD. Refer to page 16 for details on scaling.

Example:

```
SCALE0           ; Disable scaling
DEL proge        ; Delete program called proge
DEF proge        ; Begin definition of program called proge
@MA0            ; Select incremental positioning mode
HOMA10,10       ; Set homing max. accel to 10 rev/sec/sec (axes 1 and 2)
HOMAA5,10       ; Set homing avg. accel to 5 rev/sec/sec on axis 1,
                ; and 10 rev/sec/sec on axis 2
HOMAD10,10      ; Set homing max. decel to 10 rev/sec/sec (axes 1 and 2)
HOMADA5,10      ; Set homing avg. decel to 5 rev/sec/sec on axis 1,
                ; and 10 rev/sec/sec on axis 2
HOM11XX         ; Execute negative-direction homing moves on axes 1 and 2.
                ; Axis 1 executes a pure S-curve.
                ; Axis 2 executes a trapezoidal profile.
END             ; End definition of program
```

HOMBAC Home Backup Enable

Type	Homing	Product	Rev
Syntax	<!><@><a>HOMBAC	6K	5.0
Units	n/a		
Range	b = 0 (disable), 1 (enable), or X (don't change)		
Default	0		
Response	HOMBAC: *HOMBAC0000_0000 1HOMBAC: *1HOMBAC0		
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ, [LIM], LIMEN, LIMLVL		

The Home Backup Enable (HOMBAC) command enables or disables the backup to home switch function. When this function is enabled, the motor will decelerate to a stop after encountering the active edge of the home region, and then move the motor in the opposite direction at the home final velocity (HOMVF) until the active edge of the home region is encountered. This motion will occur regardless of whether or not the home input is active at the end of the deceleration of the initial go home move.

Example: Refer to the go home (HOM) command example.

HOMDF Home Final Direction

Type	Homing	Product	Rev
Syntax	<!><@><a>HOMDF	6K	5.0
Units	n/a		
Range	b = 0 (positive-direction), 1 (negative-direction), or X (don't change)		
Default	0		
Response	HOMDF: *HOMDF0000_0000 1HOMDF: *1HOMDF0		
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMEDG, HOMV, HOMVF, HOMZ, [LIM], LIMEN, LIMLVL		

The Home Final Direction (HOMDF) command specifies the direction the 6K Series product is to be traveling when the home algorithm does its final approach. This command is operational when backup to home (HOMBAC) is enabled, or when homing to an encoder Z channel (HOMZ).

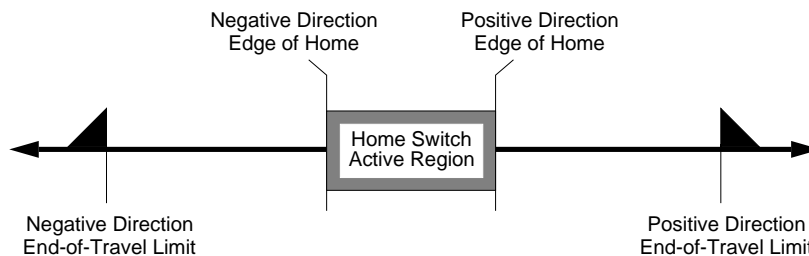
Example: Refer to the go home (HOM) command example.

HOMEDG Home Reference Edge

Type	Homing	Product	Rev
Syntax	<!><@><a>HOMEDG	6K	5.0
Units	n/a		
Range	b = 0 (positive-direction edge), 1 (negative-direction edge), or X (don't change)		
Default	0		
Response	HOMEDG: *HOMEDG0000_0000 !HOMEDG: *!HOMEDG0		
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMV, HOMVF, HOMZ, [LIM], LIMEN, LIMLVL,		

The Home Reference Edge (HOMEDG) command specifies which edge of the home switch the homing operation will consider as its final destination.

As illustrated below, the positive-direction edge of the home switch is defined as the first switch transition seen by the controller when traveling off of the positive-direction end-of-travel limit in the negative direction. The negative-direction edge of the home switch is defined as the first switch transition seen by the indexer when traveling off of the negative-direction end-of-travel limit in the positive-direction. This command is operational when backup to home (HOMBAC) is enabled.



Example: Refer to the go home (HOM) command example.

HOMV Home Velocity

Type	Homing	Product	Rev
Syntax	<!><@><a>HOMV<r>, <r>, <r>, <r>, <r>, <r>, <r>, <r>	6K	5.0
Units	r = units/sec (scalable with SCLV)		
Range	Stepper Axes: 0.00000-2,048,000 (max. depends on SCLV & PULSE) Servo Axes: 0.00000-6,500,000 (max. depends on SCLV)		
Default	1.0000		
Response	HOMV: *HOMV1.0000,1.0000,1.0000,1.0000 ... !HOMV: *!HOMV1.0000		
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMEDG, HOMVF, HOMZ, [LIM], LIMEN, LIMLVL, PULSE, SCALE, SCLV		

The Home Velocity (HOMV) command specifies the velocity to use when the home algorithm begins its initial go home (HOM) move. The velocity remains set until you change it with a subsequent home velocity command. Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered the previous velocity value is retained.

UNITS OF MEASURE and SCALING: refer to page 16.

Example: Refer to the go home (HOM) command example.

HOMVF Home Final Velocity

Type	Homing	Product	Rev
Syntax	<!><@><a>HOMVF<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>	6K	5.0
Units	r = units/sec (scalable with SCLV)		
Range	Stepper Axes: 0.00000-2,048,000 (max. depends on SCLV & PULSE) Servo Axes: 0.00000-6,500,000 (max. depends on SCLV)		
Default	0.1000		
Response	HOMVF: *HOMVF0.1000,0.1000,0.1000,0.1000 ... lHOMVF: *lHOMVF0.1000		
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMZ, [LIM], LIMEN, LIMLVL, PULSE, SCALE, SCLV		

The Home Final Velocity (HOMVF) command specifies the velocity to use when the home algorithm does its final approach. This command is only operational when backup to home (HOMBAC) is enabled, or when homing to an encoder Z channel (HOMZ).

The velocity remains set until you change it with a subsequent home final velocity command. Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered, the previous velocity value is retained.

UNITS OF MEASURE and SCALING: refer to page 16.
--

Example: Refer to the go home (HOM) command example.

HOMZ Home to Encoder Z-channel Enable

Type	Homing	Product	Rev
Syntax	<!><@><a>HOMZ	6K	5.0
Units	n/a		
Range	b = 0 (disable), 1 (enable), or X (don't change)		
Default	0		
Response	HOMZ: *HOMZ0000_0000 lHOMZ: *lHOMZ0		
See Also	[ASX], HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, [LIM], LIMEN, LIMLVL, TASX		

This command enables homing to an encoder z-channel after the initial home input has gone active. **NOTE:** The home limit input is required to go active prior to homing to the Z channel. The state of the Z-channel is reported with bit 6 of the ASX and TASX register.

Example: Refer to the go home (HOM) command example.

IF()

IF Statement

Type	Program Flow Control or Conditional Branching	Product	Rev
Syntax	<!>IF(expression)	6K	5.0
Units	n/a		
Range	Up to 80 characters (including parentheses)		
Default	n/a		
Response	n/a		
See Also	ELSE, NIF		

This command is used in conjunction with the ELSE and NIF commands to provide conditional branching. If the expression contained within the parenthesis of the IF command evaluates true, then the commands between the IF and the NIF are executed. If the expression evaluates false, the commands between the IF and the NIF are ignored, and command processing continues with the first command following the NIF.

When the ELSE command is used in conjunction with the IF command, true IF evaluations cause the commands between the IF and ELSE commands to be executed, the commands after the ELSE until the NIF are ignored. False IF evaluations cause commands between the ELSE and the NIF to be executed, with commands between the IF and the ELSE ignored. The ELSE command is optional and does not have to be included in the IF statement.

The IF() . . ELSE . . NIF structure can be nested up to 16 levels deep.

NOTE: Be careful about performing a GOTO between IF and NIF. Branching to a different location within the same program will cause the next IF statement encountered to be nested within the previous IF statement, unless an NIF command has already been encountered.

IF statement programming order: IF(expression)...commands...NIF
or
IF(expression)...commands...ELSE...commands...NIF

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the IF expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single IF expression. The limiting factor for the IF expression is the command length. **The total character count for the IF command and expression cannot exceed 80 characters.** (e.g., If you add up the letters in the IF command and the letters within the () expression, including the parenthesis and excluding each space, this count must be less than or equal to 80.)

All assignment operators (A, AD, AS, ASX, D, ER, IN, LIM, MOV, OUT, PC, PCE, PCM, PE, PER, PMAS, SEG, SS, TIM, US, V, VEL, VELA, etc.) can be used within the IF expression.

Multiple parentheses may not be used within the IF command.

Example:

```
IF(IN=b1X0 AND VAR1=1)          ; If onboard input 1 is ON, input 3 is OFF, and
                                ; variable 1 equals 1, then the IF statement evaluates
                                ; true, so commands between this statement and NIF
                                ; are executed
                                ; Transfer revision level
    TREV                          ; End IF statement
    NIF
IF(1A<5000 AND 2PC>50000)      ; If the acceleration of axis 1 is less than 5000, and
                                ; the commanded position of axis 2 is greater than
                                ; 50000, then do the IF statement. Note: The
                                ; acceleration value used is programmed acceleration,
                                ; not actual.
                                ; Increment variable 1
    VAR1=VAR1+1                  ; End if statement
    NIF
IF(4VEL<123 OR 4VEL>156)      ; If the current velocity of axis 4 is less than 123
                                ; or if it is greater than 156, then do the commands
                                ; following the IF statement
    WRITE"Something's Wrong\13" ; Put message Something's Wrong<cr> in output buffer
    NIF
IF(OUT=b110X1 AND VAR1<=13) ; If onboard outputs 1, 2 and 5 are ON, output 3 is
                                ; off and variable 1 is less than or equal to 13,
                                ; then set variable 1 equal to variable 1 plus 1,
                                ; else set variable 1 equal to variable 1 minus 1
    VAR1=VAR1+1
    ELSE
    VAR1=VAR1-1
    NIF                          ; End IF statement
```

[IN]**Input Status**

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	GOWHEN, INFNC, [LIM], ONIN, TIN, VARB		

Use the IN operator is used to assign the input value to a binary variable (VARB), or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, 0, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers 0 through 9.

Syntax: VARBn=IN where “n” is the binary variable number, or IN can be used in an expression such as IF(IN=b1101), or IF(IN=h7F). To assign only one input value to a binary variable, instead of all the inputs, the bit select (.) operator can be used. For example, VARB1=2IN.10 assigns the binary state of input 10 (2nd pin on SIM 2) on I/O brick 2 to binary variable 1.

The number of inputs available for assignment or comparison varies from one 6K Series product to another; to ascertain the input bit assignments for your 6K Series product refer to page 6. The function of the inputs is established with the INFNC command (although the IN operator looks at any trigger or external digital input, regardless of its assigned function from the INFNC command).

Example:

```
VARB1=3IN          ; Input status on I/O brick 3 assigned to binary variable 1
VARB2=2IN.12       ; Input bit 12 on I/O brick 2 assigned to binary variable 2
VARB2              ; Response if bit 12 is set to 1:
                   ; *VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
IF(1IN=b111011X11) ; If the input status contains 1's for inputs 1,2,3,5,6,8,& 9,
                   ; and 0 for input 4 on I/O brick 1, do the commands
                   ; following the IF statement
TREV              ; Transfer revision level
NIF               ; End IF statement
IF(2IN=hEF00)     ; If the input status contains 1's for I/O brick 2's inputs
                   ; 1,2,3,5,6,7,& 8, and 0's for every other input, do the
                   ; commands following the IF statement
TREV              ; Transfer revision level
NIF               ; End IF statement
```

INDEB Input Debounce Time

Type	Input	Product	Rev
Syntax	<! > INDEB<i>	6K	5.0
Units	i = time in milliseconds (ms)		
Range	i = 2-250		
Default	4		
Response	INDEB: *0INDEB4 *1INDEB4		
See Also	INFNC, LIMFNC, RE, REG, TIN, TLIM, TRGFN, TRGLOT		

The INDEB command governs the debounce time for all of the inputs on the specified I/O brick (all trigger inputs, found on the “TRIGGERS/OUTPUTS” connectors, are collectively considered I/O brick 0). The debounce is the period of time that the input must be held in a certain state before the controller recognizes it. This directly affects the rate at which the inputs can change state and be recognized. The default setting is 4 ms.

Exception for Trigger Inputs: For trigger inputs that are assigned the “Trigger Interrupt” function (INFNCi-H), the debounce is instead governed by the TRGLOT setting. The TRGLOT setting applies to all trigger inputs defined as “Trigger Interrupt” inputs. The TRGLOT debounce time is the time required between a trigger's initial active transition and its secondary active transition. This allows rapid recognition of a trigger, but prevents subsequent bouncing of the input from causing a false position capture. The default setting is 24 ms.

Limit Inputs. The limit inputs found on the “LIMITS/HOME” connectors are not normally debounced; however, if a limit is assigned a different function with the LIMFNC command (other than LIMFNCi-R, LIMFNCi-S, or LIMFNCi-T), the input is debounced using the INDEB setting for the on-board trigger inputs (I/O brick 0). If a general-purpose input or trigger input is assigned a limit input function (INFNCi-R, INFNCi-S, or INFNCi-T), the input will not be debounced.

Example:

```
INDEB6                   ; Assign all onboard trigger a debounce time of 6 ms
2INDEB10                ; Assign inputs on I/O brick 2 a debounce time of 10 ms
1INDEB12                ; Assign inputs on I/O brick 1 a debounce time of 12 ms
```

INDUSE Enable/Disable User Status

Type	Controller Configuration	Product	Rev
Syntax	<! > INDUSE	6K	5.0
Units	n/a		
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	INDUSE: *INDUSE0		
See Also	INDUST, ONUS, TUS, [US]		

The Enable/Disable User Status (INDUSE) command enables the INDUST command updates. When this command is not enabled, the user status bits (INDUST) can be defined; however, they will not be updated in the US or the TUS commands until INDUSE is enabled.

Example:

```
INDUSE1                ; Enable user status
```

INDUST User Status Definition

Type	Controller Configuration	Product	Rev
Syntax	<! >INDUST<i><-<i><c>>	6K	5.0
Units	See description below		
Range	1st i = 1 - 16; 2nd i = 1 - 32; c = A through S		
Default	See description below		
Response	INDUST: *INDUST1-1A AXIS 1 STATUS - STATUS OFF (...repeated for all 16 user status bits...) *INDUST16-4D AXIS 4 STATUS - STATUS OFF INDUST1: *INDUST1-1A AXIS 1 STATUS - STATUS OFF		
See Also	[AS], [ASX], [IN], INDUSE, ONUS, [SS], TAS, TASX, TIN, TSS, TUS, [US]		

The User Status Definition (INDUST) command establishes the user status bit function. Each bit can correspond to an axis status bit, a system status bit, an input, an interrupt bit, or an extended axis status bit. The default for each user status bit is as follows:

Default for the 6K product (first two AS status bits for each axis):

- Bits 1-2 = first 2 bits of axis status (AS) for axis 1
- Bits 3-4 = first 2 bits of axis status (AS) for axis 2
- Bits 5-6 = first 2 bits of axis status (AS) for axis 3
- Bits 7-8 = first 2 bits of axis status (AS) for axis 4
- Bits 9-10 = first 2 bits of axis status (AS) for axis 5
- Bits 11-12 = first 2 bits of axis status (AS) for axis 6
- Bits 13-14 = first 2 bits of axis status (AS) for axis 7
- Bits 15-16 = first 2 bits of axis status (AS) for axis 8

The purpose of this command is to allow the user to create his or her own meaningful status word. It allows the user to place certain status information in the order they prefer.

The syntax INDUST<i><-<i><c>> is described as follows:

- First <i> corresponds to the user status bit being defined (16 maximum).
- Second <i> corresponds to the bit of the axis status (AS), the system status (SS), the input status (IN), or the extended axis status (ASX).
- The <c> defines what status to use:

<c>	Value	Function	<c>	Value	Function
	A	Use axis status (AS) for axis 1		K	RESERVED
	B	Use axis status (AS) for axis 2		L	Use extended axis status (ASX) for axis 1
	C	Use axis status (AS) for axis 3		M	Use extended axis status (ASX) for axis 2
	D	Use axis status (AS) for axis 4		N	Use extended axis status (ASX) for axis 3
	E	Use axis status (AS) for axis 5		O	Use extended axis status (ASX) for axis 4
	F	Use axis status (AS) for axis 6		P	Use extended axis status (ASX) for axis 5
	G	Use axis status (AS) for axis 7		Q	Use extended axis status (ASX) for axis 6
	H	Use axis status (AS) for axis 8		R	Use extended axis status (ASX) for axis 7
	I	Use system status (SS) *		S	Use extended axis status (ASX) for axis 8
	J	Use input status (IN) **			

* If you are using multitasking, the "I" value requires you to prefix the INDUST command with the task identifier (e.g., 2%INDUST6-2I assigns system status bit 2 for task 2 to user status bit 6). If no task prefix is given, the system status for task 1 is used by default.

** The "J" value requires you to prefix the INDUST command with the I/O brick identifier (e.g., 2INDUST14-4J assigns the status of I/O point on I/O brick 2 to user status bit 14). If no brick prefix is given, the onboard trigger inputs are referenced by default. Refer to page 6 to fully understand the I/O bit patterns and use of the brick identifier.

Example

```
INDUSE1          ; Enable user status
INDUST1-5A      ; User status bit 1 defined as axis 1 status bit 5
INDUST2-3F      ; User status bit 2 defined as axis 6 status bit 3
3INDUST3-5J     ; User status bit 3 defined as input 5 on I/O brick 3
2%INDUST16-2I   ; User status bit 16 defined as system status bit 2 for task 2
```

INEN

Input Enable

Type	Input or Program Debug Tool	Product	Rev
Syntax	<!>INEN<d><d>...<d> (one <d> for each input)	6K	5.0
Units	n/a		
Range	d = 0 (disable, leave off), 1 (disable, leave on), E (enable), or X (don't change)		
Default	E		
Response	INEN: *INENEEEE_EEEE_EEEE_EEEE_E LINEN: *LINENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE LINEN.3 *E		
See Also	DRFEN, ERROR, [IN], INFNC, INLVL, INPLC, INSTW, LH, LIMEN, TIN, TIO, TSTAT		

The **INEN** command allows you to simulate the activation of specific trigger or external digital inputs (without actually wiring the inputs to the controller) by disabling them and setting them to a specific level (ON or OFF).

The default **INEN** condition is enabled (E), requiring external wiring to exercise the input's respective **INFNC** function.

Using Inputs on Expansion I/O Bricks: If the I/O brick is disconnected or if it loses power, the controller will perform a kill (all tasks) and set error bit #18 (see **ERROR**). The controller will remember the brick configuration (volatile memory) in effect at the time the disconnection occurred. When you reconnect the I/O brick, the controller checks to see if anything changed (SIM by SIM) from the state when it was disconnected. If an existing SIM slot is changed (different SIM, vacant SIM slot, or jumper setting), the controller will set the SIM to factory default **INEN** and **OUTLVL** settings. If a new SIM is installed where there was none before, the new SIM is auto-configured to factory defaults.

Example: **INEN1** disables trigger input A1 but leaves it in the ON state (the **TIN** command will show trigger input 1A as active). **INEN0** disables trigger input A1 but leaves it in the OFF (inactive) state. To re-enable trigger input 1A, issue the **INENE** command.

INEN has no effect on ...

- trigger inputs when they are configured as "trigger interrupt" inputs with the **INFNCi-H** command. This includes position capture and registration functions.
 - trigger or external digital inputs configured as "end-of-travel limit" inputs with the **INFNCi-aR** or **INFNCi-aS** commands. Instead, use the **LH** command.
 - limit inputs found on your product's "LIMITS/HOME" connector(s).
-

Input bit assignments for the **INEN** command vary by product and external I/O brick configuration. The input bit patterns for onboard and external I/O bricks are explained on page 6 of this document.

Example:

```
DEF tester           ; Begin definition of program tester
WHILE(IN=b11X10)    ; While onboard inputs 1, 2, and 4 are active, and input 5 is not
                    ; active, execute the statements between the WHILE & NWHILE
GO1100              ; Initiate motion on axes 1 and 2
NWHILE              ; End WHILE statement
END                 ; End definition of program tester
INEN11X10           ; Disable onboard inputs 1,2,4, & 5, and set inputs 1, 2 & 4 in
                    ; the active state, and input 5 in the inactive state
RUNtester           ; Initiate program tester
!INEN00000          ; Disable onboard inputs 1,2,3,4, & 5, and leave them in the
                    ; inactive state
INENeeeeee         ; Re-enable inputs 1 through 5
```

INFNC

Input Function

Type	Input	Product	Rev
Syntax	<! >INFNC<i>-<<a>c>	6K	5.0
Units	i = input #, a = axis #, c = function identifier letter		
Range	i = 1-32 (I/O brick dependent – see page 6); a = 1-8 (product dependent); c = A-T		
Default	A		
Response	INFNC: (input function and status of onboard inputs) 1INFNC: (input function and status of I/O brick 1 inputs) 1INFNC1: *1INFNC1-A NO FUNCTION - STATUS OFF		
See Also	COMEXR, COMEXS, ENCCNT, [ER], ERROR, [IN], INDEB, INEN, INLVL, INPLC, INSELP, INSTW, INTHW, JOY, JOYAXH, JOYAXL, JOYVH, JOYVL, K, KDRIVE, LH, LIMFNC, PSET, [SS], TER, TIN, TIO, TRGFN, TRGLOT, [TRIG], TSS, TSTAT, TTRIG		

The Input Function (INFNC) command defines the function of each individual input, where *i* is the input bit number, *a* is an axis number if required, or the program number for the case of input function P, and *c* is the function. All function definitions given below will specify whether an axis number is required. A limit of 32 inputs may be assigned INFNC functions; this excludes functions A (“general-purpose”) and H (“trigger interrupt”).

Input Debounce. Using the Input Debounce Time (INDEB) command, you can change the input debounce time for all of the inputs on the specified I/O brick (all trigger inputs, found on the “TRIGGERS/OUTPUTS” connectors, are collectively considered I/O brick 0). The debounce is the period of time that the input must be held in a certain state before the controller recognizes it. This directly affects the rate at which the inputs can change state and be recognized. Trigger inputs that are assigned the “Trigger Interrupt” function (INFNCi-H), are instead debounced by the TRGLOT value. Inputs defined as limit inputs (INFNCi-R, INFNCi-S, or INFNCi-T), will not be debounced.

Input bit assignments vary by product. The input bit patterns for onboard and external I/O bricks are explained on page 6 of this document.

Input Scan Rate. The programmable inputs are scanned once per *system update* (2 milliseconds).

Multitasking. If the INFNC command does not include the task identifier (%) prefix, the function affects the task that executes the INFNC command. The functions that may be directed to a task with % are: C, D (without an axis specified), E, F, and P (e.g., 2%INFNC3-F assigns onboard input 3 as a user fault input for task 2). Multiple tasks may share the same input, but the input may only be assigned one function.

Identifier Function Description

- A **No special function** (general-purpose input). Normal input, used with the IN assignment
- B **BCD Program Select.** BCD input assignment to programs, lowest numbered input is least significant bit (LSB). BCD values for inputs are as follows:

	BCD Value
Least Significant Bit Value	1
.	2
.	4
.	8
.	10
.	20
.	40
.	80
Most Significant Bit Value	100

Note: If fewer inputs than shown above are defined to be Program Select Inputs, then the highest input number defined as a Program Select Input is the most significant bit.

An input defined as a BCD Program Select Input will not function until the INSELP command has been enabled.

Identifier Function Description

C Kill. Kills motion on all axes and halts all command processing (refer to `K` and `KDRIVE` command descriptions for further details on the *kill* function). This is an edge detection function and is not intended to inhibit motion. To inhibit motion, use the Pause/Resume function (`INFNCi-E`). When enabled with the `ERROR` command, bit #6 of the `TER` and `ER` commands will report the kill status.

<a>D Stop. Stops motion. Axis number is optional; if no axis number is specified, motion is stopped on all axes. If `COMEXS` is set to zero (`COMEXS0`), program execution will be terminated. If `COMEXS` is set to 1 (`COMEXS1`), command processing will continue. With `COMEXS` set to 2 (`COMEXS2`), program execution is terminated, but the `INSELP` value is retained. Motion deceleration during the stop is controlled by the `AD` & `ADA` commands. If error bit #8 is enabled (e.g., `ERROR.8-1`), activating a Stop input will set the error bit and cause a branch to the `ERRORP` program.

E Pause/Continue. If `COMEXR` is disabled (`COMEXR0`), then only command execution pauses, not motion. With `COMEXR` enabled (`COMEXR1`), both command and motion execution are paused. After motion stops, you can release the input or issue a continue (`!C`) command to resume command processing (and motion of in `COMEXR1` mode).

F User Fault. Refer to the `ERROR` command. If error bit #7 is enabled (e.g., `ERROR.7-1`), activating a User Fault input will set the error bit and cause a branch to the `ERRORP` program. **CAUTION:** Activating the user fault input sends an `!K` command to the controller, “killing” motion on all axes (refer to the `K` command description for ramifications).

G Reserved

H Trigger Interrupt - This function can only be assigned to the onboard trigger inputs. A “Trigger Interrupt” input can be used for these purposes:

- **Position Capture.** Each axis has two dedicated trigger inputs, referred to as “TRIG-nA” and “TRIG-nB” (n = number of the axis). These trigger inputs are located on the 25-pin “TRIGGERS/OUTPUTS” connector. When either trigger input (TRIG-nA or TRIG-nB) for a particular axis is assigned the Trigger Interrupt function, activating the input performs a *hardware capture* of that axis' position. If the axis is used as a follower in Following, activating the trigger also performs an *interpolated capture* of the associated master axis position. An additional trigger, labeled “TRIG-M”, may be used to perform a hardware capture of the “MASTER ENCODER” (the encoder connected to the “Master Encoder” connector), as well as the position of all axes (encoder position on servo axes; commanded or encoder position for steppers, depending on the `ENCCNT` setting). To assign TRIG-M as a trigger interrupt input, use the `INFNC17-H` command.

When a Trigger Interrupt input is activated, the controller captures the relevant positions and stores them in registers that are available at the next system update (2 ms) through the use of these transfer and assignment/comparison commands:

Captured Information	Transfer	Assignment/Comparison	Offset *	Scale Factor **
Commanded position	TPCC	PCC	PSET	SCLD
Encoder position	TPCE	PCE	PSET or PESET	SCLD
Master encoder position	TPCME	PCME	PMESET	SCLMAS
Master cycle position	TPCMS	PCMS	PSET	SCLMAS

* Captured values are offset by any existing `PSET` or `PMESET` offset.

** If scaling is enabled, the captured position is scaled by `SCLD` or `SCLMAS`.

NOTES ABOUT POSITION CAPTURE:

- Hardware Capture: The encoder position is captured within ± 1 encoder count. The commanded position capture accuracy is ± 1 count.
- Interpolated Capture: There is a time delay of up to 50 μ s between activating the trigger interrupt input and capturing the position; therefore, the accuracy of the captured position is equal to 50 μ s multiplied by the velocity of the axis at the time the input was activated.
- Servo vs. Stepper. The nature of the axis position captured with a Trigger Interrupt input may be different, depending on whether the axis is configured for servo or stepper operation (`AXSDEF` command setting). For servo axes, both the commanded and encoder position for the axis are captured. Analog input feedback cannot be captured. For stepper axes, if the `ENCCNT` command is set to `ENCCNT0` (default condition), only the commanded position is captured. If `ENCCNT1` mode is enabled, only the encoder position is captured.

More about Trigger Interrupt function on next page ...

H (con't.) *Continued from previous page (Trigger Interrupt function):*

- **Registration.** (see RE description for details)
- **Special trigger functions** defined with the TRGFN command (see TRGFN for details).

NOTES ABOUT TRIGGER INTERRUPT INPUTS:

- When a trigger is assigned the "Trigger Interrupt" function, the debounce is governed by the TRGLOT command setting (default is 24 ms). The TRGLOT setting overrides the existing INDEB setting for only the trigger inputs that are assigned the "Trigger Interrupt" function.
 - When configured as Trigger Interrupts, the triggers cannot be affected by the input enable (INEN) command.
 - Trigger Interrupt Status: Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). The TTRIG/TRIG bits are cleared with the respective captured position is read (see table on previous page).
- I **Alarm Event** - Will cause the 6K controller to set an Alarm Event in the Communications Server over the Ethernet interface. You must first enable the Alarm checking bit for this input-driven alarm (INTHW. 23-1). For details on using alarms, refer to the *6K Series Programmer's Guide*.
- aJ **JOG positive-direction** - Will jog the axis specified in a positive-direction. The JOG command must be enabled for this function to work. **Axis number required.**
- aK **JOG negative-direction.** Will jog the axis specified in a negative-direction. The JOG command must be enabled for this function to work. **Axis number required.**
- aL **JOG Speed Select.** Selects the high or low velocity range while jogging. If the input is active, the high jog velocity range will be selected. Axis number is optional. If no axis number is designated, it defaults to all axes.
- M **Joystick Release.** Signals the controller to end joystick operation and resume program execution with the next statement in your program. When the input is open (high), the joystick mode is disabled (joystick mode can be enabled only if the input is closed, and only with the JOY command). When the input is closed (low), joystick mode can be enabled with the JOY command. The process of using Joystick mode is:
1. Assign the "Joystick Release" input function to a programmable input.
 2. At the appropriate place in the program, enable joystick control of motion (with the JOY command). (Joystick mode cannot be enabled unless the "Joystick Release" input is closed.) When the JOY command enables joystick mode for the affect axes, program execution stops on those axes (assuming the Continuous Command Execution Mode is disabled with the COMEXCØ command).
 3. Use the joystick to move the axes as required.
 4. When you are finished using the joystick, open the "Joystick Release" input to disable the joystick mode. This allows program execution to resume with the next statement after the initial JOY command that started the joystick mode.
- N **Joystick Axis Select.** Allows you to control two pairs of axes with one joystick. Use the JOYAXH and JOYAXL commands to assign analog inputs to control specific axes. Opening the Axis Select input (input is high) selects the JOYAXH configuration. Closing the Axis Select input (input is low) selects the JOYAXL configuration. NOTE: When this input is not connected, the JOYAXH configuration is always in effect.
- O **Joystick Velocity Select.** Allows you to select the velocity for joystick motion. The JOYVH and JOYVL commands establish two joystick velocities. Opening the Velocity Select input (input is high) selects the JOYVH configuration. Closing the Velocity Select input (input is low) selects the JOYVL configuration. The JOYVL velocity could be used to quickly move to a location, the JOYVH velocity could be used for low-speed accurate positioning. NOTE: When this input is not connected, joystick motion always uses the JOYVH velocity setting.

Identifier	Function Description
P	<p>Program Select. One to one correspondence for input vs. program number. The program number comes from the <code>TDIR</code> command. The number specified before the program name is the number to specify within this input definition. For example, in the <code>2INFNC1-3P</code> command, 3 is the program number. An input defined as a Program Select Input will not function until the <code>INSELP</code> command has been enabled.</p>
Q	<p>Program Security. Issuing the <code>INFNCi-Q</code> command enables the <i>Program Security</i> feature and assigns the <i>Program Access</i> function to the specified programmable input.</p> <p>The program security feature denies you access to the <code>DEF</code>, <code>DEL</code>, <code>ERASE</code>, <code>MEMORY</code>, <code>LIMFNC</code>, and <code>INFNC</code> commands until you activate the program access input. Being denied access to these commands effectively restricts altering the user memory allocation. If you try to use these commands when program security is active (program access input is not activated), you will receive the error message <code>*ACCESS DENIED</code>. <i>The <code>INFNCi-Q</code> command is not saved in battery-backed RAM, so you may want to put it in the start-up program (<code>STARTP</code>).</i></p> <p>For example, once you issue the <code>3INFNC12-Q</code> command, the input on the 4th pin on SIM2 (I/O point 12) of I/O brick 3 is assigned the program access function and access to the <code>DEF</code>, <code>DEL</code>, <code>ERASE</code>, <code>MEMORY</code>, <code>LIMFNC</code>, and <code>INFNC</code> commands will be denied until you activate the input.</p> <p>To regain access to these commands without the use of the program access input, you must issue the <code>INEN</code> command to disable the program security input, make the required user memory changes, and then issue the <code>INEN</code> command to re-enable the input. For example, if input 3 on brick 2 is assigned as the Program Security input, use <code>2INEN.3=1</code> to disable the input and leave it activated, make the necessary user memory changes, and then use <code>2INEN.3=E</code> to re-enable the input.</p>
aR	<p>End-of-Travel Limit, Positive Direction. This input function allows you to provide an end-of-travel limit input on your remove I/O brick. An axis number is required (e.g., <code>3INFNC1-4R</code> assigns the "Positive EOT limit" function to the 1st pin on the SIM1 (I/O point 1) on extended I/O brick #3, and makes it specific to axis 4). REMEMBER to reassign the corresponding dedicated hardware limit (on the "LIMITS/HOME" connector) to a function other than <code>LIMFNCi-aR</code>; otherwise, the <code>INFNCi-aR</code> input and the <code>LIMFNCi-aR</code> input will have the same function. Once an input is assigned a limit function, it is no longer debounced (<code>INDEB</code> has no effect), and it must be enabled/disabled with the <code>LH</code> command instead of the <code>INEN</code> command.</p>
aS	<p>End-of-Travel Limit, Negative Direction. This input function allows you to provide an end-of-travel limit input on your remove I/O brick. An axis number is required (e.g., <code>3INFNC2-4R</code> assigns the "Negative EOT limit" function to the 2nd pin on the SIM1 (I/O point 2) on extended I/O brick #3, and makes it specific to axis 4). REMEMBER to reassign the corresponding dedicated hardware limit (on the "LIMITS/HOME" connector) to a function other than <code>LIMFNCi-aS</code>; otherwise, the <code>INFNCi-aS</code> input and the <code>LIMFNCi-aS</code> input will have the same function. Once an input is assigned a limit function, it is no longer debounced (<code>INDEB</code> has no effect), and it must be enabled/disabled with the <code>LH</code> command instead of the <code>INEN</code> command.</p>
aT	<p>Home Limit. This input function allows you to provide a home limit input on your remote I/O brick. An axis number is required (e.g., <code>3INFNC3-4R</code> assigns the "Home limit" function to the 3rd pin on the SIM1 (I/O point 3) on extended I/O brick #3, and makes it specific to axis 4). REMEMBER to reassign the function of the home limit for the affected axis (e.g., given <code>3INFNC2-4T</code>, you must issue a <code>LIMFNC</code> command to assign a different function for the home input for axis 4). Once an input is assigned a limit function, it is no longer debounced (<code>INDEB</code> has no effect), and it must be enabled/disabled with the <code>LH</code> command instead of the <code>INEN</code> command.</p>

Example:

```
3INFNC1-D      ; Input at I/O point #1 on brick 3 is defined to be a
                ; stop input for all axes
```

INLVL Input Active Level

Type	Input	Product	Rev
Syntax	<!>INLVL...	6K	5.0
Units	n/a		
Range	b = 0 (active low), 1 (active high), or X (don't change)		
Default	0		
Response	INLVL: *INLVL0000_0000_0 (onboard trigger inputs) 1INLVL: *1INLVL0000_0000_0000_0000_0000_0000_0000_0000 1INLVL.3: *0 (active low)		
See Also	INEN, INFNC, INPLC, INSTW, LIMLVL		

The Input Active Level (INLVL) command defines the active state of all programmable inputs. To determine the input bit assignments for your 6K Series product, refer to page 6 of this document.

If the device driving the input is off (not sinking current), the input will show (using the TIN command) a zero (0) if the input has been defined as active low, and a one (1) if the input has been defined as active high. If the device driving the input is on (sinking current), the input will show a one (1) if the input has been defined as active low, and zero (0) if the input has been defined as active high. The default state is active low (INLVL0). The input schematics are provided in each 6K Series product's *Installation Guide*.

Example:

```
2INLVL0101 ; Set active level for these inputs on I/O brick 2:
            ; inputs 1 & 3 are active low, inputs 2 & 4 are active high.
```

[INO] Other Input Status

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[IN], [LIM], TINO, TINOF		

The Other Input Status (INO) command is used to assign an other input value to a binary variable, or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, 0, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers 0 through 9.

Syntax: VARBn=INO where n is the binary variable number
or [INO] can be used in an expression such as IF(INO=b1101), or IF(INO=h02)

There are 8 other inputs available for assignment or comparison. If it is desired to assign only one bit (one specific input) value to a binary variable, instead of all 8, use the bit select (.) operator. For example, VARB1=INO.6 assigns the status of the ENABLE input to binary variable 1.

Format for binary assignment: bbbbbbbb
 ^ ^
 Bit #1 Bit #8

Bit	Function	Location
1-5	RESERVED	
6	Enable input (1 = OK for motion)	ENABLE terminal
7-8	RESERVED	

Example:

```
VARB2=INO.6 ; ENABLE input status assigned to binary variable 2
VARB2 ; Response if bit 6 is set to 1: *VARB2=XXXX_X1XX
IF(INO.6=b1) ; If ENABLE input status is 1 (OK for motion), do the commands
            ; following the IF statement until the NIF statement
TREV ; Transfer revision level
NIF ; End if statement
```

INPLC Establish PLC Data Inputs

Type	Input	Product	Rev
Syntax	<!>INPLC<i>,<i-i>,<i>,<i>	6K	5.0
Units	See below		
Range	See below		
Default	1,0-0,0,0		
Response	INPLC1: *INPLC1,0-0,0,0 1INPLC1: *1		
See Also	INEN, INFNC, INLVL, INSTW, OUTPLC, [TW]		

The Establish PLC Data Inputs (INPLC) command, in combination with the OUTPLC command, configure the inputs and outputs to read data from a parallel I/O device such as a PLC (Programmable Logic Controller), or a passive thumbwheel module. The actual data transfer occurs with the TW command. Refer to the TW command for a description of the data transfer process.

The INPLC command has four fields (<i>,<i-i>,<i>,<i>):

Data Field	Description
Field 1: <i>	Set #: There are 4 possible INPLC sets (1-4). This field identifies which set to use.
Field 2: <i-i>	Input #s: Data is read into the 6K Series product through the programmable inputs. This field identifies the inputs to be used with the TW command. The first number is the first input, and the second number is the last input. The inputs must be consecutive. The number of inputs should be 8, because two BCD digits are read per data strobe. Refer to page 6 for help in identifying which input bits are available to place in this field.
Field 3: <i>	Sign Input #: This field identifies which input is designated to provide sign information. A zero specified in the command field specifies no sign information. An active signal on the input designated as the sign input indicates a negative data entry.
Field 4: <i>	Data Valid Input #: This field identifies which input is designated to be the data valid handshake input. A zero in this field indicates that there will be no data valid handshake input used. When an input is specified as a data valid, the input must be active in order for data to be read. If the input is not active, data will not be read until the signal becomes active.

To disable a specific PLC set, enter INPLCn,0-0,0,0 where n is the PLC set (1-4).

Example:

```
2INPLC2,1-8,9,10 ; Set INPLC set 2 as BCD digits on inputs 1-8 on I/O brick 2,
                  ; with input 9 as the sign bit, and input 10 as the data valid
1OUTPLC2,1-4,5,50 ; Set OUTPLC set 2 as output strobes on outputs 1-4 on I/O
                  ; brick 2, with output 5 as the output enable bit, and
                  ; strobe time of 50 milliseconds
A(TW6)           ; Read data into axis 1 acceleration using INPLC set 2 and
                  ; OUTPLC set 2 as the data configuration
```

INSELP Select Program Enable

Type	Input	Product	Rev
Syntax	<!>INSELP<i>, <i>	6K	5.0
Units	See below		
Range	1st i = 0, 1, or 2; 2nd i = 0 - 5000		
Default	0, 0		
Response	INSELP: *INSELP0, 0		
See Also	COMEXS, INEN, INFNC, INLVL, INPLC, INSTW, LIMFNC, [SS], TDIR, TSS		

The Select Program Enable (INSELP) command enables program selection by inputs. In addition, the command establishes the strobe time for the inputs, and if programs are selected on a one-to-one basis (INFNCi-iP or LIMFNCi-P) or on a BCD basis (INFNCi-B or LIMFNCi-B). When programs are selected on a one-to-one basis, each input defined with the INFNCi-iP or LIMFNCi-P command will run a specific program upon activation. When programs are selected by BCD values, each input defined by the INFNCi-B or LIMFNCi-B command will contribute to the BCD value, which corresponds to the program number. The program number is derived from the order in which the programs were defined (DEF). The first program defined is program #1, the second defined is program #2, etc. To verify which program number corresponds to each program, use the TDIR command. The number in front of the program name is the program number.

First i = Enable or disable function (∅ = Disable, 1 and 2 = Enable). Use INFNCi-B or LIMFNCi-B inputs if i = 1; use INFNCi-iP or LIMFNCi-P inputs if i = 2, to select program.

Second i = Strobe Time in milliseconds for inputs used to select program. The input must be active at the end of the strobe time for it to be recognized as a valid selection. The inputs are scanned once per *system update* (2 milliseconds).

The Kill (!K) command releases this mode, in addition to INSELP∅. The Stop (!S) command or an input defined as a stop input will also release this mode, as long as COMEXS has been disabled.

Example:

```
2INFNC1-1P        ; Input #1 on I/O brick 2 defined to select program #1
2INFNC2-2P        ; Input #2 on I/O brick 2 defined to select program #2
2INFNC3-7P        ; Input #3 on I/O brick 2 defined to select program #7
INSELP2,50        ; Enable continuous scan of inputs to select a program to run
```

INSTW Establish Thumbwheel Data Inputs

Type	Input	Product	Rev
Syntax	<!>INSTW<i>, <i-i>, <i>	6K	5.0
Units	See below		
Range	See below		
Default	1,0-0,0		
Response	INSTW1: *INSTW1,0-0,0 1INSTW1: *1INSTW1,0-0,0		
See Also	INEN, INFNC, INLVL, INPLC, OUTTW, [SS], TSS, [TW]		

The Establish Thumbwheel Data Inputs (INSTW) command, in combination with the OUTTW command, configure the inputs and outputs to read data from an active thumbwheel device. The actual data transfer occurs with the TW command. Refer to the TW command for a description of the data transfer process.

The INSTW command has three fields (<i>, <i-i>, <i>):

Data Field	Description
Field 1: <i>	Set #: There are 4 possible INSTW sets (1-4). This field identifies which set to use.
Field 2: <i-i>	Input #s: Data is read into the 6K Series product through the programmable inputs. This field identifies the inputs to be used with the TW command. The first number is the first input, and the second number is the last input. The inputs must be consecutive. The number of inputs should be compatible to the thumbwheel device. Refer to page 6 for help in identifying which input bits are available to place in this field.
Field 3: <i>	Sign Input #: This field identifies which input is designated to provide sign information. A zero specified in the command field specifies no sign information. An active signal on the input designated as the sign input indicates a negative data entry.

To disable a specific thumbwheel set, enter INSTWn,0-0,0 where n is the thumbwheel set (1-4).

Example:

```
3INSTW2,1-4,5      ; Set INSTW set 2 as BCD digits on inputs 1-4 on I/O brick 3,  
                   ; with input 5 as the sign bit  
2OUTTW2,1-3,4,50  ; Set OUTTW set 2 as output strobes on outputs 1-3 on I/O  
                   ; brick 2, with output 4 as the output enable bit, and  
                   ; strobe time of 50 milliseconds  
A(TW2)            ; Read data into axis 1 acceleration using INSTW set 2  
                   ; and OUTTW set 2 as the data configuration
```

INTSW Force an Alarm Event

Type	Alarm Event	Product	Rev
Syntax	<!>INTSW<i>	6K	5.0
Units	i = alarm event condition # (see list in INTHW)		
Range	i = 1-12		
Default	n/a		
Response	n/a		
See Also	INTHW		

This command forces a specific alarm event. The alarm events are available in the Communications Server (over the Ethernet interface), and a client application can read the Communications Server's "faster status" (alarm event) register to ascertain when certain conditions have occurred. 12 different software alarms are available (see table in INTHW command description). By forcing an alarm condition, you can customize the program to generate specific alarms at predefined places in your program.

The specific alarm event cannot be forced until the corresponding enable bit is set with the INTHW command. For example, before you can force alarm event bit #3 (INTSW3), you must first enable the 6K to check the state of alarm event bit #3 (INTHW.3-1).

The client application must determine the cause of the forced alarm event. This is accomplished by polling the Communication Server's "fast status" register for the alarm information. Once the register has been read for a client application, the alarm conditions are automatically cleared in the Communications Server. For more information on the alarms and using the fast status register, refer to the *Programmer's Guide*.

Example:

```
INTHW1          ; Enable alarm event bit #1
A20,20          ; Set acceleration to 20 units/sec/sec on axes 1 and 2
V2,2           ; Set velocity to 2 units/sec on axes 1 and 2
D25000,25000    ; Set move distance to 25000 units on axes 1 and 2
GO11           ; Initiate motion on axes 1 and 2
INTSW1         ; Force alarm event bit #1 as soon as the moves on axes 1 and
                ; 2 are finished.
; *****
; * Note: After the alarm occurs, it is the client application program's *
; * responsibility to examine the communication server's fast status *
; * register to determine the cause of the alarm. *
; *****
```