
UNTIL() Until Part of Repeat Statement

Type	Program Flow Control	Product	Rev
Syntax	<!>UNTIL(expression)	6K	5.0
Units	n/a		
Range	Up to 80 characters (including parentheses)		
Default	n/a		
Response	n/a		
See Also	JUMP, REPEAT		

The Until Part of Repeat Statement (UNTIL()) command, in conjunction with the REPEAT command, provide a means of conditional program flow. The REPEAT command marks the beginning of the conditional statement. The commands between the REPEAT and the UNTIL command are executed at least once. Upon reaching the UNTIL command, the expression contained within the UNTIL command is evaluated. If the expression is false, the program flow is redirected to the first command after the REPEAT command. If the expression is true, the first command after the UNTIL command is executed.

Up to 16 levels of REPEAT . . . UNTIL() commands may be nested.

NOTE: Be careful about performing a GOTO between REPEAT and UNTIL. Branching to a different location within the same program will cause the next REPEAT statement encountered to be nested within the previous REPEAT statement, unless an UNTIL command has already been encountered. The JUMP command should be used in this case.

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the UNTIL expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single UNTIL expression.

The limiting factor for the UNTIL expression is the command length. The total character count for the UNTIL command and expression cannot exceed 80 characters. For example, if you add all the letters in the UNTIL command and the letters within the () expression, including the parentheses and excluding the spaces, this count must be less than or equal to 80.

All assignment operators (A, AD, ANI, AS, D, DAC, DPTR, ER, IN, INO, LIM, MOV, OUT, PC, PCC, PCE, PCMS, PE, PER, SS, TIM, US, V, VEL, etc.) can be used within the UNTIL expression.

Example:

```
REPEAT          ; Beginning of REPEAT ... UNTIL( ) loop
GO1110         ; Initiate motion on axes 1, 2, and 3
IF(IN=b1X0)    ; IF condition: if onboard input 1 = 1, input 3 = 0
VAR1=VAR1+1    ; If condition comes true increment variable 1 by 1
ELSE           ; Else part of IF condition
TPE           ; If condition does not come true transfer position of
              ; all encoders
NIF           ; End IF statement
UNTIL(VAR1=12) ; Repeat loop until variable 1 = 12
```

[US]

User 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	INDUSE, INDUST, TUS		

The User Status (US) operator is used to assign the user status bits 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=US where “n” is the binary variable number,
or US can be used in an expression such as IF(US=b1101), or IF(US=h7)

All 16 bits of the user status word are defined with the INDUST command. Each bit can correspond to an axis status bit, a system status bit, or an input.

If it is desired to assign only one bit of the user status value to a binary variable, instead of all 16, the bit select (.) operator can be used. For example, VARB1=US.12 assigns user status bit 12 to binary variable 1.

Example:

```
VARB1=US           ; User status assigned to binary variable 1
VARB2=US.12       ; User status bit 12 assigned to binary variable 2
VARB2              ; Response, if bit 12 is set to 1, will be:
                  ; *VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
IF(US=b111011X11) ; If the user status contains 1's in bit locations
                  ; 1, 2, 3, 5, 6, 8, and 9, and a 0 in bit location 4,
                  ; do the IF statement
    TREV          ; Transfer revision level
ELSE              ; Else
    IF(US=h7F00)  ; If the user status contains 1's in bit locations
                  ; 1, 2, 3, 5, 6, 7, and 8, and 0's in every other bit
                  ; location, do the IF statement
    TSTAT         ; Transfer statistics
NIF               ; End of second if statement
NIF               ; End of first IF statement
```

V

Velocity

Type	Motion	Product	Rev
Syntax	<!><@><a>V<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>	6K	5.0
Units	r = units/sec		
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	V: *V1.0000,1.0000,1.0000,1.0000 ... 1V: *1V1.0000		
See Also	GO, MC, PULSE, SCALE, SCLV, TSTAT, TVEL, TVELA, [V], [VEL], [VELA], VF		

The Velocity (v) command defines the speed at which the motor will run when given a GO command. The motor will accelerate at a predefined acceleration (A) rate, before reaching the velocity (v) specified. The maximum velocity attainable is 2,048,000 units/sec (stepper axes) or 6,500,000 units/sec (servo axes).

The velocity remains set until you change it with a subsequent 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.

ON-THE-FLY CHANGES: While running in the continuous mode (MC1), you can change velocity *on the fly* (while motion is in progress) in two ways. One way is to send an immediate velocity command (!V) followed by an immediate go command (!GO). The other, and more common, way is to enable the continuous command execution mode (COMEXC1) and execute a buffered velocity command (V) followed by a buffered go command (GO).

Example:

```
SCALE1           ; Enable scaling
SCLA25000,25000,1,1 ; Set the acceleration scaling factor for axes 1 and 2 to
                  ; 25000 steps/unit/unit, axes 3 and 4 to 1 step/unit/unit
SCLV25000,25000,1,1 ; Set the velocity scaling factor for axes 1 and 2 to
                  ; 25000 steps/unit, axes 3 and 4 to 1 step/unit
SCLD1,1,1,1      ; Set the distance scaling factor for axes 1, 2, 3, and 4
                  ; to 1 step/unit
DEL proge        ; Delete program called proge
DEF proge        ; Begin definition of program called proge
MA0000           ; Incremental index mode for axes 1-4
MC0000           ; Preset index mode for axes 1-4
A10,12,1,2       ; Set the acceleration to 10, 12, 1, and 2 units/sec/sec
                  ; for axes 1, 2, 3 and 4
V1,1,1,2         ; Set the velocity to 1, 1, 1, and 2 units/sec for
                  ; axes 1, 2, 3 and 4
D100000,1000,10,100 ; Set the distance to 100000, 1000, 10, and 100 units
                  ; for axes 1, 2, 3 and 4
GO1100           ; Initiate motion on axes 1 and 2, 3 and 4 do not move
END              ; End definition of proge
```

[V] Velocity (Programmed) Assignment

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	GO, SCALE, SCLV, SSV, V, [VEL]		

The velocity assignment (V) operator is used to compare the programmed velocity value to another value or variable, or to assign the current programmed velocity to a variable.

Syntax: VARn=aV where “n” is the variable number, and “a” is the axis number,
or V can be used in an expression such as IF(1V<25)

When assigning the velocity value to a variable, an axis specifier must always precede the assignment (V) operator or it will default to axis 1 (e.g., VAR1=1V). When making a comparison to the programmed velocity, an axis specifier must also be used (e.g., IF(1V<20)). The (V) value used in any comparison, or in any assignment statement is the programmed (V) value. If the actual velocity information is required, refer to the VEL command.

UNITS OF MEASURE and SCALING: refer to page 16.

Example:

```
IF(2V<25)        ; If the programmed velocity on axis 2 is less than 25
                  ; units/sec, then do the statements between the IF and NIF
VAR1=2V*2        ; Variable 1 = programmed velocity of axis 2 times 2
V,(VAR1)         ; Set the velocity on axis 2 to the value of variable 1
NIF              ; End the IF statement
```

VAR Numeric Variable Assignment

Type	Variable	Product	Rev
Syntax	<!>VAR<i><=r>	6K	5.0
Units	i = variable number r = number or expression		
Range	i = 1-225 r = ±999,999,999.99999999		
Default	n/a		
Response	VAR1: *VAR1=+0.0		
See Also	DVAR, VARB, VARCLR, VARI, VARS, WRVAR		

Numeric variables can be used to store any real number value, with a range from -999,999,999.99999999 to +999,999,999.99999999. The information is assigned to the variable with the equal sign (e.g., VAR1=32.3).

All variables (numeric [VAR], integer [VARI], binary [VARB], and string [VARS]) are automatically stored in battery-backed RAM.

Variables are also used in conjunction with mathematical (=, +, -, *, /, SQRT), trigonometric (ATAN, COS, PI, SIN, TAN), and bitwise operators (&, |, ^, ~). For example, VAR1=(3+4-7*4/4+3-2/1.5)*3.

Each variable expression must be less than 80 characters in length, including the VAR1= part of the expression.

Numeric data can also be read into a variable, through the use of the READ, DAT, or TW commands (e.g., VAR1=READ1).

All variables can be used within commands that require a real or integer value. For example, the A command requires real values for acceleration; therefore, the command A(VAR1), 10, 12, (VAR2) is legal. Indirect variable assignments are also legal; (e.g., VAR(VAR1)=5 or VAR(VAR2)=VAR(VAR4)).

Rule of Thumb for command value substitutions: If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the VAR substitution.

Example:

```
VAR1=2*PI           ; Set Variable 1 to 2p
D(VAR2),,(VAR3)    ; Set the distance value on axis 1 equal to variable 2,
                   ; and the distance on axis 3 equal to variable 3
```

Indirect Variables: Numeric variables can be used indirectly. Only one level of indirection is possible (e.g., VAR(VAR(VARn)) is not a legal command). The example below shows how indirect variables are used to clear 50 variables (from 1 to 50).

Example:

```
VAR51 = 1           ; Set Variable 51 to 1
REPEAT             ; Begin repeat/until loop
VAR(VAR51) = 0     ; Clear variables (e.g., if VAR51 = 8,
                   ; then VAR(VAR51)=0 is equivalent to VAR8=0)
VAR51 = VAR51 + 1  ; Increment counter
UNTIL (VAR51 = 51) ; End repeat/until loop
```

VARB Binary Variable Assignment

Type	Variable	Product	Rev
Syntax	<!>VARB<i><=bb...bbb> (32 bits)	6K	5.0
Units	i = variable number		
Range	i = 1-125 b = 0, 1, X, or x		
Default	n/a		
Response	VARB1: *VARB1=XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX		
See Also	DVARB, PLCP, VAR, VARI, VARCLR, VARS, VCVT, WRVARB		

Binary variables can be used to store any 32-bit or less binary value. The 32-bit binary value must be in the form of 32 ones, zeros, or Xs. The information is assigned to the binary variable with the equal sign.

All variables (numeric [VAR], integer [VARI], binary [VARB], and string [VARS]) are automatically stored in battery-backed RAM.

Example: VARB1=b111100001111XXXX11110000xxxx1111
Notice that the letter b is required. The b signifies binary, 1's, 0's, and X's only.

Example: VARB1=h7F4356A3
Notice that the letter h is required. The h signifies hexadecimal, 0-9, A-F only.

Binary variables are also used in conjunction with bitwise operators (&, |, ^, and ~).

Example: VARB1=VARB2 | VARB3 & b1111000011001

The expression must be less than 80 characters in length, including the (VARB1=b or VARB1=h) part of the expression.

All binary variables can be used to set bits for commands that require at least 4 bits of binary information. For example, the OUT command requires 24 bits of binary information; therefore, the command OUT(VARB1) is legal.

Rule of Thumb for command value substitutions: If the command syntax shows that the command field requires a binary value (denoted by), you can use the VARB substitution.

Example:
VARB1=b1110 & hA ; Binary variable 1 is set to binary 1110 bitwise
; "AND"ed with hexadecimal A
VARB1=IN.7 ; State of onboard input bit 7 assigned to binary variable 1
OUT(VARB2) ; State of all onboard outputs assigned to binary variable 2

VARCLR Variable Clear

Type	Variable	Product	Rev
Syntax	<!>VARCLR	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	DVAR, DVARI, DVARB, VAR, VARB, VARI, VARS		

VARCLR resets all numeric variables (VAR), integer variables (VARI), binary variables (VARB), and string variables (VARS) to their factory default values:

Numeric (VAR) and Integer (VARI) variables are set to 0.0

Binary (VARB) variables are set to bxxxx_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX

String (VARS) variables are set to " "

VARI Integer Variable Assignment

Type	Variable	Product	Rev
Syntax	<!>VARI<i><=i>	6K	5.0
Units	1 st i = variable number 2 nd i = integer number		
Range	1 st i = 1-225 2 nd i = -2,147,483,647 to +2,147,483,647		
Default	n/a		
Response	VARI1: *VARI1=+Ø		
See Also	DVARI, PLCP, VARB, VAR, VARCLR, WRVARI		

Integer variables can be used to store integer number value, with a range from -2,147,483,647 to +2,147,483,647. The information is assigned to the variable with the equal sign (e.g., VARI1=32.).

All variables (numeric [VAR], binary [VARB], integer [VARI], and string [VARS]) are automatically stored in battery-backed RAM.

Integer variables can be used with mathematical (=, +, -, *, /) and bitwise operators (&, |, ^, ~). For example, VARI1=(3+4-7*4/4+3-2/2)*3. Numeric (VAR) and integer (VARI) variables can be mixed in the mathematical expressions. The results, if fractional, are truncated. **NOTE:** VARI cannot be used with trigonometric operators (ATAN, COS, PI, SIN, TAN) and square root (SQRT).

Each variable expression must be less than 80 characters in length, including the VAR1= part of the expression.

Numeric data can also be read into a variable, through the use of the READ, DAT, or TW commands (e.g., VARI1=READ1). Setting an integer variable to a real number results in a truncation.

All integer variables can be used within commands that require a real or integer value. For example, the A command requires real values for acceleration; therefore, the command A(VARI1), 1Ø, 12, (VARI2) is legal. Indirect variable assignments are also legal; (e.g., VARI(VARI1)=5 or VARI(VARI2)=VARI(VARI4)).

Integer variables should be used whenever possible to allow faster math operation than the numeric variables (VAR).

Rule of Thumb for command value substitutions: If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the VARI substitution.

Example:

```
VARI1=2*3           ; Set Variable 1 to 6
D(VARI2),,(VARI3)  ; Set the distance value on axis 1 equal to
                   ; integer variable 2, and the distance on axis 3
                   ; equal to integer variable 3
```

Indirect Variables: Integer variables can be used indirectly. Only one level of indirection is possible (e.g., VARI(VARI(VARI_n)) is not a legal command). The example below shows how indirect variables are used to clear 50 variables (from 1 to 50).

Example:

```
VARI51 = 1          ; Set Integer Variable 51 to 1
REPEAT             ; Begin repeat/until loop
VARI(VARI51) = Ø    ; Clear variables (e.g., if VARI51 = 8, then
                   ; VARI(VARI51)=Ø is equivalent to VARI8=Ø)
VARI51 = VARI51 + 1 ; Increment counter
UNTIL (VARI51 = 51)
```

VARs **String Variable Assignment**

Type	Variable	Product	Rev
Syntax	<!>VARs<i><="message">	6K	5.0
Units	i = variable number message = text string		
Range	i = 1-25 Message = up to 20 characters		
Default	n/a		
Response	VARs1: *VARs1="Hi John"		
See Also	' , [\] , EOT, [READ], VAR, VARB, VARCLR, VARI, VCVT, WRITE, WRVARS		

String variables can be assigned a character string up to 20 characters long. The characters within the string can be any character except the quote ("), the semicolon (;), and the colon (:). The backslash character (\) immediately followed by a number is okay.

All variables (numeric [VAR], integer [VARI], binary [VARB], and string [VARs]) are automatically stored in battery-backed RAM.

To place specific control characters that are not directly available on the keyboard within a character string, use the backslash character (\), followed by the control character's ASCII decimal equivalent. Multiple control characters can be sent.

For example, to set the string for variable #1 equal to HI MOM<cr>, use the command VARs1="HI MOM\13" where \13 corresponds to the carriage return character.

Common characters and their ASCII equivalent value:

Character	Description	ASCII Decimal Value
<lf>	Line Feed	10
<cr>	Carriage Return	13
"	Quote	34
:	Colon	58
;	Semi-colon	59
\	Backslash	92

Example:

```
VARs1="Enter velocity >" ; Assign a message to string variable #1
VAR2=READ1               ; Transmit string variable 1, and wait for numeric
                          ; data entered in the format of !'<data>.
                          ; Once numeric data is received, place it in
                          ; numeric variable 2.
                          ; Example of data entry is to type "'10.0", which
                          ; will assign numeric variable 2 the value 10.00
```

VCVT Variable Type Conversion

Type	Operator (Mathematical)	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	VAR, VARB, VARI		

Using the Variable Type Conversion (VCVT) operator, you can convert numeric (VAR or VARI) values to binary (VARB) values, and vice versa. The operation is a signed operation as the binary value is interpreted as a two's complement number, with the least significant bit (LSB) on the left and the most significant bit (MSB) on the right. A *don't care* (X) in a binary value will be interpreted as a zero (Ø).

If the mathematical statement's result is a numeric value, then VCVT converts binary values to numeric values. If the statement's result is a binary value, then VCVT converts numeric values to binary values.

You can also convert real (VAR) values to integer (VARI) values (real values are truncated in the process).

NOTE: Numeric variables (VAR) have insufficient range to convert a full 32-bit binary variable (VARB). For example, executing the VARB1=h00000004 command and then the VAR1=VCVT(VARB1) command yields an INVALID DATA error.

Numeric-to-Binary Conversion:

```
VAR1=-5           ; Set numeric variable value = -5
VARB1=VCVT(VAR1) ; Convert the numeric value to a binary value and
                  ; store in VARB1
VARB1             ; Display value of VARB1. The response should be:
                  ; *VARB1=1101_1111_1111_1111_1111_1111_1111_1111

VAR1=25          ; Set numeric variable value = 25
VARB1=VCVT(VAR1) ; Convert the numeric value to a binary value and
                  ; store in VARB1
VARB1            ; Display value of VARB1. The response should be:
                  ; *VARB1=1001_1000_0000_0000_0000_0000_0000_0000
```

Binary-to-Numeric Conversion:

```
VARB1=b0010_0110_0000_0000_0000_0000_0000_0000 ; Set binary variable = +100.0
VAR1=VCVT(VARB1) ; Convert the binary value to a numeric value
VAR1             ; *VAR1=+100.0
```

[VEL] Velocity (Commanded) Assignment

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	SCALE, SCLV, SFB, TVEL, TVELA, V, [V], VELA		

Use the VEL operator to compare the current *commanded* velocity to another value or variable, or to assign the current commanded velocity to a variable. The velocity value used in any comparison, or in any assignment statement is the current commanded velocity value, not the *programmed* velocity (V) or the actual velocity as measured from the feedback device (VELA).

Syntax: VARn=aVEL where “n” is the variable number, and “a” is the axis number, or VEL can be used in an expression such as IF(2VEL>4). When assigning the current velocity value to a variable, an axis specifier must always precede the assignment (VEL) operator (e.g., VAR1=1VEL). When making a comparison to the current velocity, an axis specifier must also be used, or else it will default to axis 1 (e.g., IF(1VEL<2Ø)).

The VEL value represents the current commanded velocity. It is not the programmed velocity (*v*). If scaling is enabled (SCALE1), the VEL value is scaled by the velocity scaling factor (SCLV).

Stepper Axes: If scaling is disabled (SCALE0), the value is measured in revolutions/sec (actual velocity in commanded counts/sec divided by the drive resolution DRES value).

Servos: If scaling is disabled (SCALE0), the value is measured in encoder revs/sec or ANI volts/sec.

Example:

```
IF(2VEL<25)      ; If the current velocity on axis 2 is less than 25 units/sec,
                  ; then do the statements between the IF and NIF
  VAR1=2V*2      ; Variable 1 = programmed velocity of axis 2 times 2
NIF              ; End the IF statement
```

[VELA]		Velocity (Actual) Assignment		
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	ENCCNT, ERES, SCLV, TVEL, TVELA, [V], V, [VEL]			

The VELA operator is used to compare the current *actual* velocity (as derived from the feedback device) to another value or variable, or to assign the current velocity to a variable. If the programmed velocity information is required, refer to the [V] operator; if the current commanded velocity information is required, refer to the [VEL] operator.

The sign determines the direction of motion. You can use the VELA operator at all times; therefore, even if no motion is being commanded, TVELA will still report a non-zero value as it detects the servoing action.

Syntax: VARn=aVELA where “n” is the variable number, and “a” is the axis number, or VELA can be used in an expression such as IF(2VELA>4). When assigning the current velocity value to a variable, an axis specifier must always precede the VELA assignment operator (e.g., VAR1=1VELA). When making a comparison to the current velocity, an axis specifier must also be used, or else it will default to axis 1 (e.g., IF(1VELA<20)).

Units of Measure:

Steppers: The velocity is always revs/sec (actual velocity in counts/sec multiplied by the ERES value if in ENCCNT1 mode, or multiplied by DRES if in ENCCNT0 mode).

Servos: If scaling is enabled (SCALE1), the velocity value will be scaled by the velocity scaling factor (SCLV). If scaling is not enabled (SCALE0), the value returned will be in encoder revs/sec or ANI volts/sec.

Example:

```
IF(2VELA<25)    ; If the current velocity on axis 2 is less than 25 units/sec,
                  ; then do the statements between IF and NIF
  VAR1=2V*2     ; Variable 1 = programmed velocity of axis 2 times 2
NIF            ; End the IF statement
```

VF		Final Velocity	
Type	Compiled Motion	Product	Rev
Syntax	<!><@>VF<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>	6K	5.0
Units	n/a		
Range	0 (non-zero values result in error message)		
Default	0		
Response	n/a		
See Also	FOLRN, FOLRD, FOLMAS, FOLMD, GOBUF, SCLD, FOLEN, V		

The Final Velocity (VF) command designates that the motor will move the load the programmed distance in a preset GOBUF segment, completing the move at a final speed of zero. VF applies only to the next (subsequent) GOBUF, which marks an intermediate “end of move” within a profile. VF is used only in conjunction with the GOBUF command. Normal preset GO moves always finish with zero velocity.

The VF command remains in effect for the affected axis until a GOBUF is executed on that axis, or until you issue a RESET command.

Any non-zero value that is entered for VF will result in an immediate error message.

[VMAS]		Current Master Velocity	
Type	Following; 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	FFILT, FMCNEW, FMCP, FOLMAS, FOLMD, SCALE, SCLMAS, TVMAS		

The Master Velocity (VMAS) command is used to assign the master velocity value to a variable, or to make a comparison against another value. The master must be assigned first (FOLMAS command) before this command will be useful.

Syntax: VARn=aVMAS where “n” is the variable number and “a” is the axis number, or VMAS can be used in an expression such as IF(2VMAS>1Ø). The VMAS command must be used with an axis specifier, or it will default to axis 1 (e.g., VAR1=1VMAS, IF(2VMAS>5), etc.).

The precision of the VMAS value is dependent upon the FFILT filter value.

If scaling is enabled (SCALE1), the velocity value is scaled by the master scaling factor (SCLMAS). If scaling is disabled (SCALEØ), the velocity value is in counts/sec.

Example:

```
IF(2VMAS>4.3) ; If the master of axis 2 is traveling at more than
                ; 4.3 user units/sec then do the IF statement
    OUT.4=b1    ; Set onboard output #4 to 1
NIF           ; End of IF statement
VAR14=3VMAS   ; Set VAR14 to axis 3's master velocity
```

WAIT() Wait for a Specific Condition

Type	Program Flow Control	Product	Rev
Syntax	<!>WAIT(expression)	6K	5.0
Units	n/a		
Range	Up to 80 characters (including parentheses)		
Default	n/a		
Response	n/a		
See Also	FMCLEN, FMCNEW, FMCP, GOWHEN, IF, NWHILE, REPEAT, [SS], T, TSS, UNTIL, WHILE		

The Wait for a Specific Condition (WAIT) command is used to wait for a specific expression to evaluate true. No commands, except for immediate commands, after the WAIT command will be processed until the expression contained within the parentheses of the WAIT command evaluates true. The COMEXC command has no effect on the WAIT command.

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the WAIT() expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single WAIT() expression.

The limiting factor for the WAIT() expression is the command length. The total character count for the WAIT() command and expression cannot exceed 80 characters. For example, if you add all the letters in the WAIT command and the letters within the () expression, including the parenthesis and excluding the spaces, this count must be less than or equal to 80.

All assignment operators (A, AD, AS, D, ER, IN, INO, LIM, MOV, OUT, PC, PCE, PCMS, PE, PER, SS, TIM, US, V, VEL, etc.) can be used within the WAIT() expression.

Example:

```
MC1                   ; Mode continuous
COMEXC1               ; Enable continuous command mode
GO1                   ; Initiate motion on axis 1
WAIT(IN=b1)           ; Wait for onboard input 1 to be active
S1                    ; Stop motion on axis 1
WAIT(MOV=b0)          ; Wait for motion complete on axis 1
COMEXC0               ; Disable continuous command execution mode
```

WHILE() WHILE Statement

Type	Program Flow Control; Conditional Branching	Product	Rev
Syntax	<!>WHILE(expression)	6K	5.0
Units	n/a		
Range	Up to 80 characters (including parentheses)		
Default	n/a		
Response	n/a		
See Also	IF, JUMP, NWHILE, REPEAT, UNTIL		

The While Statement (WHILE) command, in conjunction with the NWHILE command, provide a means of conditional program flow. The WHILE command marks the beginning of the conditional statement, the NWHILE command marks the end. If the expression contained within the parenthesis of the WHILE command evaluates true, then the commands between the WHILE and NWHILE are executed, and continue to execute as long as the expression evaluates true. If the expression evaluates false, then program execution jumps to the first command after the NWHILE. Up to 16 levels of WHILE . . . NWHILE commands may be nested.

Programming order: WHILE(expression) . . . commands . . . NWHILE

NOTE: Be careful about performing a GOTO between WHILE and NWHILE. Branching to a different location within the same program will cause the next WHILE statement encountered to be nested within the previous WHILE statement, unless a NWHILE command has already been encountered. The JUMP command should be used in this situation.

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the WHILE() expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single WHILE() expression.

The limiting factor for the WHILE() expression is the command length. The total character count for the WHILE() command and expression cannot exceed 80 characters. For example, if you add all the letters in the WHILE command and the letters within the () expression, including the parenthesis and excluding the spaces, this count must be less than or equal to 80.

All assignment operators (A, AD, AS, D, ER, IN, INO, LIM, MOV, OUT, PC, PCE, PCMS, PE, PER, SS, TIM, US, V, VEL, etc.) can be used within the WHILE() expression.

Example:

```
WHILE(IN=b1X0) ; While onboard input 1 = 1, input 3 = 0,
               ; execute commands between WHILE and NWHILE
T5             ; Wait 5 seconds
TPE           ; Transfer position of all encoders
NWHILE        ; End WHILE statement
WHILE(1ANV<2.3) ; While analog channel 1's voltage is less than 2.3 volts,
               ; execute commands between WHILE and NWHILE
TPC           ; Transfer commanded position of all axes
NWHILE        ; End WHILE statement
```

WRITE

Write a Message

Type	Communication Interface	Product	Rev
Syntax	<!>WRITE"<message>"	6K	5.0
Units	n/a		
Range	Up to 69 characters (may not use ", ; or :)		
Default	n/a		
Response	WRITE"message": message		
See Also	[\], EOT, PORT, [READ], VARS, WRVAR, WRVARB, WRVARS		

The Write a Message (WRITE) command provides an efficient way of transmitting message strings to the Ethernet port and the RS-232C or RS-485 ports. These messages can then be used by the operating program. The EOT command characters will be transmitted after the message.

Each message can be assigned a character string up to 69 characters long. The characters within the string can be any character except the quote ("), the colon (:), and the asterisk (*).

To place specific control characters that are not directly available on the keyboard within the character string, use the backslash character (\), followed by the control character's ASCII decimal equivalent.

Multiple control characters can be sent. For example, to set the message equal to HI MOM<cr>, use the command WRITE"HI MOM\13" where \13 corresponds to the carriage return character. Common characters and their ASCII equivalent values are listed below:

Character	Description	ASCII Decimal Value
<lf>	Line Feed	10
<cr>	Carriage Return	13
"	Quote	34
:	Colon	58
;	Semi-colon	59
\	Backslash	92

Example:

```
WRITE"It's a wonderful life!" ; Send the message "It's a wonderful life!"
```

WRVAR Write a Numeric Variable

Type	Communication Interface	Product	Rev
Syntax	<!>WRVAR<i>	6K	5.0
Units	i = variable number		
Range	i = 1-225		
Default	n/a		
Response	WRVAR1: +0.0		
See Also	EOT, [READ], VAR, WRITE, WRVARB, WRVARI, WRVARS		

Use the WRVAR command to transfer a specific numeric variable (VAR) to the Ethernet port and the RS-232C or RS-485 ports. Only the value and the EOT command characters are transmitted.

Example:

```
VAR1=100          ; Set variable 1 equal to 100
WRVAR1           ; Transmit variable 1 (the value +100.0 is transmitted)
```

WRVARB Write a Binary Variable

Type	Communication Interface	Product	Rev
Syntax	<!>WRVARB<i>	6K	5.0
Units	i = variable number		
Range	i = 1-125		
Default	n/a		
Response	WRVARB1: XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX		
See Also	EOT, [READ], VARB, WRITE, WRVAR, WRVARI, WRVARS		

Use the WRVARB command to transfer a specific binary variable (VARB) to the Ethernet port and the RS-232C or RS-485 ports. Only the binary value and the EOT command characters are transmitted.

Example:

```
VARB1=b1101      ; Set binary variable 1 to 1101
WRVARB1         ; Transmit binary variable 1
                ; (value transmitted =1101_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX)
```

WRVARI Write an Integer Variable

Type	Communication Interface	Product	Rev
Syntax	<!>WRVARI<i>	6K	5.0
Units	i = variable number		
Range	i = 1-225		
Default	n/a		
Response	WRVARI1: +0		
See Also	EOT, [READ], VARI, WRITE, WRVAR, WRVARB, WRVARS		

Use the WRVARI command to transfer a specific integer variable (VARI) to the Ethernet port and the RS-232C or RS-485 ports. Only the integer value and the EOT command characters are transmitted.

Example:

```
VAR11=100        ; Set integer variable 1 equal to 100
WRVARI1         ; Transmit integer variable 1 (the value +100 is transmitted)
```

WRVARS Write a String Variable

Type	Communication Interface	Product	Rev
Syntax	<!>WRVARS<i>	6K	5.0
Units	i = variable number		
Range	i = 1-25		
Default	n/a		
Response	WRVARS1: No response until a string is placed in VARS1		
See Also	EOT, [READ], VARS, WRITE, WRVAR , WRVARB, WRVARI		

Use the WRVARS command to transfer a specific string variable (VARS) to the Ethernet port and the RS-232C or RS-485 ports. Only the string and the EOT command characters are transmitted.

Example:

```
VARs1="John L" ; Set string variable 1 = "John L"  
WRVARS1        ; Transmit string variable 1 (string "John L" is transmitted)
```

XONOFF Enable/Disable XON / XOFF

Type	Communication Interface	Product	Rev
Syntax	<!>XONOFF	6K	5.0
Units	n/a		
Range	0 (disable), 1 (enable)		
Default	1 for COM1, 0 for COM2 (PORT command setting determines which COM port's XONOFF setting is checked)		
Response	XONOFF *XONOFF1		
See Also], [, BOT, DRPCHK, E, EOT, ERRCAD, ERROK, LOCK, PORT		

Use the XONOFF command to enable or disable XON/XOFF (ASCII handshaking).

XONOFF1 enables XON/XOFF, which allows the 6K product to recognize ASCII handshaking control characters. When XON/XOFF is enabled, ASCII 17 or ^Q is a signal to start sending characters; ASCII 19 or ^S is a signal to stop sending characters. XONOFFØ disables XON/XOFF.

The PORT command determines which COM port is affected by the XONOFF command. Each port will track its XON/XOFF values

RS-485 Multi-drop: If you are using RS-485 multi-drop, disable XON/XOFF by executing the PORT2 command followed by the XONOFFØ command.

NOTE: COM1 is the "RS-232" connector or "ETHERNET" connector; COM2 is the "RS-232/485" connector.