PROTEUS XES
Language Manual

First edition, Jan 2003

Revision: 1.0 – First version
Revision 1.01 - Updated as needed
Revision 1.02 – Removed DI and DO command
Revision 1.03 – Added EAACEL, EHSPD, ELSPD
Revision 1.04 – Added SETSYNCM, RSTSYNCM, SYNCOUT,
Revision 1.10 - Added LATCH
Revision 1.11 - Added BUFMOVE
Revision 1.12 – Added CSPD

Table of Contents

1. Introduction

Proteus XES controller is a standalone multi-axis motion controller with USB, Ethernet, RS-232 communication.

Proteus language is a text script language that is simple to use and understand.

Multi-tasking programs:	4 programs total capable of running in multi-tasking mode Maximum cycle time of 50 milliseconds per task 6K bytes per program with total of 24K bytes
Math functions:	Addition, Subtraction, Multiplication, and Division
Bit Functions:	AND, OR, SHIFT LEFT/RIGHT
General Purpose Variable:	256 variable with 24 bit range float
Setup Variables:	48 variable with 32 bit range long integer
Conditionals:	IF, ELSE, and WHILE statements
Subroutine:	64 subroutines per program
Coordinated Motions:	XYZU axis control with single or multiple move control
Program Control:	Run, stop, pause, and continue.

For Proteus XES controller communication is done using USB, Ethernet, or RS-232.

2. Communication

Communication with Proteus XES controller is done using:

USB
- 1 Megabits/second Communication Speed
- USB 1.1 Compliant

ETHERNET
- 10Base-T Ethernet

RS-232
- 9600 Baud rate
- 8 Data bits, No parity, 1 Stop bit, No flow control

Using Proteus IPE (Integrated Programming Environment) Windows program, you can easily communicate with the controller to program, setup, and debug. See Proteus IPE manual for details.

You can also write your custom user interface program using any of the popular GUI development tools such as: Visual Basic, Visual C++, LabView, etc. Sample codes using Visual Basic and LabView is provided to speed up your understanding and development.

2.1 Communication Protocol

Following communication protocol applies to ***all communication*** methods: USB, ETHERNET, and RS-232.

All communication is ASCII text based plus Carriage Return and End of Transmission.

ASCII Character	ASCII Value
CR (Carriage Return)	13 (0D in hex)
EOT (End of Transmission)	4 (04 in hex)

2.1.1 HOST ?   CONTROLLER Command Format

All commands from the Host to Controller is a single line command with following format:

[Valid Command][CR]

2.1.2 HOST ?   CONTROLLER Reply Format

Single Line Reply Format: (Reply to command such as PX)

[Reply][CR][EOT]

Multiple Line Reply Format: (Reply to command such as LIST PROG)

[Line 1][CR] …[Line N][CR][EOT]

No Reply Format: (Reply to command such as ABS)

[EOT]

2.1.3 Invalid Reply

Any incomplete or invalid commands are replied with "?" start character plus error code content.

2.1.3 Examples

Example 1: Get variable 1 value

HOST ? CONTROLLER

V1[CR]

HOST ? CONTROLLER

0[CR][EOT]

Example2: Set the move mode to incremental mode

HOST ? CONTROLLER

INC[CR]

HOST ? CONTROLLER

[EOT]

Example3: Send Command RUN without specifying the program number

HOST ? CONTROLLER

RUN[CR]

HOST ? CONTROLLER

?1-4[CR][EOT]

2.2 Communication DLL

Proteus XES comes with communication DLL called ProteusCom.dll. There are three API's in the DLL:

1) long **fnProteusComOpen**(	int nComSelect,
int nInstance,
char* szAddress);

Parameters:

nComSelect – communication channel selection
1 – USB
2 – TCPIP
3 – RS-232

nInstance – USB instance for multiple USB connection
1 to number of Proteus USB connection

szAddress – Communication address for RS-232 and TCPIP
For RS-232, com port selection: Example: "COM2"
For TCPIP, IP address string. Example: "10.10.6.101"

Return:

Communication handle is returned. This handle is used in other API's

Description:

This API used to open one of the communication channel: USB, TCPIP, RS-232

2) BOOL **fnProteusComClose**(	int nComSelect,
long lHandle);

Parameters:

nComSelect – communication channel selection
1 – USB
2 – TCPIP
3 – RS-232

lHandle – Communication handle

Return:

True for successful transaction or false for unsuccessful transaction

Description:

This API is used to close the communication channel that is open.
When exiting program, make sure to close the communication channel.

3) BOOL **fnProteusComSendRecv**(int nComSelect,
long lHandle,
char * szCommand,
char * szResponse);

Parameters:

nComSelect – communication channel selection

1 – USB
2 – TCPIP
3 – RS-232

lHandle – Communication handle
szCommand – command string
szResponse – reply string

Return:

True for successful transaction or false for unsuccessful transaction

Description:

This API is used to send command and get reply

3. Programming Language Overview

Proteus controller can be in one of two modes:

Interactive Mode – In this mode interactive mode commands can be sent to perform actions on the controller such as querying a motor position value, setting a variable, moving a motor, or start running a program.

Program Mode – In this mode all the commands sent to the controller are appended as a part of a program. This mode is started by command OPEN PROG [program number] and this mode is ended by CLOSE command. During downloading of the program simple syntax check is performed to ensure commands are valid ones.

Some commands are valid only for interactive mode, some only for program mode, some valid for both mode. Following are summary of commands and the valid modes.

Command	Description	Interactive Mode	Program Mode
ABORT	Stops all motion programs and all motors immediately	?	
ABS	Sets the move mode to absolute mode	?	?
ACCEL	Sets the acceleration and deceleration time	?	?
BUFMOVE	Enables buffered move	?	?
CIRCLE	Moves the X and Y axis in circular motion	?	?
CLEAR	Clears the content of currently opened program		?
CLOSE	Closes the currently opened program and	?	?
CMD	Issues Interactive command from program		?
CO	Sets or return Clear Output used for digital servo	?	?
CONTINUE	Continues the program that is paused or in warning	?	
CSPDX, CSPDY, CSPDZ, CSPDU	Performs on the fly speed change	?	?
DELAY	Delays the next line program execution by milliseconds		?
DIO	Sets or returns the configurable digital IO value	?	?
EACCEL	Sets the acceleration time for closed loop correction	?	
EHSPD	Sets the high pulse speed for closed loop correction	?	
ELSPD	Sets the low speed for closed loop correction	?	
END	Ends the program execution		?
EO	Sets or returns the enable digital outputs	?	?
EX,EY,EZ,EU	Sets or returns the encoder positions	?	?
GOSUB	Jumps to subroutine		?
HOME	Performs home search using the limit and home switch	?	?
HSPD	Sets the high pulse speed	?	?
I	Sets or returns I variable	?	?
IF ELSE	Performs if conditional check		?
INC	Sets the move mode to incremental mode	?	?
IPA	Returns or sets the IP address	?	
JOYOFF	Turns off the joy stick operation	?	
JOYON	Turns on the joy stick operation	?	

LATCHIO	Returns the latch digital input state	?	?
LATCHSTAT	Checks for latch status	?	?
LATCHX, LATCHY, LATCHZ, LATCHU	Enables position latch	?	?
LHOME	Home using the limit switch	?	?
LISTPROG	Uploads the motion program	?	
LOAD	Load the motion programs and variables from flash memory	?	
LSPD	Sets the low speed	?	?
MECLEAR	Clears the motor in error	?	
MIO	Returns the motor IO status, Lim/Home/Alarm/InPos	?	?
MMODE	Returns the move mode of incremental or absolute	?	
MST	Returns the motor status	?	
OPENPROG	Opens the program for downloading	?	
PAUSE	Pauses the selected program that is running	?	
PECLEAR	Clears the error of the motion program	?	
PEMSG	Returns the error message of the motion program	?	
PSTAT	Return the motion program status	?	
PX,PY,PZ,PU	Sets or returns the pulse positions	?	?
QUIT	Stops running motion program	?	
RSTSYNCM	Clears and resets the synchronization move	?	?
RSTOP	Performs decelerated stops to selected motors	?	?
RUN	Runs motion program	?	
SETSYNCM	Sets and enables the synchronization move	?	?
STOP	Stops selected motors	?	?
STORE	Loads the programs and variables to flash memory	?	
SUB	Indicated start of subroutine		?
SX,SY,SZ,SU	Returns the pulse rate of motors	?	?
SYNCOUTX, SYNCOUTY, SYNCOUTZ, SYNCOUTU	Sets synchronization digital output	?	?
SYNCSTAT	Returns the synchronization output status	?	?
V	Sets or returns the variable value	?	?
WAIT	Waits for the motion of the motor to complete and then continue		?
WHILE	While the condition is true execute the commands in while loop		?
X,Y,Z,U	Move the motor	?	?
ZHOME	Home using the z channel of encoder	?	?
#	Any text following this is considered program comment.		?
$	Gets communication OK reply	?	
???	Returns status of all the motors	?	

3.1 Motor Letter Assignment

In Proteus controller there are four axes that can be controlled. Each axis is assigned a letter:

Motor Number	Letter Assigned
1	X
2	Y
3	Z
4	U

All move related commands use the motor letter. For example to move X and Y motors to 1000,2000 following command is used: X1000Y2000

To move Z motor only to 250, following command is used: Z250

To clear the motor error (if the limit switch or alarm is detected while in motion) use MECLEAR command.

3.2 Speed and Acceleration Setting

For all motion, following are the motion parameters and their range:

Name	Min	Max	Unit
Low Speed	1	6,553,500	Pulse/second
High Speed	1	6,553,500	Pulses/second
Acceleration	0	10,000	msec

For low and high speed setting, sometimes the target speed is not reach to the exact value set due to discrete pulse rate generator devisor. For example, if the high speed is set to 500 the actual speed can be 498.

Acceleration time is the time for low speed to reach the high speed. If the low speed and high speed are close to each other, actual acceleration time might be reach.

Acceleration profile is true S-curve profile.

3.3 Absolute or Incremental Moves

There are two ways to move: incrementally or absolutely. Issuing INC command sets incremental moves and ABS command sets absolute moves.

If the moves are set to incremental moves, all the move commands issued will move incrementally.

For example, in the following statement motor will move continually in 1000 pulse (with acceleration and deceleration):

```
INC
WHILE 1
        X1000
ENDWHILE
```

In the following statement, motor will move to 1000 pulse counter position and not move afterwards.

```
ABS
WHILE 1
        X1000
ENDWHILE
```

3.4 Linear Interpolation Moves

Proteus has power linear interpolation capability. Linear interpolation enables all the assigned motors to start and stop at the same time as well as keeping the same ratio of speed and acceleration. This means that if the X and Y axis are in gantry system, moving X and Y in linear interpolation will make a straight line.

Any combination of the motors can be used in linear interpolation.
For example, to move X and Z motor in linear interpolation: X3000Z5000
To move all axes in linear interpolation: X1000Y2000Z3000U4000

3.5 Homing

There are three ways to perform home search:

Command	Description
HOMEX+, HOMEX- HOMEY+, HOMEY- HOMEZ+, HOMEZ- HOMEU+, HOMEU-	Uses home and limit to perform home position search. If the limit switch is triggered before the home switch, the home search direction is reversed to search home position.
LHOMEX+, LHOMEX- LHOMEY+, LHOMEY- LHOMEZ+, LHOMEZ- LHOMEU+, LHOMEU-	Uses limit switch only to perform home position search
ZHOMEX+, ZHOMEX- ZHOMEY+, ZHOMEY- ZHOMEZ+, ZHOMEZ- ZHOMEU+, ZHOMEU-	Uses Z index encoder channel to perform home position search

Home speed can be set in two ways.

	Description
I Variable bit 22 = 0	Uses low-speed home search. Axis stops immediately at home position.
I Variable bit 22 = 1	Uses high-speed home search with acceleration and deceleration. Axis accelerates to high speed and decelerates as soon as home is triggered.

See Proteus IPE setup for easy and graphical setup of ramp homing.

In home search, always the positive edge of the home switch triggered in the moving in the direction will be used to detect the home position. This ensures that consistent and reliable home position will be found no matter where the motor is.

When home position is found, the pulse position as well as the encoder position is set to zero.

3.6 Joystick Operation

Using the two analog input channels, encoders, or variables any axis can be controlled in joystick mode.

JOYON command enables the joystick operation.

JOYOFF command disables the joystick operation.

3.7 Limits, Home, Alarm and Motor Error

Home, Alarm, and Limit switch polarity can be set by software. See I variable definitions.

When limit or alarm switch is triggered when the motor is in motion, motor goes into error.

Once motor is in error, it cannot perform any more moves. Motor error must be cleared to perform any moves.

In the program, if a motor goes into error (by hitting alarm or limit), the program stops and program goes into error as well as the motor.

Motor error can be cleared using MECLEAR command.

Program error can be cleared using PECLEAR command.

3.8 Program Line and Size

Maximum length of each line in a program is 60 characters.
Maximum total size per program is 6,000 bytes.
With 4 programs, total program size per controller is 24,000 bytes.

3.9 Adding Comments to Program

Using # command, you can add comments anywhere in the program. Comments are part of the motion program, which means

1) Comments do take up program memory
2) Comments are saved when STORE command is used and loaded when LOAD command is used.
3) Comments do take up (albeit negligible) execution time.

Following are the recommendations for using comments in the program:

1) Use comments in the beginning of the program. This will minimize program execution time.
2) Use short comments. This will reduce program size usage.

3.10 Synchronization Move

Start of one axis's can be precisely synchronized with a position of another axis using the synchronization move command. Two commands are

SETSYNCM – for setting the synchronization
RSTSYNCM – for resetting the synchronization

Following is an example of X-axis that starts to move to 5000 pulses as soon as Y reaches 6000

```
HSPD10000             '***Set high speed
LSPD1000              '***Set low speed
ACCEL300              '***Set acceleration 300 msec
SETSYNCM 1, 0 , 6000  '***Master axis is Y, Slave axis is X, slave X starts move
                      '***when master Y reaches 6000
Y25000                '***Start moving the master Y to position
X5000                 '***Slave X starts the move precisely when Y reaches 6000
```

For detailed example of synchronization move start, see Example 5.

3.11 Synchronization Digital Output

A predefined digital output can be triggered in synchronization when axis position reaches set position. Following are commands to set the synchronization digital output.

SYNCOUTX – synchronization output for X-axis position.
SYNCOUTY – synchronization output for Y-axis position.

SYNCOUTZ – synchronization output for Z-axis position.
SYNCOUTU – synchronization output for U-axis position.

Following digital outputs are used to output the synchronization output signal.

	Digital IO	34 pin connector pin #
SYNCOUTX	DIO3	Pin 5
SYNCOUTY	DIO9	Pin 6
SYNCOUTZ	DIO15	Pin 17
SYNCOUTU	DIO21	Pin 18

To check if the synchronization output is done, use SYNCSTAT command.

For detailed example of synchronization output, see Example 6.

3.12 Position Latch Input

Position latch inputs are used to capture and store pulse and encoder positions. Following commands are used to enable the position latch, and to select rising or falling edge of latch input.

LATCHX
LATCHY
LATCHZ
LATCHU

To disable latch, set the value to zero (example: LATCHX=0). To enable latch at rising edge of signal, set the value to 1 (example: LATCHX=1), and at falling edge, set the value to 2 (example: LATCHX=2).

Both the encoder and the pulse positions are latched and are available in following commands.

For pulse latch position: LPX, LPY, LPZ, LPU
For encoder latch position: LEX, LEY, LEZ, LEU

Latch status can be checked using following command: LATCHSTAT. Bit 0 to 3 represent latch status of X, Y, Z, and U axis. Latch status is cleared whenever the latch is enabled. Latch status can also be set manually.

Following are dedicated latch inputs on 34 pin IO connector.

	Pin number
Latch Position X	25
Latch Position Y	26
Latch Position Z	27
Latch Position U	28

To check the latch input IO status, use LATCHIO command. First four bits of latch input status of X, Y, Z, and U.

For an example of latch input use, see Example 7.

4. Proteus Program Examples

Example 1 – Moving Motors

```
;*** BEGIN PROG 1 ***
ACCEL100                      Acceleration set to 100 msec
HSPD10000                     High speed set to 10000
LSPD1000                      low speed set to 1000
ABS                           set the move mode to absolute mode
X0 Y0 Z0 U0                   move all motors to zero position
X1000 Y2000 Z3000 U4000       move  motors to 1000,2000,3000,4000  in coordinated motion
END                           end the motion program
;*** END PROG 1 ***

;*** BEGIN PROG 1 ***
ABS                           set the move mode to absolute mode
X0 Y0                         following moves will perform square move
X1000 Y0
X1000 Y1000
X0 Y1000
X0 Y0
END                           end the motion program
;*** END PROG 1 ***

;*** BEGIN PROG 1 ***
ABS                           set the move mode to absolute mode
V1=1                          initialize variable 1
WHILE V1<10                   repeat until variable 1 reaches 10
   X0 Y0                      move all motors to zero position
   X1000 Y2000                move  motors to 1000,2000  in coordinated motion
   V1=V1+1                    increment variable by 1
ENDWHILE                      go back to while statement to check condition
END                           end the motion program
;*** END PROG 1 ***

;*** BEGIN PROG 3 ***
INC                           set the move mode to incremental mode
HSPD100000                    set the high speed
PZ=0                          set the Z pulse counter position to 0
V1=0                          set the variable 1 to 0
WHILE V1<10                   repeat until variable 1 reaches 10
  Z1000                       move Z motor by 1000 incrementally
  V1=V1+1                     index variable 1
ENDWHILE                      go back to while condition check
END                           at the end of the program, z pulse position is 10000
;*** END PROG 3 (9:63)***
```

Example 2 – Digital Inputs and Outputs and Analog Inputs

Proteus XES has 24 bits of configurable digital IO. To use the DIO bit as output, the bit has to be configured as digital output. To use the DIO bit as input, the bit has to be configured as digital input.

I variable 12 is used to configure the DIO as input or output. For example, to configure first 12 bits as outputs and the remaining 12 bits as inputs, set I variable 12 to 4095 (value of FFF in hex). Alternatively and preferably, Proteus IPE program can be used to easily configure the DIO using the setup screen. See Proteus IPE manual in setup screen section for details.

```
;*** BEGIN PROG 1 ***
DIO=4095                      Set first 12 bit of digital IO on
DIO=0                         Set all digital outputs off
DIO2=1                        turn on bit 2
DIO2=0                        turn off bit 2
V1=AI1                        set variable 1 to analog channel 1 value (0 to 5000)
V2=AI2                        set variable 1 to analog channel 1 value (0 to 5000)
END                           end the motion program
;*** END PROG 1 ***

;*** BEGIN PROG 1 ***
IF DIO1=1                     Check if digital input bit 1 is on
   DIO2=1                     If input 1 on, then Set digital output bit 1 on
ELSE
   DIO2=0                     If input 1 off, then Set digital output bit 1 off
ENDIF
END                           end the motion program
;*** END PROG 1 ***

;*** BEGIN PROG 1 ***
HSPD 500000                   set high speed to high value for quick response
LSPD 10000
ACCEL 100
ABS                           absolute mode
WHILE 1                       forever loop
  V1=DIO*10                   set variable 1 to digital IO multiplied by 10 (V1=0 to 40950)
  V2=AI1*10                   set variable 2 to analog input 1 multiplied by 10 (V2=0 to 50000)
  X V1                        move X motor to V1 position
  Y V2                        move Y motor to V2 position
ENDWHILE
END
;*** END PROG 1 ***
```

Example 3 – Subroutines

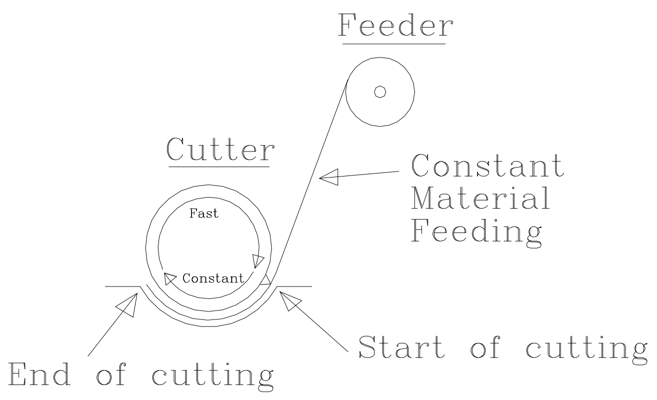
```
;*** BEGIN PROG 4 ***
V1=0                          Set variable 1 to zero
GOSUB 1                       Jump to sub routine 1
END
;****** SUB1 ******
SUB 1
  V1=V1+1                     increment variable 1
  GOSUB 2                     jump to subroutine 2
ENDSUB
;****** SUB2 ******
SUB 2
  V1=V1+1                     increment variable 1
  GOSUB 3                     jump to subroutine 3
ENDSUB
;****** SUB3 ******
SUB 3
  V1=V1+1                     increment variable 1
ENDSUB
;*** END PROG 4 (14:90)***    at the end of program value of variable 1 should be 3
```

```
;*** BEGIN PROG 1 ***
WHILE 1                       forever loop
  IF DI1=1                    if digital input 1 is on
    GOSUB 1                   jump to subroutine
    WHILE DI1=1               wait for digital input 1 to be off
    ENDWHILE
  ENDIF
ENDWHILE                      loop again
END
;****** SUB1 ******
SUB 1
  X1000                       move X motor to 1000
  X0                          move X motor to 0
  WAITX                       wait for X motor to reach 0 before exiting subroutine
ENDSUB
;*** END PROG 1 (12:81)***
```

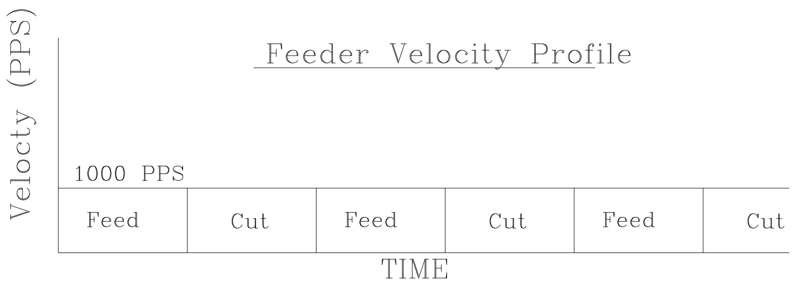
Example 4 – Math and Bit Operations

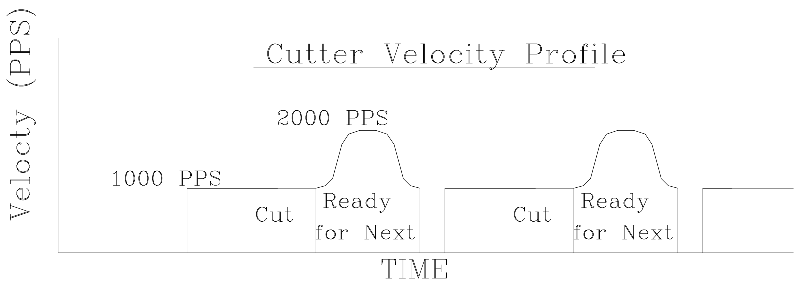
```
;*** BEGIN PROG 3 ***
V1=2
V1=V1|4                     Do bit OR of variable 1 and 4  and get 6
V1=V1{3                     Do left shift of variable 1 by 3 bits.
END                         Final value of V1 is 64
;*** END PROG 3 (4:25)***

;*** BEGIN PROG 1 ***
IF DO&8                     if bit 4 of digital output is on move to 1000
  X1000
ELSE                        if bit 4 of digital output is off move to -1000
 X-1000
ENDIF
END
;*** END PROG 1 (6:35)***

;*** BEGIN PROG 1 ***
IF DO&8
   X1000
ELSE
   X-1000
   IF V1=1                  check variable 1.  Calculate V3 depending on variable 1 status
      V3=V5*1200+123
   ELSE
      V3=V5/1200+123
   ENDIF
   YV3                      move Y axis to V3 value
ENDIF
END
;*** END PROG 1 (11:66)***
```

Example 5 – Synchronization Move



The diagram above shows an example of a Feeder and Cutter. These two must synchronize to cut the material that is fed at constant speed at precise intervals. X-axis is a Feeder conveyor that constantly feeds a roll of material at 1000 PPS. Y-axis is a Cutter module that must cut at precise intervals of 5000 pulses (Position from start of Cutting to End of Cutting) while the material is continuously fed at constant speed 1000 PPS. As soon as Cutter finishes cutting, it must rotate 8000 pulses to the start of cutting position but moving at higher speed before the start of next cutting cycle.

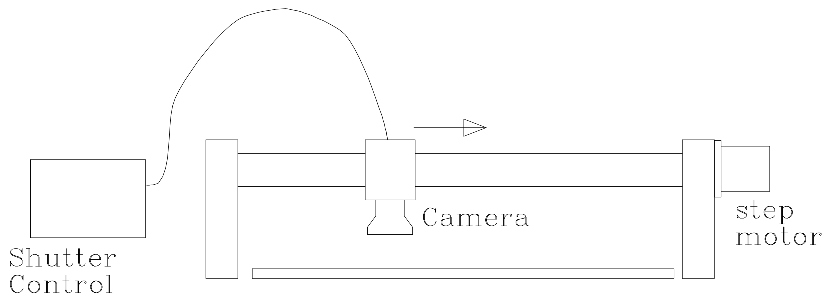




For this example, program 1 controls the X feeder axis. Program 2 controls the Y cutter axis.

```
'***** Program 1 *****
HSPD1000                '***Set high speed
LSPD1000                '***Set low speed same for constant move
BUFMOVE=1               '***Enable buffer move for smooth motion
INC                     '***Set incremental move mode
WHILE 1                 '***Forever loop
        X5000           '***Constantly feed the material
ENDWHILE
END

'***** Program 2 *****
HSPD1000
LSPD1000
ACCEL300
BUFMOVE=1
INC
V1=5000
WHILE 1                 '*** Forever loop
        SETSYNCM 0, 1, V1 '***Y will start move when X position is at V1
        HSPD1000        '*** Y sync move speed is 1000 which is same as X speed
        Y5000           '*** Y starts move exactly when X reaches V1
        RSTSYNCM 1      '*** When moving back, reset the sync move
        HSPD2000        '*** Move at higher speed to return to ready to cut
        Y8000           '***Move back to ready to cut position
        V1=V1+10000     '***Increment the next position to synchronize
ENDWHILE
END
```

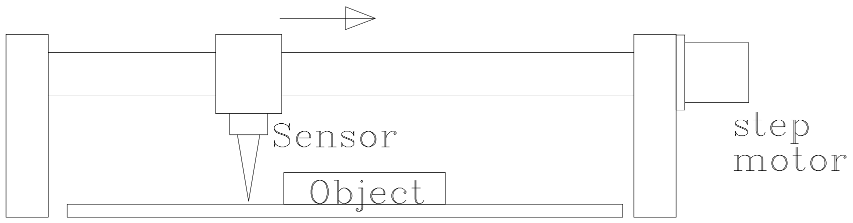
Example 6 – Synchronization Output



The diagram above shows an example of a camera on a linear track and shutter control for taking pictures. Shutter control is connected to DIO3 which is used for sync output for X axis. As the camera is moved, pictures are to be taken at exact interval (400 steps) without stopping the camera movement.

```
HSPD1000        ‘***Set High Speed
LSPD500
ACCEL300
X10000          ‘***Move X to the end of the stroke
V1=400          ‘***V1 is used to track sync output motor position
WHILE V1<10000
        SYNCOUTX=V1 ‘***Setup sync output
        V2=SYNCSTAT ‘***Check the sync output status
        WHILE V2=0      ‘***While sync output not happened, loop
                V2=SYNCSTAT
        ENDWHILE
        V1=V1+400       ‘***Increment to next sync output motor position
ENDWHILE
END
```

Example 7 – Latch Inputs



The diagram above shows an example of an optical sensor mounted on a linear track to determine the length of the object using latch function.

Assume the full stroke is 10,000 pulses. Example program below determines the length in one stroke

```
LATCHX=1        ;***Enable latch with rising edge
X10000          ;***Move the axis in one direction
WHILE LATCHSTAT=0 ;***Loop until latch is done
ENDWHILE
V1=LPX          ;***Store the latch position
LATCHX=2        ;***Enable latch with falling edge
WHILE LATCHSTAT=0
ENDWHILE
V2=LPX          ;***Store the falling edge position
V3=V2-V1        ;***Determine the length of object in pulse units.
```

In the example above, if the speed is too fast, or the object is too short, the second latch trigger can occur before the setting of LATCHX=2. For this situation, use the following example that uses two passes.

```
LATCHX=1        ;***Enable latch with rising edge
X10000          ;***Move the axis in one direction
WAITX
V1=LPX          ;***Store the first latch position
LATCHX=1        ;***Enable latch with rising edge
X0              ;***Move the axis in the other direction
WAITX
V2=LPX          ;***Store the second latch position
V3=V2-V1        ;***V3 is the length of the object in pulse units.
```

5. Proteus Language

ABORT

Format:	ABORT
Return:	
Description:	Stops all running programs and motors
Valid Mode:	Interactive Mode
Interactive Mode Example:	
	Send: ABORT
	Reply:
See Also:	
	QUIT, STOP

ABS

Format:	ABS
Return:	None
Description:	Set the move mode to absolute mode. All the moves issued after are considered absolute move commands. If this command is a part of motion program, it sets the mode to absolute for that program only.
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode Example:

ABS ***Interactive motions are now set to absolute mode

Program Mode Example:

ABS *** Motion program executing this command is set to absolute mode

See Also:

INC

ACCEL

Format:	ACCEL [Expression]
Return:	Acceleration time (for Interactive Mode)
Description:	Sets the acceleration time in milliseconds. Acceleration time is the time to ramp from low speed to high speed. If the high speed and low speed are close, actual acceleration time can be less than set acceleration time.
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode
Example:

```
Send:   ACCEL
Reply:  300              ***Current interactive mode acceleration is 300 msec

Send:   ACCEL500
Reply:                   ***Interactive mode acceleration is set to 500 msec
```

Program Mode
Example:

```
ACCEL1500

ACCEL V1                 ***Set acceleration of current program to variable 1
```

See Also:

HSPD, LSPD

AI

Format:	AI [#Analog Channel Number]
Return:	
Description:	Returns the analog input value range from 0 to 5000 for 0 to 5V range. Analog channel number if from 1 to 2 Analog input value is in milli-volts Analog input value resolution is 12 bits
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode
Example:

```
Send:   AI1           ***Ask for AI channel 1
Reply:  1500          *** value is 1.50 volt
```

Program Mode
Example:

```
V1=A2                 ***Set variable 1 to AI channel 2

WHILE A2>1000         ***While AI channel 2 is greater than 1 volt

ENDWHILE
```

See Also:

BUFMOVE

Format:	BUFMOVE=[1 or 0]
Return:	None
Description:	Set all move moves in buffer move mode. Buffer move mode enables continuous move without stopping between the move. For example, when several move is issued without buffered move mode there will be stops between the moves. With buffer move enabled, there are no stops between the moves since as soon as a move is done next move is started immediately.
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode
Example:

```
BUFMOVE=1     ***Enables buffer move mode
```

Program Mode
Example:

```
BUFMOVE=0     *** Disables buffered move mode
```

See Also:

CIRCLE

Format:	CIRCLE [Relative Center X location], [Relative Center Y location], [CW/CCW]
Return:	
Description:	Performs 360-degree circular motion for X-axis and Y-axis. Center X location is the relative location of the center of circle with respect to current location. Center Y location is the relative location of the center of circle with respect to current location. CW is the clockwise motion and CCW is the counterclockwise motion.
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode
Example:

```
Send:     CIRCLE 1000,0,CW ***Performs clockwise circle motion
                           *** Radius is 1000
                           *** Center location at (1000, 0) from the current location.
Reply:
```

Program Mode
Example:

```
CIRCLE 0,500,CCW        ***Performs counterclockwise circle motion
                         *** Radius is 500
                         *** Center location at (0, 500) from the current location.
```

See Also:

CLEAR

Format:	CLEAR
Return:	None
Description:	When downloading a program, CLEAR command clears the currently opened motion program.
Valid Mode:	Interactive Mode
Interactive Mode Example:	CLEAR
See Also:	OPEN, CLOSE

CLOSE

Format:	CLOSE
Return:	None
Description:	When downloading a program, CLOSE command closes the currently opened motion program and set the controller to interactive mode.
Valid Mode:	Interactive Mode
Interactive Mode Example:	CLOSE
See Also:	OPEN, CLEAR

CMD

Format:	CMD[Interactive Command]
Return:	None
Description:	From a program, any interactive command can be executed using the CMD command.
Valid Mode:	Program Mode

Program Mode
Example:

```
CMD JOYON     ;Issues joy stick enable command
CMD RUN 2     ;Runs program 2
```

See Also:

CO

Format:

```
COX
COX = [1 or 0]
```

Return: Returns the current clear output signal status or sets the clear output

Description: CO command is used to set or query the clear output.
CO command can be used in expression and conditional statements.
While CO output is on, the motor cannot move.

Valid Mode: Interactive Mode
Program Mode (Only assignment command using "=" is valid)

Interactive Mode Example:

```
Send:   COX
Return: 1

Send:   COY=1  (Y axis Clear output is turned on)
Return: None
```

Program Mode Example:

```
COX=1

IF COX=0
ENDIF

WHILE V1=COX
ENDWHILE

IF COX=1
ENDIF
```

See Also:

EO

CONTINUE

Format:	CONTINUE [#Program Number]
Return:	None
Description:	If the program is in PASUED state, CONTINUE command continues the program from where it is paused. If the program is in WANRING state, CONTINUE command bypass the warning and continues the program running. Program number is from 1 to 4.
Valid Mode:	Interactive Mode
Interactive Mode Example:	Send: CONTINUE 2 Return:
See Also:	PAUSE, RUN, QUIT, RETRY, WAITON/OFF

CSPDX, CSPDY, CSPDZ, CSPDU

Format:	CSPDX[speed in pps] CSPDY[speed in pps] CSPDZ[speed in pps] CSPDU[speed in pps]
Return:	None
Description:	While the motor is in motion and CSPD command can be used to change the high speed.
Valid Mode:	Interactive Mode Program Mode
Interactive Mode Example:	
	Send: CSPDX3000 ;***Changes the high speed to 3000 pps Return:
See Also:	
	HSPD, LSPD

DELAY

Format:	DELAY [Expression]
Return:	
Description:	Valid in program mode only. Waits number of milliseconds before moving to next line of program
Valid Mode:	Program Mode
Program Mode Example:	

```
DELAY 1500      ***Waits for 1.5 seconds before moving the X axis
X1000
```

See Also:

DIO

Format:	DIO DIO [#bit number] DIO = [Expression] DIO [#bit number] = [Expression]
Return:	In interactive mode, it returns the current state of configurable digital output status of each bit or all 24.
Description:	DIO command is used to access and set values of 24 configurable digital IO's. DIO command can be used in expression and conditional statements.
Valid Mode:	Interactive Mode Program Mode (Only assignment command using "=" is valid)

Interactive Mode
Example:

```
Send:   DIO
Return: 255      (this indicates that first 8 bits of configurable digital IO are on)

Send:   DIO=4095         (this sets digital IO state on)
Return: [none]

Send:   DIO1
Return  1        (This indicates that digital output bit 1 is on)

Send:   DIO1=0 (This sets the digital output bit 1 to off)
Return: [none]

Send:   DIO=V1+2         (DO is set to value of variable V1 plus 2)
Return: None

Send:   V1=DIO+5         (Variable V1 value is set to DO value plus 5)
Return: None
```

Program Mode
Example:

```
DIO=8   ***Sets bit 4 on and the other bits off

DIO8=1 ***Set bit 8 on

IF DIO>10
ENDIF

WHILE V1<DIO
ENDWHILE

IF DIO1=1
ENDIF
```

See Also:

DI, DO, I14

EACCEL

Format:	EACCEL [Expression]
Return:	Acceleration time for closed loop correction
Description:	Sets the acceleration time in milliseconds. Acceleration time is the time to ramp from low speed to high speed. If the high speed and low speed are close, actual acceleration time can be less than set acceleration time.
Valid Mode:	Interactive Mode

Interactive Mode Example:

```
Send:   EACCEL
Reply:  300             ***Current closed loop correction acceleration is 300 msec

Send:   EACCEL500
Reply:
```

See Also:

EHSPD, ELSPD

EHSPD

Format: EHSPD [Expression]

Return: High pulse speed for closed loop correction

Description: Sets the high speed in pulses per second for closed loop correction

Valid Mode: Interactive Mode

Interactive Mode Example:

```
Send:   EHSPD
Reply:  100000          ***Current closed loop correction high speed is 100000 pps

Send:   EHSPD 50000
Reply:
```

See Also:

EACCEL, ELSPD

ELSPD

Format:	ELSPD [Expression]
Return:	Low pulse speed for closed loop correction.
Description:	Sets the low speed in pulses per second for closed loop correction.
Valid Mode:	Interactive Mode

Interactive Mode
Example:

```
Send:   ELSPD
Reply:  1000            ***Current closed loop correction low speed is 1000 pps

Send:   ELSPD 500
Reply:
```

See Also:
EACCEL, EHSPD

END

Format:	END
Return:	
Description:	Valid in program mode only. When END is encountered while program is running, program is stopped and goes to idle.
Valid Mode:	Program Mode
Program Mode Example:	
	END
See Also:	

EO

Format:

EOX
EOX = [0 or 1]

Return: Returns the current enable output signal status or sets the enable output

Description: EO command is used to set or query the enable output.
EO command can be used in expression and conditional statements.

Valid Mode: Interactive Mode
Program Mode (Only assignment command using “=” is valid)

Interactive Mode Example:

```
Send:    EOY=1  (Y axis Enable out is on)
Return:  None
```

Program Mode Example:

```
EOX=1

IF EOX>0
ENDIF

WHILE V1<EOX
ENDWHILE

IF EOX=1
ENDIF
```

See Also:

CO

EX, EY, EZ, EU

Format:	EX EX=[Expression]	
Return:	signed 24 bit integer number	
Description:	Returns the current encoder position of XYZU motor. EX,EY,EZ,EU can be used in expression statements and conditional statements	
Valid Mode:	Interactive Mode	
Interactive Mode Example:		
	Send: EY	
	Return: -999	***Current Encoder position is –999
	EY=V80	***Set encoder position of Y motor to variable 80
Program Mode Example:		

	IF EZ!V1
	ENDIF
		***If encoder value is not equal to V1 execute commands after IF

	EZ=V19	***Set encoder position of Z motor to variable 19
	EU=0	***set encoder position of U motor to zero
See Also:		
	PX, PY, PZ, PU, EY, EZ, EU	

GOSUB

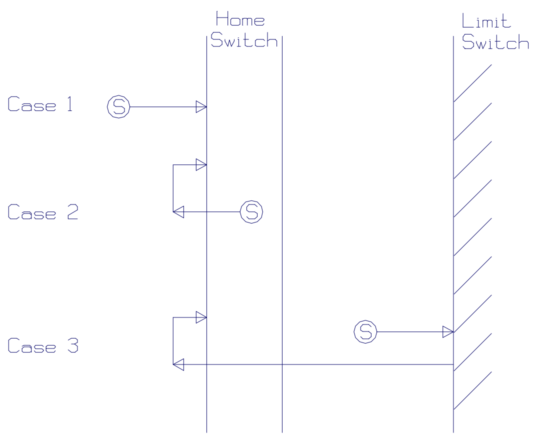
Format:	GOSUB [#Subroutine Number]
Return:	NA
Description:	Go to subroutine number and execute the commands in the subroutine and return to next line after GOSUB Subroutine number range must be from 1 to 64. Maximum number of nested subroutines is 8.
Valid Mode:	Program Mode
Program Mode Example:	GOSUB 12 ***Jump to subroutine 12 and after ENDSUB return to next line.
See Also:	SUB ENDSUB

HOMEX, HOMEY, HOMEZ, HOMEU

Format:	HOMEX+ HOMEX- HOMEY+ HOMEY- HOMEZ+ HOMEZ- HOMEU+ HOMEU-
Return:	
Description:	Home search routine uses home switch as well as limit switch to find repeatable and reliable home position. When homing low speed or ramp can be used. When low speed is used, the motor stop as zero position when home switch is detected. When ramp speed is used, the motor ramps down at zero position, and the end position is not necessarily zero. Ramp homing is useful for reducing sudden jerk when the home switch is detected. See I variable 1 for ramp home setup. In case no home switch is available, limit switch can used for homing by using LHOME command.

Following chart shows the home search routine. Note that no matter where the motor is, the homing routine guarantees consistent home position which is the positive trigger of home switch.

<u>HOME SWITCH SEARCHING</u>



Valid Mode: Program Mode/Interactive Mode

Interactive Mode Example:

Send:	HOMEX+	***Homes the X motor in positive direction
Reply:		

Program Mode Example:

HOMEY-	***Homes the Y motor in negative direction

See Also:

I Variable 1

HSPD

Format:	HSPD [Expression]
Return:	High pulse speed (for Interactive Mode)
Description:	Sets the high speed in pulses per second. Maximum value for high speed is 6M
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode Example:

Send:	HSPD	
Reply:	100000	***Current interactive mode high speed is 100000 pps
Send:	HSPD 50000	
Reply:		***Set interactive mode high speed to 50000 pps

Program Mode Example:

HSPD 12000	
HSPD V2	***Set high speed of current program to variable 2

See Also:

ACCEL, LSPD, CSPD

I

Format:	I[#Variable Number] I[#Variable Number] =[Expression]
Return:	Variable value (only in interactive mode)
Description:	I variables are setup variables. I variable Number range is from 1 to 48 I Variable can be used in expression and conditional statements. I Variables are save in non-volatile memory when STORE command is used.

I Variable Assignments:

IVAR 1

- bit 0 - boot up run prog 1
- bit 1 - boot up run prog 2
- bit 2 - boot up run prog 3
- bit 3 - boot up run prog 4
- bit 4 - enable joystick on boot up
- bit 5 –
- bit 6 – z channel polarity for X
- bit 7 - z channel polarity for Y
- bit 8 - z channel polarity for Z
- bit 9 - z channel polarity for U
- bit 10 - pulse polarity X
- bit 11 - dir polarity X
- bit 12 - pulse polarity Y
- bit 13 - dir polarity Y
- bit 14 - pulse polarity Z
- bit 15 - dir polarity Z
- bit 16 - pulse polarity U
- bit 17 - dir polarity U
- bit 18 – two clock X
- bit 19 - two clock Y
- bit 20 - two clock Z
- bit 21 - two clock U
- bit 22 – Enable Ramp Homing

IVAR 2

- bit 0 home switch polarity X
- bit 1 home switch polarity Y
- bit 2 home switch polarity Z
- bit 3 home switch polarity U
- bit 4 limit sw polarity,
- bit 5 limit home alarm noise filter on
- bit 6 Enable Closed loop control X
- bit 7 Enable Closed loop control Y
- bit 8 Enable Closed loop control Z
- bit 9 Enable Closed loop control U
- bit 10 Closed Loop Attempt Number X
- bit 11 Closed Loop Attempt Number Y
- bit 12 Closed Loop Attempt Number Z
- bit 13 Closed Loop Attempt Number U
- bit 14 In Pos Polarity X

bit 15 In Pos Polarity Y
bit 16 In Pos Polarity Z
bit 17 In Pos Polarity U
bit 18 Alarm Polarity X
bit 19 Alarm Polarity Y
bit 20 Alarm Polarity Z
bit 21 Alarm Polarity U
bit 22 Enable deceleration setting

IVAR 3 X Axis Encoder Control
bit 0-11 (encoder resolution), bit 12-23 (motor resolution)

IVAR 4 Y Axis Encoder Control
bit 0-11 (encoder resolution), bit 12-23 (motor resolution)

IVAR 5 Z Axis Encoder Control
bit 0-11 (encoder resolution), bit 12-23 (motor resolution)

IVAR 6 U Axis Encoder Control
bit 0-11 (encoder resolution), bit 12-23 (motor resolution)

IVAR 7
bit 0-11 (X closed loop tol)
bit 12-23 (Y closed loop tol)

IVAR 8
bit 0-11 (Z closed loop tol)
bit 12-23 (U closed loop tol)

IVAR 9
bit 0-11 (X closed loop err),
bit 12-23 (Y closed loop err)

IVAR 10
bit 0-11 (Z closed loop err),
bit 12-23 (U closed loop err)

IVAR 11
Bit 0-1 Encoder Multiplication X
Bit 2-3 Encoder Multiplication Y
Bit 4-5 Encoder Multiplication Z
Bit 6-7 Encoder Multiplication U

IVAR 12
Configurable Digital IO Configuration (24bit, 1- output, 0-input)

IVAR 13
Digital IO boot-up state

IVAR 14
Jog X positive outer Limit

IVAR 15
Jog Y Pos Outer Slim

IVAR 16
Jog Z Pos Outer Slim

IVAR 17
Jog U Pos Outer Slim

IVAR 18
Jog X filtering max allowed change

IVAR 19
Jog Y filtering max allowed change

IVAR 20
Jog Z filtering max allowed change

IVAR 21
Jog U filtering max allowed change

IVAR 22
bit 0-3 (enable joy),
bit 4-7(enable joy slim),
bit 8-11(Joy Dir),
bit 12-15(sourceX)
0-encodrX, 1-encoderY, 2-encoderZ, 3-encoderU,
4-AI1, 5-AI2, 6-Command
bit 16-19(sourceY)
bit 20-23(sourceZ)
bit 24-27(sourceU)

IVAR 23
Jog X Encoder Input Range

IVAR 24
Jog Y Encoder Input Range

IVAR 25
Jog Z Encoder Input Range

IVAR 26
Jog U Encoder Input Range

IVAR 27
Jog X Max Spd

IVAR 28
Jog Y Max Spd

IVAR 29
Jog Z Max Spd

IVAR 30
Jog U Max Spd

IVAR 31
Jog X zero tol

IVAR 32
Jog Y zero tol

IVAR 33
Jog Z zero tol

IVAR 34
Jog U zero tol

IVAR 35
Jog X Neg Inner Slim

IVAR 36
Jog Y Neg Inner Slim

IVAR 37
Jog Z Neg Inner Slim

IVAR 38
Jog U Neg Inner Slim

IVAR 39
Jog X Pos Inner Slim

IVAR 40
Jog Y Pos Inner Slim

IVAR 41
Jog Z Pos Inner Slim

IVAR 42
Jog U Pos Inner Slim

IVAR 43
Jog X Neg Outer Slim

IVAR 44
Jog Y Neg Outer Slim

IVAR 45
Jog Z Neg Outer Slim

IVAR 46
Jog U Neg Outer Slim

IVAR 47
Reserved

IVAR 48
EEPROM Write Cycle
Automatically incremented at every EEPROM write

Valid Mode: Program Mode/Interactive Mode

Interactive Mode Example:

Send:	I1=3	***Only program 1 and 3 are set to auto start after power up
Reply:		
Send:	I2=1	***X axis home switch is set to normally closed

Reply:

Send:	I3	
Reply:	15	***All alarms are normally closed

Program Mode
Example:

I2=10

See Also:

IF ELSE ENDIF

Format:

```
IF [Expression]
ELSE
ENDIF
```

Return: NA

Description: If the [Expression] value is non zero, the commands following the IF statement are executed until reaching ELSE or ENDIF. If the [Expression] value is zero, commands after ELSE or ENDIF are executed.

Valid Mode: Program Mode

Example:

```
‘**** If V1 is 1 turn on digital output bit 1.  If not then turn off the output bit 1.
IF V1=1
        DO1=1
ELSE
        DO1=0
ENDIF
```

```
‘**** X pulse position is greater than 0, move X axis 1000 in minus direction.
IF PX>0
        X-1000
ENDIF
```

See Also:

WHILE ENDWHILE

INC

Format:	INC
Return:	None
Description:	Set the move mode to incremental mode. All the moves issued after are considered incremental move commands. If this command is a part of motion program, it sets the mode to incremental for that program only.
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode Example:

INC ***Interactive motions are not set to incremental.

Program Mode Example:

INC *** Motion program executing this command is set to incremental mode

See Also:

ABS

IPA

Format:	IPA
Return:	None
Description:	Returns or sets the IP address of the controller for TCPIP communication. TCPIP port is always set to 101.
Valid Mode:	Interactive Mode

Interactive Mode Example:

```
Send:   IPA       ***Interactive motions are not set to incremental.
Return: 10.10.6.101

Send:   IPA=10.10.6.20  ***Sets the new IP address
Return:
```

See Also:

JOYOFF

Format:	JOYOFF
Return:	None
Description:	Turns off the joystick operation for X and Y axis.
Valid Mode:	Interactive Mode

Interactive Mode Example:

Send: JOYOFF
Return: [None]

See Also:

JOYON

JOYON

Format:	JOYON
Return:	None
Description:	Turns on the joystick operation for X and Y axis. I variable 16 sets the zero range tolerance. I variable 17 sets the maximum jog speed. Analog input 1 is used for X axis control and analog input 2 is used for Y axis control. 2.5 volts (plus/minus zero tolerance) corresponds to zero jog speed. 0 volt corresponds to maximum minus speed and 5 volt corresponds to maximum positive speed.
Valid Mode:	Interactive Mode

Interactive Mode
Example:

Send:	JOYON
Return:	[None]

See Also:

JOYOFF

LATCHIO

Format: LATCHIO

Return: 0 to 15

Description: Latch IO command is used to get the latch input status.

Bit 0 – Latch X input status
Bit 1 – Latch Y input status
Bit 2 – Latch Z input status
Bit 3 – Latch U input status

Following pins on the 34-pin IO connector are latch inputs.

	Pin number
Latch Position X	25
Latch Position Y	26
Latch Position Z	27
Latch Position U	28

Valid Mode: Program Mode/Interactive Mode

Interactive Mode Example:

```
Send:   LATCHIO          ***Enable latch X using rising edge latch input.
Reply:  5                ***X and Z axis latch inputs are on
```

Program Mode Example:

```
V1=LATCHIO               ***Store the latch input status in variable 1
```

See Also:

LATCHX, LATCHY, LATCHZ, LATCHU, LATCHSTAT

LATCHSTAT

Format:	LATCHSTAT LATCHSTAT=[value]
Return:	
Description:	Latch status command is used to determine the occurrence of latch input triggering.

Bit 0 – Latch X trigger status
Bit 1 – Latch Y trigger status
Bit 2 – Latch Z trigger status
Bit 3 – Latch U trigger status

Latch status is reset to zero when LATCHX, LATCHY, LATCHZ, and LATCHU command is used.

Latch status can be manually reset.

Valid Mode: Program Mode/Interactive Mode

Interactive Mode Example:

Send:	LATCHSTAT	***Enable latch X using rising edge latch input.
Reply:	2	***Y axis latch triggered occurred.
Send:	LATCHSTAT=0	***manually clears the latch status
Reply:		

Program Mode Example:

V1=LATCHSTAT	***Store the latch trigger status in variable 1
LATCHSTAT=2	***Manually sets the latch trigger status.

See Also:

LATCHX, LATCHY, LATCHZ, LATCHU, LATCHIO

LATCHX, LATCHY, LATCHZ, LATCHU

Format: LATCHX=[setup value]
LATCHY=[setup value]
LATCHZ=[setup value]
LATCHU=[setup value]

Return:

Description: Latch command is used for high-speed position capture of pulse and encoder positions.
Setup value
0 – disable latch
1 – enable latch using rising edge
2 – enable latch using falling edge

Following pins on the 34-pin IO connector are latch inputs.

	Pin number
Latch Position X	25
Latch Position Y	26
Latch Position Z	27
Latch Position U	28

Valid Mode: Program Mode/Interactive Mode

Example:

```
Send:   LATCHX=1            ***Enable latch X using rising edge latch input.
Reply:

Send:   LATCHX=0            ***Disable latch X.
Reply:
```

See Also:

LATCHIO, LATCHSTAT

LHOMEX, LHOMEY, LHOMEZ, LHOMEU

Format:	LHOMEX+ LHOMEX- LHOMEY+ LHOMEY- LHOMEZ+ LHOMEZ- LHOMEU+ LHOMEU-
Return:	
Description:	LHOME is used to home using only the limit switch.
Valid Mode:	Program Mode/Interactive Mode

Interactive Mode
Example:

```
Send:     LHOMEX+                    ***Homes the X motor in positive direction
Reply:
```

Program Mode
Example:

```
LHOMEY-                    ***Homes the Y motor in negative direction
```

See Also:

I Variable 1

LISTPROG

Format:	LIST PROG [#Program Number]
Return:	program contents
Description:	LIST PROG is used to upload the program contents. Program number is from 1 to 4
Valid Mode:	Interactive Mode

Interactive Mode
Example:

```
Send:    LIST PROG 1
Return:
         '*** BEGIN PROG 1***
         X1000Y1000
         END
         '*** END PROG 1 (2:15)*** (Total number of lines, total number of bytes)
```

See Also:

LOAD

Format:	LOAD
Return:	None
Description:	Loads motion programs, V variables, and I variable from non-volatile flash memory to controller. Load from flash can be done as much as possible and does not effect flash lifetime. Loading from flash memory takes from several seconds up to 20 seconds depending on motion program size.
Valid Mode:	Interactive Mode

Interactive Mode Example:

```
Send:    LOAD
Return:  Loading…
         Done!
```

See Also:

STORE

LSPD

Format:	LSPD [Expression]
Return:	Low pulse speed (for Interactive Mode)
Description:	Sets the low speed in pulses per second. Maximum value for high speed is 6M
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode Example:

```
Send:   LSPD
Reply:  1000            ***Current interactive mode low speed is 1000 pps

Send:   LSPD 500
Reply:                  ***Set interactive mode low speed to 500 pps
```

Program Mode Example:

```
LSPD 100

LSPD V2                 ***Set low speed of current program to variable 2
```

See Also:

ACCEL, HSPD

MECLEAR

Format:	MECLEARX MECLEARY MECLEARZ MECLEARU
Return:	None
Description:	MECLEAR is used to clear the motor error that might have been caused by the limit or alarm switch.
Valid Mode:	Interactive Mode
Interactive Mode Example:	Send: MECLEARX Reply:
See Also:	

MIO

Format:	MIOX MIOY MIOZ MIOU
Return:	Motor IO Status Value
Description:	Returns the motor status. bit 0 – Alarm Switch Status bit 1 - +Limit Switch Status bit 2 - -Limit Switch Status bit 3 – Home Switch Status bit 4 – InPos Switch Status bit 5 – Z Encoder Index Channel Status
Valid Mode:	Interactive Mode
Interactive Mode Example:	Send: MIOX Reply: 0
See Also:	MST, ???

MMODE

Format:	MMODE MMODE1 MMODE2 MMODE3 MMODE4
Return:	ABS or INC
Description:	Returns the absolute or incremental mode of programs or interactive environment
Valid Mode:	Interactive Mode

Interactive Mode
Example:

Send:	MMODE	***Resets program 1 from single step mode
Reply:	ABS	***Interactive environment is in absolute mode

See Also:

MST

Format:	MSTX MSTY MSTZ MSTU
Return:	Motor Status Value
Description:	Returns the motor status. bit 0 - accelerating bit 1 - decelerating bit 2 - constant speeding bit 3 – + Limit Error bit 4 – - Limit Error bit 5 – Alarm Error bit 6 – Encoder Closed Loop Out of Range Error bit 7 – Encoder Closed Loop Trial Exceed Error
Valid Mode:	Interactive Mode

Interactive Mode Example:

Send:	MSTX	
Reply:	0	***Motor not moving and home switch on

See Also:

MIO, ???

OPENPROG

Format:	OPEN PROG [#Program Number]
Return:	None
Description:	Open the program for downloading program contents. Program number is from 1 to 4. While program is running, program cannot be downloaded.
Valid Mode:	Interactive Mode

Interactive Mode
Example:

Send: OPEN PROG 1
Return:

See Also:

CLEAR, CLOSE

PAUSE

Format:	PAUSE [#Program Number]
Return:	None
Description:	Pauses the currently running program at currently executing line. Program number is from 1 to 4.
Valid Mode:	Interactive Mode

Interactive Mode Example:

```
Send:     PAUSE 3
Return:
```

See Also:

CONTINUE, QUIT, RUN

PECLEAR

Format:	PECLEAR[#Program Number]
Return:	None
Description:	PECLEAR is used to reset the motion program in error to idle.
Valid Mode:	Interactive Mode

Interactive Mode Example:

```
Send:    PECLEAR
Reply:
```

See Also:

PEMSG

PEMSG

Format:	PEMSG[#Program Number]
Return:	Latest program error message
Description:	PEMSG is used to retrieve the error message of the motion program
Valid Mode:	Interactive Mode

Interactive Mode
Example:

```
Send:   PEMSG 1
Reply:  EMSG: Abrupt End
```

Program Mode
Example:

See Also:

PECLEAR

PSTAT

Format:	PSTAT [#Program Number]
Return:	[Program Status],[Current Line],[Current Index],[Total Line],[Total Length],[Step Mode]

Description: Returns the program status
Program Status Enumeration

- 0 – idle
- 1 – running
- 2 – erred
- 3 – paused
- 4 – warning (see WAITON/OFF command)

Valid Mode: Interactive Mode

Interactive Mode
Example:

Send:	PSTAT1	***Resets program 1 from single step mode
Reply:	0,5,27,9,57,0	

See Also:

???

PX,PY,PZ,PU

Format:	PX PX=[Expression]
Return:	signed 24 bit integer number
Description:	Returns the current pulse position of XYZU motor. PX, PY, PZ, PU can be used in expression statement and conditional statements
Valid Mode:	Interactive Mode

Interactive Mode
Example:

```
Send:     PX
Return:   1000      ***Current X motor pulse position is 1000

PX=V1+DI            ***Set the X motor position to variable 1 and digital inputs
```

Program Mode
Example:

```
WHILE PX>1000            *** While the position of Y motor is greater than 1000 do the
                         ***commands below
ENDWHILE

V2 = PY * 100
```

See Also:
EX, EY, EZ, EU

QUIT

Format:	QUIT [#Program Number]
Return:	None
Description:	Stops the program from executing. Program status is set to IDLE unless program is in q error. Program number is from 1 to 4.
Valid Mode:	Interactive Mode

Interactive Mode Example:

Send: QUIT 1
Return:

See Also:

RUN, PAUSE, CONTINUE

RSTOP

Format:	RSTOP	***Stops all motors
	RSTOPX	***Stops X motor only
	RSTOPY	
	RSTOPZ	
	RSTOPU	

Return: None

Description: Stops motors in motion with deceleration.

Valid Mode: Interactive Mode
Program Mode

Interactive Mode Example:

Send: RSTOP
Return:

Send: RSTOPX
Return:

Send: RSTOPY
Return:

See Also:
STOP

RSTSYNCM

Format:	RSTSYNCM [slave axis]
Return:	None
Description:	Resets and disables the synchronization move.
	[Slave axis] – 0 to 3 corresponding to X, Y, Z, U respectively
Valid Mode:	Interactive Mode Program Mode

Interactive Mode Example:

```
Send:     RSTSYNCM2      '**** Disables sync move for Z axis
Return:
```

Program Mode Example:

```
X2000
SETSYNCM 0, 1, 300 '***Sets sync move start for Y axis when X reaches 300
Y500
RSTSYNCM 1     '**** Disables sync move for Y axis
```

See Also:

SETSYNCM

RUN

Format:	RUN [#Program Number]
Return:	None
Description:	Start the program executing from line 1. Program number is from 1 to 4.
Valid Mode:	Interactive Mode

Interactive Mode Example:

```
Send:    RUN 4
Return:
```

See Also:

PAUSE, CONTINUE, QUIT

RUNSUB

Format:	RUNSUB[#Program Number]=[Subroutine#]
Return:	None
Description:	Runs only the subroutine and stops program. Program number is from 1 to 4. Subroutine number is from 1 to 64
Valid Mode:	Interactive Mode

Interactive Mode Example:

```
Send:     RUNSUB1=10    ;*** Runs subroutine 10 from program 1
Return:
```

See Also:

RUN, PAUSE, CONTINUE, QUIT

SETSYNCM

Format: SETSYNCM [master axis],[slave axis],[master sync position]

Return: None

Description: SETSYNCM enables start of motion of slave axis when master axis reaches sync position.

[Master axis] – 0 to 3 corresponding to X, Y, Z, U respectively
[Slave axis] – 0 to 3 corresponding to X, Y, Z, U respectively
[Master Sync Position] – master axis position when the slave axis will start move

Valid Mode: Interactive Mode
Program Mode

Interactive Mode
Example:

```
Send:     SETSYNCM 0,1,2500
Return:
```

Program Mode
Example:

```
10000
SETSYNCM 0,1,300
Y3000                    '***As soon as X axis reaches 3000 pulses, Y will start move
```

See Also:

RSTSYNCM

STOP

Format:	STOP	***Stops all motors
	STOPX	***Stops X motor only
	STOPY	
	STOPZ	
	STOPU	

Return: None

Description: Immediately stops motors if in motion.

Valid Mode: Interactive Mode

Interactive Mode Example:

Send: STOP
Return:

Send: STOPX
Return:

Send: STOPY
Return:

See Also:

RSTOP

STORE

Format:	STORE
Return:	None

Description: Stores to non-volatile storage memory the following:

1) 4 motion programs,
2) 256 V variables, and
3) 64 I variable

When controller is initially powered, all the motion programs and V and I variables are loaded from non-volatile memory.
Maximum number of write to memory is 10K.
Storing to non-volatile memory takes from several seconds up to 20 seconds depending on motion program size.

Valid Mode: Interactive Mode

Interactive Mode Example:

```
Send:    STORE
Return:  Storing…
         Done!
```

See Also:

LOAD

SUB

Format: SUB[#sub number]

[#sub number] – subroutine number ranging from 1 to 100

Return: NA

Description: Sub signifies the start of subroutine in a motion program.
Subroutine must be outside of motion program.
SUB must have accompanying ENDSUB command.
Subroutine can call another subroutine but must not exceed more than 10 nested subroutines.

Valid Mode: Program Mode

Program Mode Example:

```
SUB 1
        X1000
        V1=V1+1
        X2000
        V2=V2+1
ENDSUB
```

See Also:

ENDSUB, GOSUB

SX, SY, SZ, SU

Format:	SX SY SZ SU
Return:	signed 24 bit integer number
Description:	Returns the current pulse speed of X motor. SX, SY, SZ, SU can be used in expression and conditional statements.
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode Example:

```
Send:    SU
Return:  599      '***Current pulse speed of U axis is 599
```

Program Mode Example

```
IF SY=0          ***If current speed of Y axis is zero execute the commands below IF
ENDIF

V1=SX            ***Assign pulse rate of X axis to variable 1
```

See Also:

PX,PY,PZ,PU

SYNCOUTX, SYNCOUTY SYNCOUTZ, SYNCOUTU

Format:
```
SYNCOUTX = [pulse position]
SYNCOUTY = [pulse position]
SYNCOUTZ = [pulse position]
SYNCOUTU = [pulse position]
SYNCOUTX
SYNCOUTY
SYNCOUTZ
SYNCOUTU
```

Return: signed 24 bit integer number

Description: Sync out command is used to trigger dedicated outputs when the pulse position equals the sync out position regardless of pulse speed. The width of sync pulse output equals pulse width of pulse output.

Sync output uses following dedicated IO from the 34-pin connector.

	Digital IO	34 pin connector pin #
SYNCOUTX	DIO3	Pin 5
SYNCOUTY	DIO9	Pin 6
SYNCOUTZ	DIO15	Pin 17
SYNCOUTU	DIO21	Pin 18

To disable the sync output, set the value to zero. (Example: SYNCOUTX=0 disables X axis sync output feature). When disabled, the IO goes back to original configuration state of input or output.

When sync output is set, sync status is automatically cleared. Once the sync output is triggered, sync status bit is set.

Valid Mode: Interactive Mode/Program Mode

Interactive Mode Example:

```
Send:    SYNCOUTX
Return:  1200     '***Current
```

Program Mode Example

```
HSPD 1000
LSPD 200
ACCEL 300
SYNCOUTX=5000        ***Set sync output at 5000 pulse position
X20000               ***Move X axis to 20000 pulse position
                     ***Sync output will occur at 5000 pulse position
```

See Also:

SYNCSTAT

SYNCSTAT

Format:	SYNCSTAT = [sync output trigger status] SYNCSTAT
Return:	4 bit integer number
Description:	Sync status command is used to determine if the sync output trigger has occurred or not. Sync status is a 4 bit number. Bit 0 – Sync output X status Bit 1 – Sync output Y status Bit 2 – Sync output Z status Bit 3 – Sync output U status Sync status is cleared automatically when SYNCOUT command is issued. Sync status can also be manually set.
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode
Example:

```
Send:    SYNCSTAT
Return:  1                 '***X-axis sync output trigger occurred

Send:    SYNCSTAT=0        '***X-axis sync status is cleared manually.
Return:
```

Program Mode
Example

```
HSPD 1000
LSPD 200
ACCEL 300
SYNCOUTX=5000        ***Set sync output at 5000 pulse position
X20000               ***Move X axis to 20000 pulse position
                     ***Sync output will occur at 5000 pulse position
WHILE SYNCSTAT=0     ***Wait for sync output trigger
ENDWHILE

V1=SYNCSTAT          ***Store the sync output status to variable 1
```

See Also:

SYNCOUTX, SYNCOUTY, SYNCOUTZ, SYNCOUTU

V

Format:	V[#Variable Number] V[#Variable Number] =[Expression]
Return:	Variable value (only in interactive mode)
Description:	V variables are general purpose variables. Returns the variable value or sets the variable value. Variable Number range is from 1 to 256 Variable can be used in expression and conditional statements Variables are save in non-volatile memory when STORE command is used.
Valid Mode:	Program Mode/Interactive Mode

Interactive Mode Example:

```
Send:   V1=1
Reply:

Send:   V1=V2*3+1
Reply:

Send:   V256
Reply:  300
```

Program Mode Example:

```
V100=101

V200=V201+1

IF V1>10
ENDIF

WHILE V1+1<101
ENDWHILE
```

See Also:

WAITX, WAITY, WAITZ, WAITU

Format:	WAITX WAITY WAITZ WAITU
Return:	
Description:	Wait for the motion of XYZU to be done before going on to the next line of program.
Valid Mode:	Program Mode

Program Mode Example:

```
WAITX
DO1=1               ***Wait for motion of X to be done then turn on the output
```

See Also:

WHILE ENDWHILE

Format:	WHILE [Expression] ENDWHILE
Return:	NA
Description:	While the [Expression] value is non zero, the commands following the WHILE statement are executed until reaching ENDWHILE at which time program goes back to WHILE statement.
Valid Mode:	Program Mode

Example:

```
INC
WHILE DI10=1
        X1000Y1000 ‘***While the digital input bit 10 is 1 continuously index X and Y
ENDWHILE
```

See Also:

IF

X, Y, Z, U

Format:	X[Expression] Y[Expression] Z[Expression] U[Expression]
Return:	
Description:	Move the selected axis to locations in linear coordination
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode
Example:

```
Z2000

U3000 X1200

X1000Y1000Z1000U1000  ***Move all the motors to 1000 in coordinated motion.

X V1 Y V1          ***Move X to V1 location and Y to V2 location in coordinated motion.
```

Program Mode
Example:

```
U100 Z2000

XV3 X1200

Z V2 Y V2
```

See Also:

ZHOMEX, ZHOMEY, ZHOMEZ, ZHOMEU

Format:	ZHOMEX+ ZHOMEX- ZHOMEY+ ZHOMEY- ZHOMEZ+ ZHOMEZ- ZHOMEU+ ZHOMEU-
Return:	
Description:	ZHOME is used to home using only the index channel of encoder.
Valid Mode:	Program Mode/Interactive Mode

Interactive Mode Example:

```
Send:   ZHOMEX+             ***Homes the X motor in positive direction
Reply:
```

Program Mode Example:

```
ZHOMEY-             ***Homes the Y motor in negative direction
```

See Also:

HOME, LHOME

# - Program Comment

Format:	#
Return:	None
Description:	# allows addition of comments in the motion program. Any text followed by this command will be considered as comments that will be a part of the motion program. The comment can be added to any part of the program. When storing or loading the motion program to and from the non-volatile memory, the comments are stored as well as a part of the motion program. Maximum number of characters for comments is 60 characters.
Valid Mode:	Program Mode
Example:	
	# This is a comment for this motion program
See Also:	

$ - Gets communication OK

Format:	$
Return:	OK
Description:	$ is used to check the communication. If communication is open and controller responds with OK string.
Valid Mode:	Interfactive Mode
Example:	

```
$
OK
```

See Also:

??? – Get All Status

Format: ???

Return: [pulse X position], [pulse Y position], [pulse Z position], [pulse U position],
[encoder X position], [encoder Y position], [encoder Z position], [encoder U position],
[pulse X speed], [pulse Y speed], [pulse Z speed], [pulse U speed],
[X motor status], [Y motor status], [Z motor status], [U motor status],
[digital input status],[digital output status],[enable output status]

Description: Returns status of all the motors
[Pulse X position] – 24 bit signed integer number of current pulse position
[Encoder X position] – 24 bit signed integer number of current encoder position
[Pulse X speed] – 24 bit signed integer number of current pulse rate
[X motor status] – 12 bits motor status value with each bit representing:

- bit 0 - accelerating
- bit 1 - decelerating
- bit 2 - constant speeding
- bit 3 - Alarm Input on
- bit 4 - +Limit on
- bit 5 - -Limit on
- bit 6 - Home on
- bit 7 - SD on
- bit 8 – Plus Limit Error
- bit 9 – Minus Limit Error
- bit 10 – Alarm Error

[digital input status] – 12 bit digital input status
[digital output status] – 12 bit digital output status
[enable output status] – 4 bit motor enable output status

Valid Mode: Interactive Mode

Interactive Mode Example:

Send: ???
Return: 1,2,3,4,11,12,13,14,1000,2000,3000,4000,4,4,4,4,0,0,15

(Above returns the current pulse position of XYZU as 1,2,3,4 and
current encoder position of 11,12,13,14 and
current pulse rate of 1000,2000,3000,4000
current motot status of 4,4,4,4 indicating that all motors are
moving at constant speed
current digital inputs are all off
current digital outputs are all off
current motor enable outputs are all on

See Also:

MSTAT, PSTAT

+, -, *, /   *Math operations*

Format:	[Expression] + [Expression] [Expression] - [Expression] [Expression] * [Expression] [Expression] / [Expression]
Return:	
Description:	Following math operation is possible: + addition - subtraction * multiplication / division Precedence order is + - * /.
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode
Program Mode
Example:

V1=1+2*3	***variable 1 is assigned 7
V2=1*2+3	***variable 2 is assigned 5
V1=10/3	***variable 1 is assigned value 3.33333
XV1	***X axis is sent to 3 pulse position 3

See Also:
&, |, {, }

&, |, }, {   *Bit operations*

Format:	[Expression] & [Expression] [Expression] | [Expression] [Expression] } [Expression] [Expression] { [Expression]
Return:	
Description:	Following bit manipulation operation is possible: & bit wise AND | bit wise OR } bit shift to right { bit shift to left Precedence order is & | { }.
Valid Mode:	Interactive Mode/Program Mode

Interactive Mode
Program Mode
Example:

```
V1=15&4          ***variable 1 is assigned value 4

V1=1
V2=v1{3          ***variable 1 is assigned value 8
```

See Also:

+, -, *, /

<, >, =, ! Conditional operations

Format:	[Expression] > [Expression] [Expression] < [Expression] [Expression] = [Expression] [Expression] ! [Expression]
Return:	
Description:	Following bit manipulation operation is possible: > greater than < less than = equal to ! not equal to Precedence order is < > = !.
Valid Mode:	Program Mode
Program Mode Example:	

```
IF V1>1         ***If V1 is greater than 1 do the commands following if then
ENDIF

WHILE V2!10     ***While V2 is not equal to 10 do the commands in the while loop
ENDWHILE
```

See Also:

IF ELSE THEN, WHILE ENDWHILE