Intelliware Software for LinStep+

Installation & Operating Manual

7/01 MN1855

Table of Contents

Section 1
General Information

Overview

An RS–232 communication path to a PC is required to use the Intelliware software to configure, program, and operate the LinStep+. Baldor strongly recommends the use of the optional Windows–based Intelliware software tools for configuration and programming. The optional keypad can be used for programming but not all commands described in this manual are available to the keypad. Some can only be used with serial programming techniques. Table 1–1 is a summary of the command language.

Intelliware provides a graphical control configuration environment, a program development editor, and a terminal communication package. Intelliware also provides application upload and download utilities, and an on–line help utility.

Note: RS232C and RS485 single and multi–drop connections are described in Installation and Operating manual for your LinStep+.

A sequential, interpretive command processor is used. Commands are executed one at a time, and that one command must be completed before the next command is processed. The following example shows this type of program:

Program: **`[Move] VE4 DI10 OT01 GO OT10`**

For the program [Move], the maximum move velocity is set to 4, the command incremental distance is set to 10, output 1 and output 2 are turned off and on simultaneously, axis one then moves 10 units. After axis one stops moving, output 1 is turned on and output 2 is turned off.

Command Description Format

Each of the commands listed in Table 1–1 are described in this manual. The example format. The 2–character ASCII command appears in brackets next to the keypad command. This is the Intelliware software command. Configuring LinStep+ to a specific application requires customizing a number of software parameters to match the mechanics of the system. These parameters include motor, encoder, distance, acceleration and velocity scaling, I/O, jog, home, and serial communication.

Sample format of a procedure description:

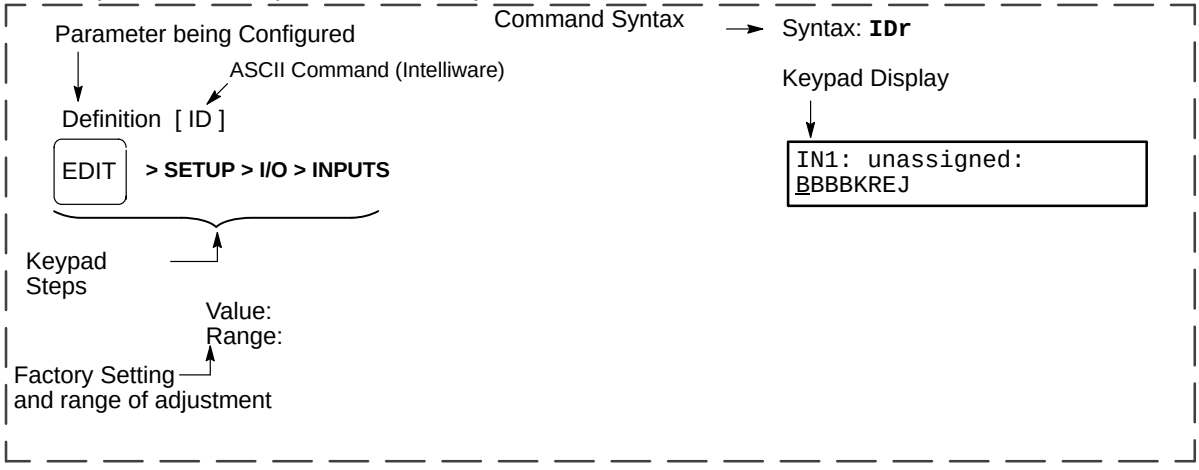


Table 1–1 Baldor Command Language Summary

Motion Commands	
AC	Acceleration
DA	Distance Absolute
DC	Distance to Change
DE	Deceleration
DI	Distance Incremental
GH	Go Home
GI	Go Immediate
GO	Start Move
GP	Go Point (Linear Interpolation)
MC	Move Continuous
RG	Registration
VE	Velocity
Program Flow Commands	
BR	Break
EB	End Block
EN	End Routine
FK	Function Key
GS	Go to Subroutine
GT	Go to Program
IF	If Conditional
LP	Loop
LU	Loop Until
LW	Loop While
ON	ON Condition
ST	Stop Move on Input
TD	Time Delay
WT	Wait
I/O and Display Commands	
IV	Input Variable
OT	Outputs On/Off
MS	Message to Keypad
“ “	Message out Serial Port
Miscellaneous Commands	
{ }	Comments
EA	Enable Amplifier
SP	Set Position
SQ	Square Root

Serial Immediate Commands	
CB	Clear Buffer
K	Kill
S	Stop
IS	Input Status
OS	Output Status
PAC	Tell Commanded Position
PAE	Tell Encoder Position
RS	Reset
SA	Axis Status
SD	Drive Status
SS	System Status
SW	Firmware Version
UN	Unit Number
Serial Supervisory Commands	
AA	Auto Address
DP	Delete Program
DR	Download Programs to RAM
EC	Enable Terminal Echo
EP	End Program
EX	End Load All
LA	Load All
LS	List Programs
OC	Original Configuration
PR	Define Program
PW	Password
RN	Run Program
UA	Upload All
UL	Upload Programs
Setup Commands	
AM	Acceleration Maximum
AU	Accel/Decel Units
DF	Display Format
DU	Distance Units
EM	Encoder Mode
GR	Gear Ratio
HE	Home Edge
HF	Home Final Direction
HM	Homing Method
HO	Home Offset
HS	Home Switch Type
ID	Input Definition
MD	Motor Direction
VU	Velocity Units

Section 2
Intelliware Overview

Overview

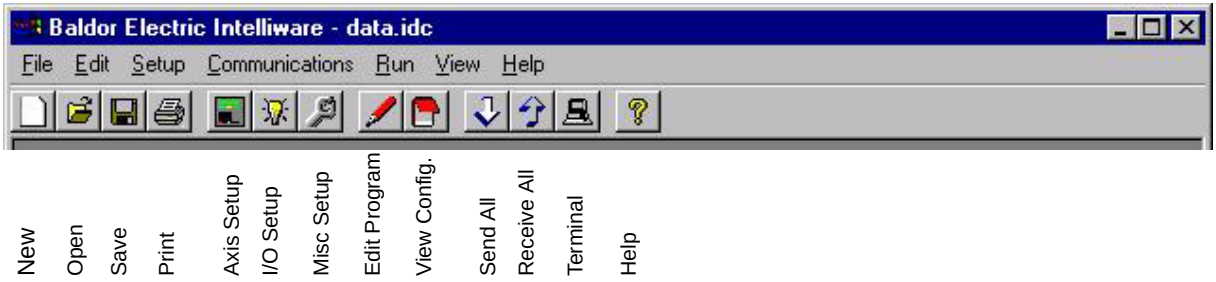
Intelliware's graphical interface helps you setup and program your LinStep+ using your personal computer. It guides you through configuring LinStep+ and motor. Applications (programs and configuration files) may be created, saved, edited and downloaded (sent) to your control for execution. A program file (setup and program memory) can also be uploaded (received) from a control for storage or modification.

Intelliware allows configuration of LinStep+, encoder, mechanics, and motion parameters using its graphical user interface. This is much faster and easier than using either the keypad or the Serial Setup Commands.

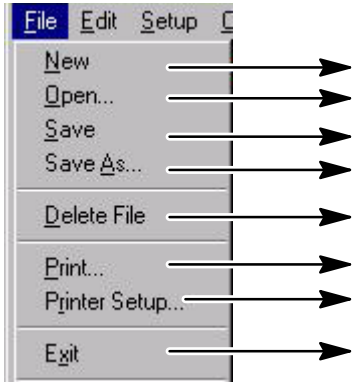
Main Menu

The main menu is shown in Figure 2-1. Start Intelliware by clicking the ICON on your desktop or selecting Start→Programs→Intelliware 1.0 →Baldor from the Windows menu bar.

Figure 2-1 Intelliware Main Menu

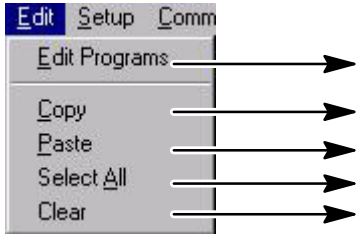


File



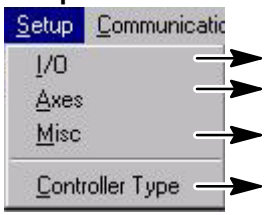
- New — Open a new configuration file (*.idc).
- Open... — Open an existing configuration file.
- Save — Save the active configuration to a file.
- Save As... — Save the active configuration to a new file name .
- Delete File — Delete an existing configuration file from your hard disk.
- Print... — Print the contents of the active editor window.
- Printer Setup... — Save all editor windows.
- Exit — Exit and close the Set up software.

Edit



- Edit Programs — Open a program file in the editor window.
- Copy — Copy the selected text in the active editor window to the clipboard.
- Paste — Paste text from the clipboard at the cursor location in the active editor window.
- Select All — Select all text in the active window.
- Clear — Delete all selected text.

Setup



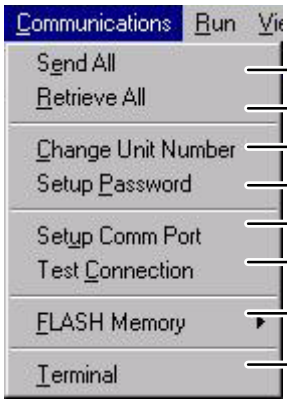
Configuration of the Discrete I/O.

Configuration of the motor, encoder, mechanics, jog and Limits parameters for each axis.

Configuration of the Diagnostic Display Format, Set Unit #, Powerup Program etc.

Select 'LinStep+ Single or Dual axis controls.

Communications



Select download target (to keypad or drive).

Select upload source (from keypad or drive).

Set unit number or select auto number for daisy chain.

Set operator and administrator password protection.

Select and set PC com port and LinStep+ unit number.

Perform a communication test with selected unit.

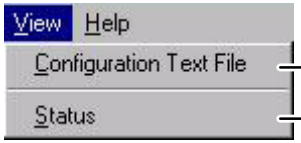
Download firmware (to unit number).

Select terminal mode communications (with unit number).

Run

Select and run a program.

View



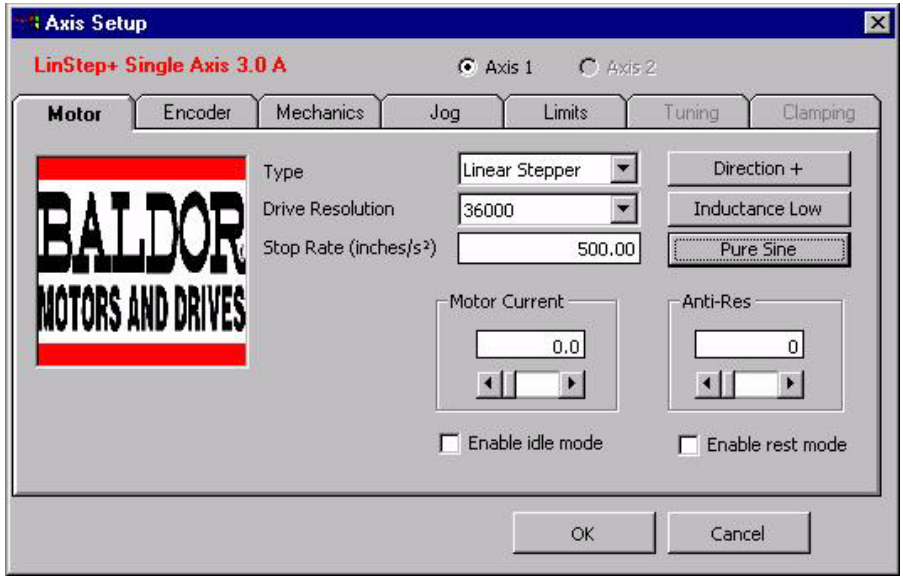
Display the configuration file (*.idc).

Display status of all I/O.

Axis Setup

Click the Axis Setup icon.
Or select “Setup→Axes”

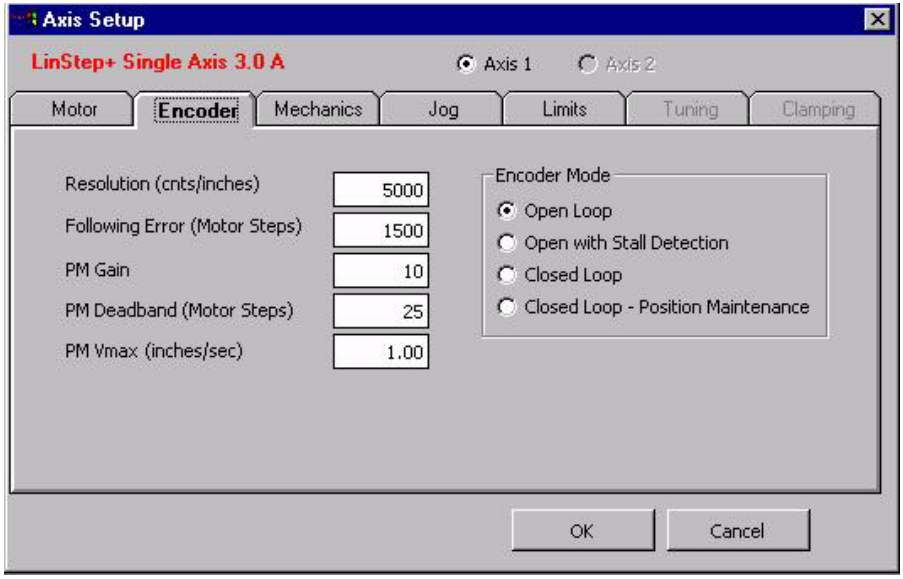
Figure 2-2 Axis Setup Menu



Motor Menu
Settings for Drive Type, Resolution, Stop Rate (Decel), and motor Directions are selected in the Motor menu. Motor Type automatically is set to Linear Stepper (no options).

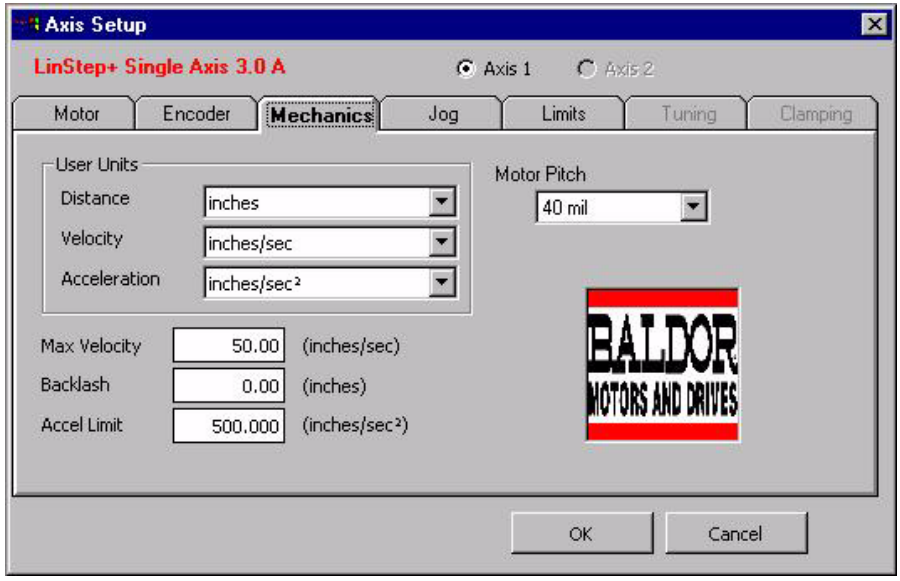
Encoder Menu

If you are not using an encoder, only the Encoder Mode must be configured. Ensure that OPEN LOOP is selected if you are not using an encoder, and skip to the Mechanics menu.



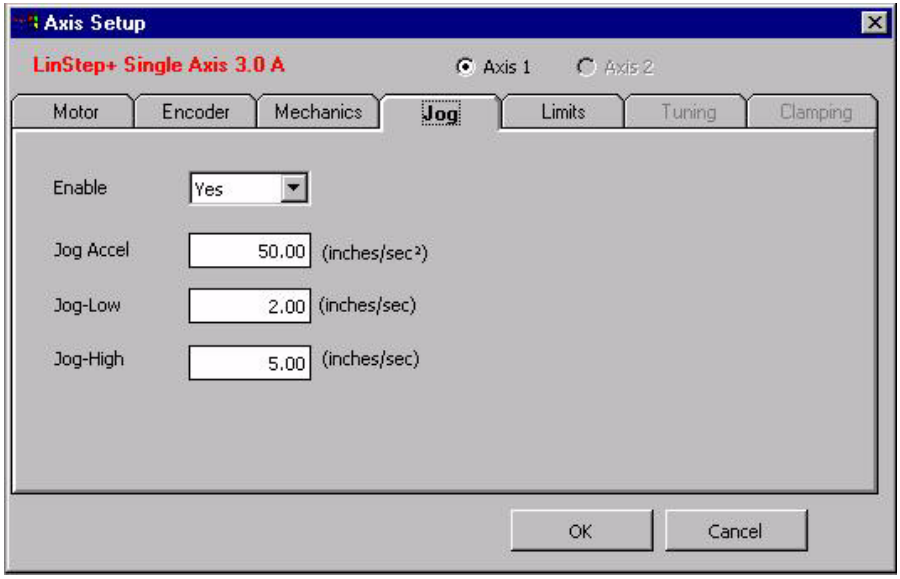
Mechanics

The Mechanics menu allows you to set program parameters like distance, velocity, and acceleration units convenient for your application. This menu also allows you to set a maximum allowable speed and acceleration for each axis.



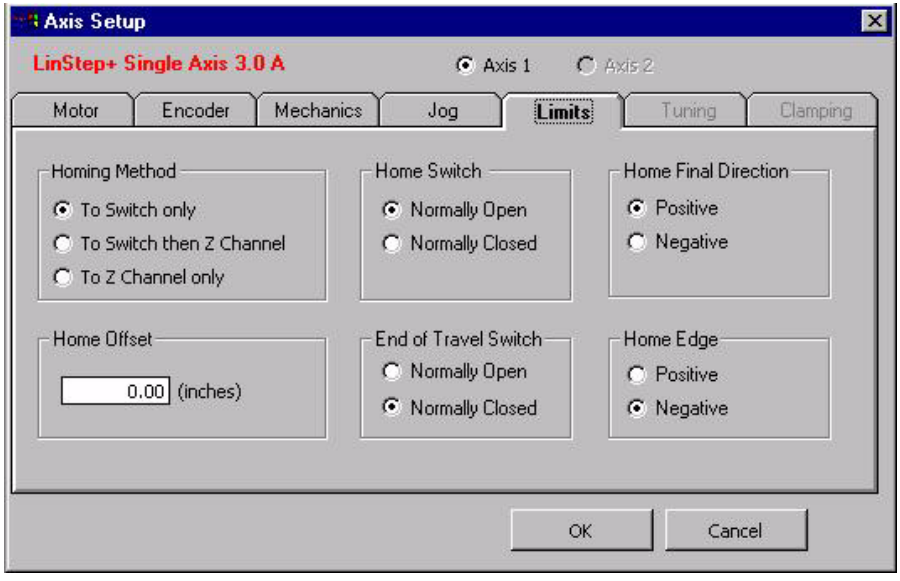
Jog Menu

The parameters which control your jog operation are configured using the Jog menu shown below.



Limits

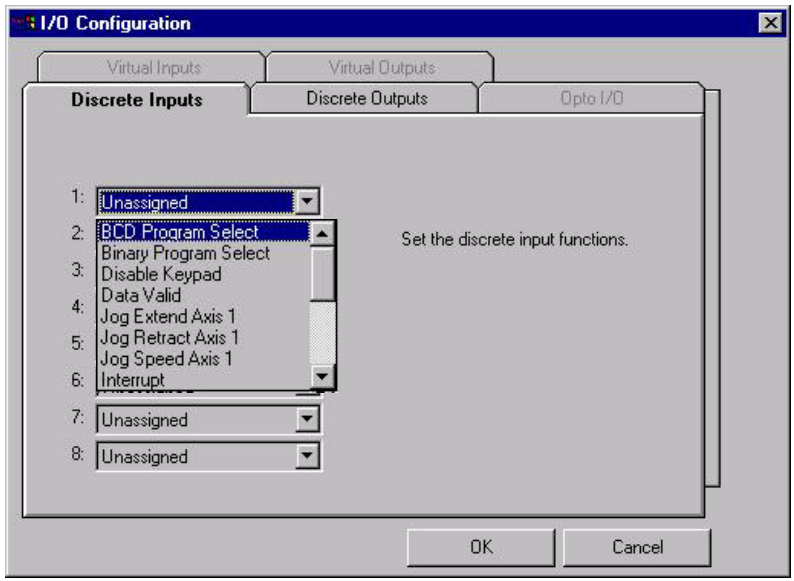
Your LinStep+ has a built–in homing function which combines the flexibility of a customized homing routine with the ease of use of calling a canned program. Also see the GH command for more details on homing.



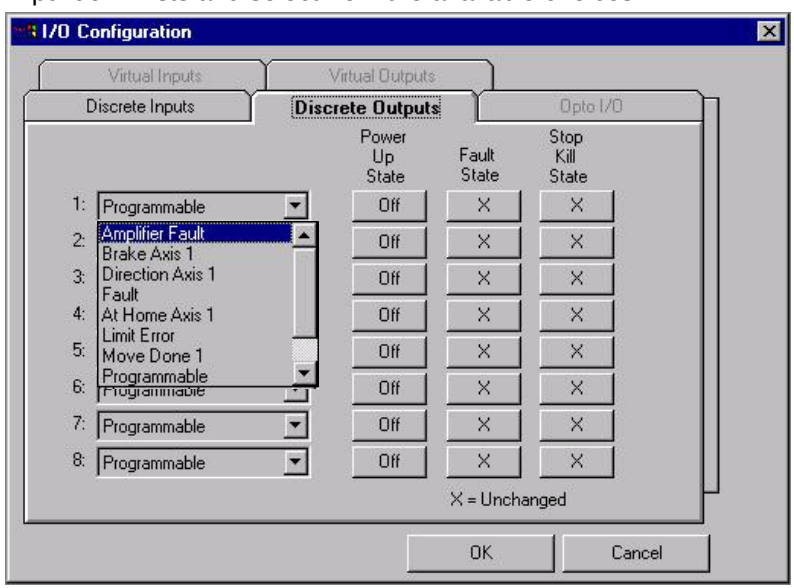
I/O Setup

Click the I/O Setup icon.
Or select “Setup→I/O”

Figure 2-3 I/O Setup Menu



Define a dedicated function for each discrete input and output, scroll through the pulldown lists and select from the available choices.

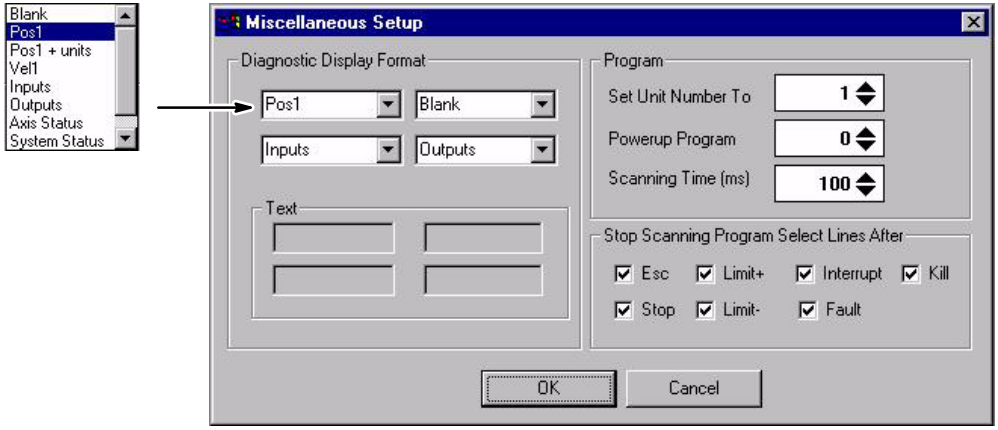


Misc Setup

Click the Misc Setup icon.
Or select “Setup→MISC”

Misc Setup allows configuration of certain diagnostic display and program formats.

Figure 2-4 Miscellaneous Setup Menu



Diagnostic Display Format (only for use with keypad display)
Diagnostic Display Format allows you to customize the data that is displayed on the Keypad display. Four displays may be set to any of the choices in the drop down box.

Program (click ▲ to increase or ▼ to decrease or type a number)

a. Sets the LinStep+ address (Set Unit Number To).
b. Sets the program to run on power–up (Powerup Program). No program will run if set to 0.
c. Sets the debounce (Scanning Time) of the program select inputs in milliseconds.

Stop Scanning Program Select Lines After...
Select the conditions to terminate program. Click on the check box to turn selection on or off.

Communications (Send All, Receive All and Terminal)

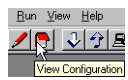
Click View Configuration any time to see the system configuration. All configuration parameters are listed and may be viewed by scrolling the list. To change a parameter, use the “Setup” menus.

Send All (Download)
Use Send All to download the application you have developed. In addition to motion programs, your application file will include the setup commands derived from the choices you made in the Setup dialog boxes. Send All completely configures the LinStep+ control, and will overwrite any existing programs or configurations in the control. The feature allows easy configuration of repeat machines. Program comments will be stripped off before being sent to the LinStep+. Save the commented version of your application before downloading.

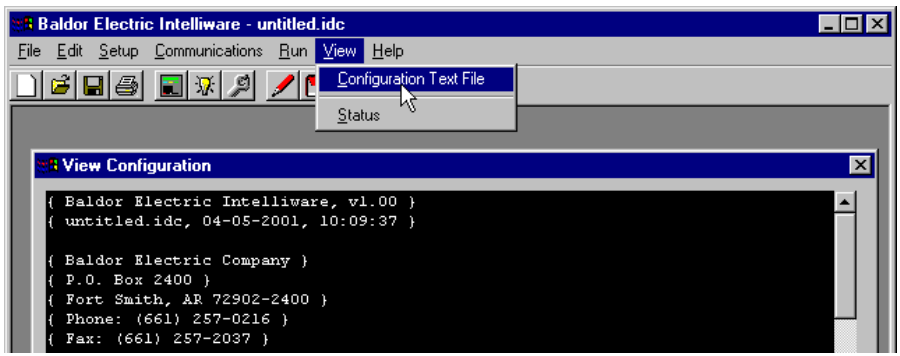
Retrieve All (Upload)
Use Retrieve All to upload the entire contents of a LinStep+ control to a new file that can then be edited, downloaded to another LinStep+, or saved to a PC file for documentation purposes. This file contains the complete contents of the LinStep+ including all the programs defined, I/O definition, and mechanical scaling parameters. Please note that this version of your application does not contain any comments, as they are stripped off during download to conserve memory in the LinStep+.

Terminal (Emulator)
Terminal emulator is used for on–line communication with a LinStep+ control. It is very useful for troubleshooting interactive host/control communications for a PC.

View Configuration (View the configuration file)



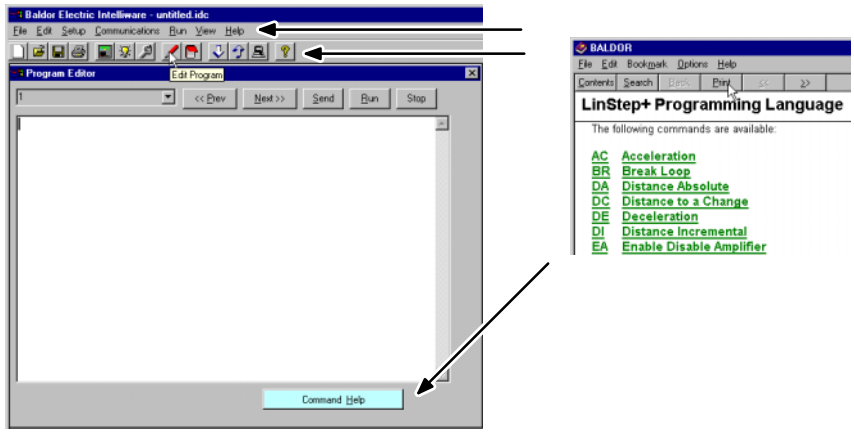
Click View Configuration any time to see the system configuration. All configuration parameters are listed and may be viewed by scrolling the list. To change a parameter, use the “Setup” menus.



Program Editor (Create or modify programs)

The Program Editor features standard windows editing features. Cut, Copy, Paste, Undo, Delete, and Select All are available by pressing the right mouse button within the window. Online help is available for commands.

Figure 2-5 Program Editor

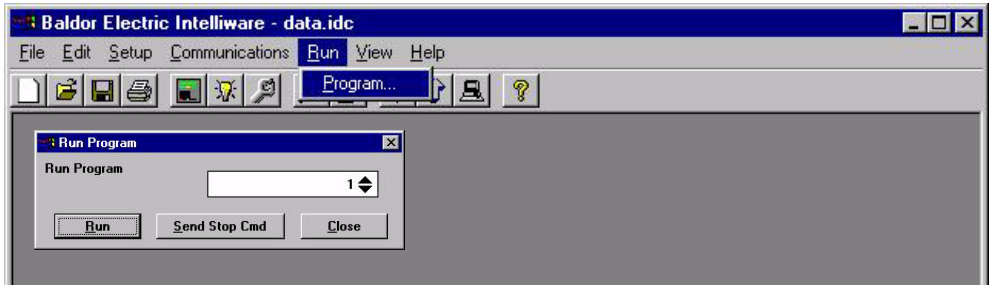


The drop down menu box (A) in the upper left hand corner shows the number and name of the currently active program, plus a list of up to 399 more programs. When the entire file is downloaded to the LinStep+, these program numbers correspond to the program numbers the controller uses for binary and BCD program selections.

Program comments are placed between brackets {comments}. These comments are not downloaded to the LinStep+ Total program length, not including comments, is limited to 1k. Total program length with comments is 8k.

Run Menu

Run→Program selects a specific program and runs that program. Programs can also be selected and ran by using dedicated program select inputs, through the keypad, or by using the RN command from the serial port.



Section 3
Programming Commands

Programming Commands

This section provides an alphabetical listing and a detailed description of each command. Commands may be sent for execution over an RS232 serial connection or written to a program file and downloaded to the LinStep+ non–volatile memory for later execution.

(This category of commands is identical to the equivalent keypad commands described in the LinStep+ manual.)

All commands use two letter UPPER CASE ASCII characters. Command delimiters can be a carriage return (<cr>) or space (<sp>) character. This listing is intended only to help programmers with command syntax.

- <n> Unit address number is optional with RS–232C, and the command is sent to all units if no address is specified. All Status commands require an address.
  Example: <n>AUi RS–485 users must address all commands.
- , Field separator.
- a Alphabetic character.
- h Hexadecimal number.
- i Decimal integer number.
- r Decimal real number (up to 4 places to the right of the decimal).
- : A colon (:) is a neutral character. It can be used in a command to make it more readable to the programmer. For example OP:OOOOIIII is easier to understand than OPOOOOIIII. Note: The colon is required in GR command and is not neutral. (Unit Ratio). Example: GR4:1

Some commands request a response from the control. Responses are preceded by an asterisk (*) (which notifies other controls on a daisy chain to ignore the subsequent response characters). For example, the Input State (IS) command might return *AF09<cr> Your computer program will need to mask the asterisk before decoding the value returned. You can document your programs by placing comments between brackets. For example: {this is a comment}. To maximize program storage space, the control strips these comments when a program file is downloaded.

AC Acceleration syntax – ACr

Value: N/A
Units: seconds, in/sec^2 or units/sec^2 (set in EDIT > SETUP > MECH > ACCEL**)**
Range: Unit scaling dependent

Sets the acceleration and deceleration ramps on all velocity changes. The deceleration value (DE) is the same as the acceleration value unless DE is specifically set after the AC command (the value of DE must then be changed every time AC is changed). Change only AC if you want a symmetrical move profile.

Examples:

AC2 VE12 DA3 GO	Sets acceleration and deceleration to 2.
DE.5 VE12 DA6 GO	Accel stays at 2, decel changes to 0.5.
VE20 DA0 GO	Acceleration and deceleration remain at 2 and 0.5.
AC4 DA2 GO	Acceleration and deceleration become 4.
DE3 AC1 DI3 GO	DE reset to 1 by AC1 before the move is made.

BR Break syntax – BR

Value: N/A
Units: N/A
Range: Unit scaling dependent

For use within program only, not for hosted mode.

The Break command instantly “Breaks” the program loop in which it is defined and continues program execution from the loop’s terminating EB command. This allows for more complex loop conditioning than the “LU” or “LW” commands.

Example:

```
(A)=0                          {Define variable A}
(B)=0                          {Define variable B}
LP                             {Define loop block}
   IF(A)>10                    {Check if A is greater than 10}
      IF2,0                    {Check if Input #2 is Off}
         BR                    {Break loop}
      EB
   EB
(A)++                          {Increment A by 1}
EB                             {BR command jumps to here}
MS1, “A is greater than 10”    {Display message}
```

DA Distance Absolute syntax – DA±r

Value: N/A
Units: set in EDIT > SETUP > MECH > DIST
Range: Unit scaling dependent

Sets the next move position, referenced from absolute zero. The absolute zero position is established after a Go Home move (GH) and/or with the Set Position (SP) command. Absolute positioning is typically used to move between a number of known locations, or if the physical work area is restricted.

Incremental (DI) and absolute moves may be mixed; the control keeps track of the absolute position.

Example:

AC2 DE.5 VE12 DA3 GO	Moves to absolute position 3 units.
DA3 GO DA3 GO	Moves once to absolute position 3 units.

DC Distance to Change syntax – DC±r

Value: N/A
Units: set in EDIT > SETUP > MECH > DIST
Range: Unit scaling dependent

Defines complex, multiple velocity move profiles, or to change an Output at a specific point during the move. It defines the distance at which a change will occur, “on the fly”, while the motor is still moving. At the specified distance you can change the velocity, acceleration, deceleration or change the state of an output(s). The DC command must follow the DA or DI command which specifies the total move distance.

The DC distance is interpreted as an absolute position when used with DA and an incremental position when used with DI. When used with DI, the value of DC should be specified as a positive number. When multiple DC’s are specified within an incremental move (DI), the incremental distance specified by the DC command is taken from the last DC command, not from the beginning of the move. A maximum of 20 DC commands within a move profile are supported.

Examples:

AC.05 DE.05 VE10 DA4 DC1 OT100 DC2 OT010 DC3 OT001 GO
While moving to an absolute position of 4 units turn on output 1 at 1 unit, output 2 at 2 units and output 3 at 3 units.

AC.05 DE.09 VE30 DA6 DC3 VE15 GO
Move to absolute position 6 units with a starting speed of 30. At 3 units, reduce speed to 15 (change–on–fly) and complete move.

AC1 DE.5 VE20 DI–8 DC1 OT10 DC3 OT01 GO
Move an incremental distance of negative 8 units. After 1 unit turn on output 1 and after 3 MORE units of motion, turn off output 1 and turn on output 2.

AC.05 DE.15 VE50 DI15 DC5 VE10 DC5 VE5 GO
At a starting speed of 50, begin moving an incremental distance of 15 units. After 5 units, ramp down to 10 speed. After 5 MORE distance units ramp down to 5 speed and continue until the final position is reached.

Applications Information: (Distance to Change)

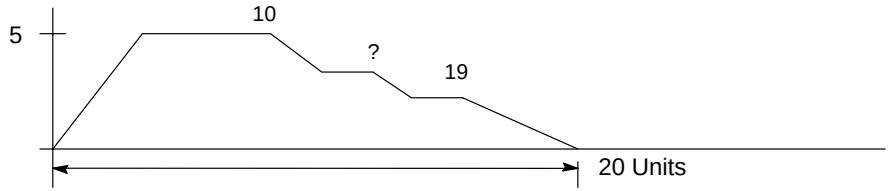
The DC command can only be used when the motor is moving at constant speed (no acceleration of deceleration). Issuing a DC command before a previous DC command has finished executing is invalid and can cause unpredictable results. (For example, “AC1 VE DA20 DC1.75 VE7.5 GO” is incorrect use of the DC command). The initial acceleration ramp requires 2.5 units to reach velocity $S= 0.5V_t$, the DC1.75 is an invalid trigger position and is ignored. The following formula ensures the use of valid DC trigger positions:

$$DC_n - DC_{n-1} - \frac{(|V_{n-1} - V_n|)^t}{2} \geq 0$$

Where:
n is the commanded DC distance (n=19 in this example)
n–1 is the previous commanded DC distance (for example 10)
V is the velocity
t is the acceleration time
(The first commanded DC move in a profile, n–1 is the beginning of the move).

Examples of DC move profiles:, AC = seconds, VE=ips.

AC1.6 DE0.8 VE5 DA20 DC10 AC2.5 VE3 DC? VE2.5 GO



To calculate DC_n (or DC? in the example)

$$DC_n = \frac{(|5 - 3|)\ (2.5)}{2} + 10 = 12.5$$

“?” must be ≥ 12.5 distance units. Also, the DC trigger position and the DC? VE2.5 segment must be verified or determined before the beginning of the move declaration. If “?” is chosen to be 13.35 (a valid trigger position), use the beginning of the decel ramp as DC_n in the DC formula. A decel ramp from 2.5 to 0 requires 1 distance unit in 0.8 seconds ($S= 0.5V_t$).

$$19 - 13.35 - \frac{(|0 - 2.5|)\ (0.8)}{2} = 5.65$$

Since the result is positive, the DC13.35 VE2.5 is a valid segment.

AC.05 DE.05 VE10 DA4 DC1 OT100 DC2 OT010 DC3 OT001 GO

{While moving to an absolute position of 4 units, turn on output 1 at 1 unit, output 2 at 2 units and output 3 at 3 units}

AC.05 DE.09 VE30 DA6 DC3 VE15 GO

{Move to absolute position 6 units with a starting speed of 30. At 3 units, reduce speed to 15 (change–on–fly) and complete move}

AC1 DE.5 VE20 DI–8 DC1 OT10 DC3 OT01 GO

{Move an incremental distance of negative 8 units. After 1 unit, turn on output 1, and after 3 additional units of motion, turn off output 1 and turn on output 2}

AC.05 DE.15 VE50 DI15 DC5 VE10 DC5 VE5 GO

{At a starting speed of 50, begin moving an incremental distance of 15 units. After 5 units, ramp down to 10 speed. After an additional 5 distance units, ramp down to 5 speed and continue until the final position is reached}

DE Deceleration syntax – DE±r

Value: N/A
Units: seconds, in/sec^2 or units/sec^2 (set in EDIT > SETUP > MECH > ACCEL**)**
Range: Unit scaling dependent

Sets the deceleration ramp for all negative velocity changes. This value is the same as the acceleration value unless a deceleration is specified. The value is used on subsequent moves unless it is re–specified by an acceleration (AC) or deceleration (DE) command.

Examples:

AC2 VE12 DA3 GO	Sets acceleration and deceleration to 2.
DE.5 VE12 DA6 GO	Accel stays at 2 and decel changes to 0.5.
VE20 DA0 GO	Acceleration and deceleration remain at 2 and 0.5.
AC4 DA2 GO	Both acceleration and deceleration become 4.
DE3 AC1 DI3 GO	AC1 sets both the accel and decel to 1.

DI Distance Incremental syntax – DI±r

Value: N/A
Units: set in EDIT > SETUP > MECH > DIST
Range: Unit scaling dependent

Specifies a move distance relative to the current position. Such moves are called incremental moves, as opposed to the absolute move (DA). Use incremental moves when there is no concern for origin, such as feed–to–length applications. DI is also often used inside a loop to shorten a program. Incremental and absolute moves may be mixed; the control keeps track of the absolute position.

Example:

AC.1 VE60 DI2 GO DI1 GO DI–4 GO
Move 2 units in the + direction. Move 1 more unit in the positive direction. Move 4 units in the negative direction. The final absolute position is –1.0000.

EA Enable/Disable Amplifier syntax – EAi

Value: 0 (disable), 1 (enable), 2 (standby)
Units: N/A
Range: N/A

EA sets the state of the amplifier enable signal. The polarity can be changed in EDIT > SETUP > MISC > ENABLE.

Example:

EA0 Disables the amplifiers on axis one.

EB End of Block — syntax – EB

Value: N/A
Units: N/A
Range: N/A

For use within program only, not for hosted mode.

The EB command designates the End of a Block of loop or IF commands. Every LP, LW, LU, and IF statement must have an EB associated with it.

Examples:

LP2 DI3 GO EB	Performs the move twice
IF1,1 DI5 GO DI10 GO EB GH3	If input 1 is On, make 2 moves before homing. If input 1 is Off, jump to the GH command.

EN End of Routine — syntax – EN

Value: N/A
Units: N/A
Range: N/A

For use within program only, not for hosted mode.

EN marks the end of a program or subroutine. It is optional at the end of a program. If EN marks the end of a subroutine, command execution continues from the command following the gosub (GS) command that called the subroutine. If the routine was not called from another program, the EN command simply stops execution. The control continues to monitor the program select inputs (if defined). The EN command can be used anywhere in a program to stop command execution.

Example:

IF2,1 EN EB DI2 GO	If input #2 is on, stop the program, or return to the calling program. If not, move 2 units.

FK Function Key

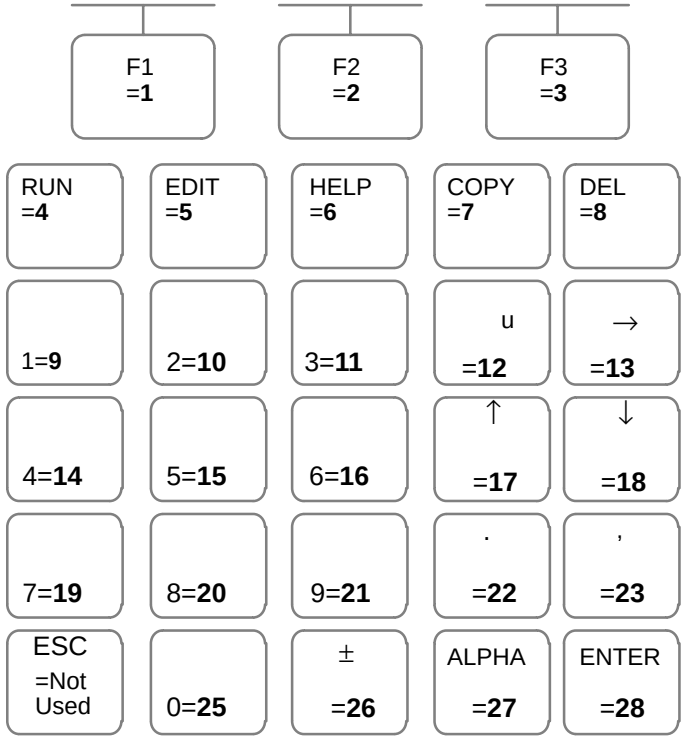
syntax – FKi,i,..,i

Value: N/A
Units: N/A
Range: i=1–28

For use within program only, not for hosted mode.

The FK command allows you to define a function key within your program. The FK command pauses processing until the buttons you have "armed" are pressed. The number of the button pressed is assigned to the system variable, (FKEY). You can then manipulate or directly use this variable to branch to other routines or make other decisions. FK allows the programmer to redefine the keypad's function keys as operator menu selection buttons. You can even write your program with menus that look and feel like our setup menus. The returned values of the FKEY's are:

Note: 24, the ESC key, cannot be assigned since it stops the program.



Example:

FK1,2,3,4
GS(FKEY)

Pauses command execution until F1, F2, F3, or RUN is pressed on the keypad. (FKEY) is assigned a value of 1–4. Subroutine 1–4 is called with the GS(gosub) command.

Figure 3-6 shows how to use the keypad function keys as an operator interface. A 3–screen menu program is provided

1. Write a menu message (MS) on the keypad display above the corresponding function keys.
2. Use the FK command to pause command processing until the operator selects a valid function key. Only keys explicitly defined in the FK statement are considered valid.
3. Gosub to the appropriate program.

Figure 3-6 Example 3–Screen Menu Program

Program 20:	
[SCREEN 1]	Name the main program
MS1,“ “	Clears keypad screen
MS3,“Select a Part”	Writes a Message
MS21,”Part A Part B Part C”	Writes a message above function keys
FK1,2,3,17,18	Wait for selected key press
GT(FKEY)	Jumps to prog# 1, #2, or #3 if F1,F2, or F3 is pressed Jumps to prog #17, or #18 if the up or down arrow keys are pressed.
EN	End of Routine
Program 18:	
[SCREEN 2]	
MS21,“Part D Part E Part F”	Writes a message above F1, F2, F3.
FK1,2,3,17,18	Wait for selected key press
IF(FKEY)=17 GT[SCREEN 1]	EB If Up arrow goto screen 1
IF(FKEY)=18 GT[SCREEN 3] EB	If Down arrow goto screen 3 (FKEY)=(FKEY)+3 Add offset to FKEY variable to goto correct part subroutine.
GT(FKEY)	Jumps to part D, E, F in program#4, 5, or 6
EN	End of Routine
Program 17:	
[SCREEN 3]	
MS21,“Part G Part H Part J”	Writes a message above function keys.
FK1,2,3,17,18	Wait for selected key press
IF(FKEY)=17 GT[SCREEN 2] EB	If Up arrow goto screen 2
IF(FKEY)=18 GT[SCREEN 1] EB	If Down arrow goto screen 1
(FKEY)=(FKEY)+6	Add offset to FKEY variable to goto correct part subroutine
GT(FKEY)	Jumps to part G, H, J in program #7,8 or 9
EN	End of Routine

The programs to make Parts A, B, C, D, etc. are in program numbers 1–9. To continuously cycle through put a GT[SCREEN 1] at the end of each part program.

GH Go Home syntax – GH±r

Value: N/A
Units: set in EDIT > SETUP > MECH > ACCEL > VEL
Range: Unit scaling dependent

Initiates a homing routine (seeks the home switch) to establish a home reference position. When it reaches home, the position counter is set to zero or to the Home Offset (HO) value selected in the EDIT > SETUP > HOME menu. The motor will move at the GH velocity (n) and direction (±) specified until it either finds a home limit switch or determines that it can not find one between the two end–of–travel limit switches. The Go Home move uses the last acceleration and deceleration specified.

The exact homing routine used, and the ultimate end position of your system's home reference, depends on the values of your EDIT > SETUP > HOME parameters (edge, level, final approach direction, and offset) and if you have specified open or closed loop moves in the EDIT > SETUP > ENCODER menu. The control will reverse direction when the first End of Travel limit switch is encountered while searching for a Home switch. If the second End of Travel switch is encountered, the unit will abort the Go Home move and generate a fault. Assuming the presence of an operational home switch, the control will seek a home position according to the parameters you specified (edge, level, final approach direction, and offset).

Closed loop systems will normally home with more accuracy than open loop systems because encoders have a Z marker pulse. In a typical Go Home routine, the control will first sense the edge of the switch defined in the Go Home SETUP menu. It will then decelerate the motor to a stop at the last defined deceleration rate. The final homing motion is determined by the Go Home options selected in the SETUP menu. The final homing direction dictates the direction from which the final approach to the switch is made. The edge selected will determine which side of the home switch this final approach will be based from.

In a "closed loop" mode Go Home routine, the control will additionally slow to a creep speed and stop when it sees the encoder's "Z" Marker Pulse after seeing the reference edge of the switch. If a marker pulse is not seen within one motor revolution after the reference edge of the switch is seen, the final homing routine will be aborted.

Note: Homing Mode directly affects or reconfigures the function of the GH command.

Examples:

AC.5 DE.5 GH–20	Go Home in the negative direction at a speed of 20
AC.5 DE.5 GH20	Axis one Go Home in the positive direction at a speed of 20.

GI Go Immediate syntax – GI or GIi

Value: N/A
Units: N/A
Range: N/A

The GI command begins a defined move profile in the same manner as the GO command. Unlike the GO command, where program execution waits until all defined moves have terminated, GI allows program execution to continue when the move has begun. This allows for other program defined processes to take place while an axis is moving, such as independent multi–axis moves, OT commands, and conditional IF blocks.

Examples:

VE1 DI20 GI MS1, “Axis #1 is moving” TD2

In this example, when the DI20 move begins, program execution immediately displays the “Axis #1 is moving” message for 2 seconds. When the TD2 command has executed, the program will terminate; however, axis #1 will continue to move until the DI20 distance is reached. A Stop, Kill, or press of the ESC key will halt a GI based move either inside or outside program execution.

The GI command can cause program execution and moves to be asynchronous. To re–synchronize the end of a GI move with program execution, use the Wait (WT) command, i.e. WT#1 will wait for only axis #1. If a program error occurs during a GI move, the move will stop at the Stop Decel Rate.

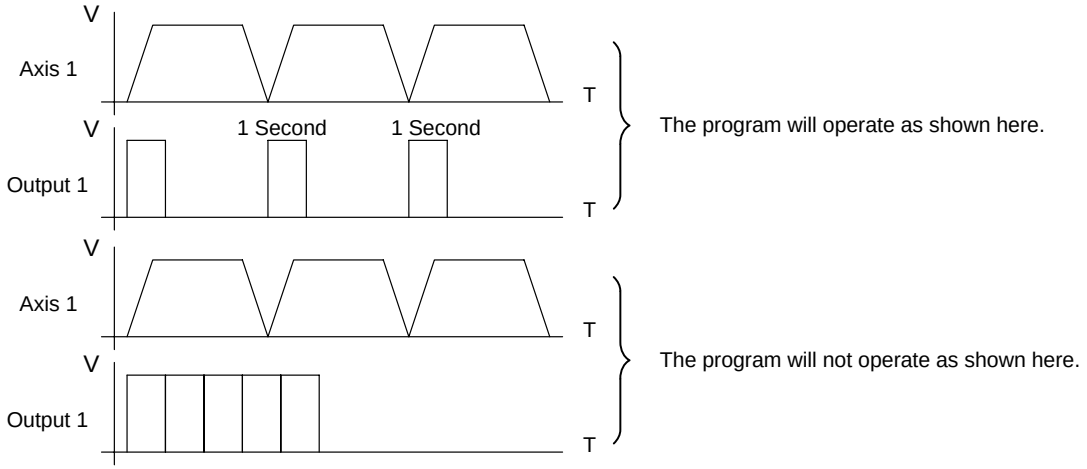
VE1 DA100 GI OT1,1 DA0 GI IF1,1 MS1, “All moves done” EB

If a GI move is in progress and an additional move is commanded on the same axis, the additional move will not begin until the GI move has completed.

In this program, one may expect to see the message “All moves done” immediately after the DA100 move begins. In reality, the program will wait at the DA0 GI until the DA100 move has completed before continuing. More simply stated, a move cannot be commanded to begin on an axis that is already moving.

Since GI allows program execution to continue, there can be programming issues when using GI. For example, in the program fragment “LP VE2 DI10 GI OT1 TD1 OT0 EB” shown in Figure 3-7, after the first pass through, the loop command (LP) will wait at the GI command since subsequent GI moves must wait for the present move to finish.

Figure 3-7



GO Go (Start a Move) syntax – GO or GOi

Value: N/A
Units: N/A
Range: i=1–16

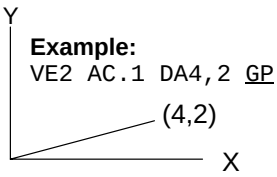
GO executes a move profile defined by some combination of AC, VE, DE, DI, DA, DC, or MC commands. Actual motion of a new profile will occur after a short calculation of the motion trajectory. GOn pre–calculates the move and waits for Input number "n" to activate before executing. This variation is sometimes useful for applications needing very short, repeatable move calculation delays. It is more often used simply to shorten code, since it functions like the combination of Wait on Input and Go (WTn GO) yet it pre–calculates the move. Like other commands using I/O, GOn does not restrict you from using an input even if it has been configured for some predefined function.

Examples:

AC.05 DE.05 VE50 DI5 GO	GO initiates calculation of a move profile using buffered parameters (.05 unit Accel and Decel Ramp, speed 50, 5 unit incremental move) and executes it.
AC.05 DE.05 VE50 DI5 GO2	When input 2 is activated, immediate execution of the motion calculation already in the buffer is performed.

GP Go Point(Start a Move) syntax – GP or GPn

Value: N/A
Units: N/A
Range: N/A



The GP command allows a LinStep+ two axis to execute a linear interpolated move. An example of the GP move is shown. The path velocity, acceleration and deceleration are specified by the parameters traditionally defined for one axis move. These parameters are defined once and each GP move there after will use these values until new values are entered. The end point of the move is specified by a two axis DA or DI command to the appropriate X and Y coordinates.

In the example, the path velocity is 2 user units/sec, path acceleration is .1 sec and the X,Y position is (4,2).

The GPn command specifies a discrete input must be active before executing the move. Although both axes move during a GP move, all GP parameters refer to the path movement rather than the individual axis movements.

Notes:

GP will work with any velocity or acceleration unit.

The largest GP move is restricted to $X2+Y2 <= (231–1)^2$ in units of steps.
For example, the longest simultaneous X,Y point move is:
Steps: `DA1518500249,1518500249`
If resolution is 8000: `DA189,812.5311,189,812.5311`
If resolution is 25000: `DA60,740.0099,60,740.0099`

Commanding moves larger than $X2+Y2 <= (231–1)^2$ will produce unpredictable results.
The DC command does not recognize an interpolated move as a 'single' move and will treat the axes independently. Therefore, using a DC during a GP move will cause unpredictable results unless the user has calculated the necessary values to preserve the vector move.

GS Gosub — syntax – GSi and GS[name]

Value: N/A
Units: N/A
Range: i=1–400, [name] = any legal program name

For use within program only, not for hosted mode.

Jumps to program number or name and returns to the calling program when command processing reaches the EN command in the sub–routine. After the return, execution continues at the command immediately following the GS statement. Subroutines may be nested 16 levels deep. A Goto (GT) clears the subroutine stack, preventing future Gosubs from overflowing the stack or returning to the wrong location.

Example:

DI10 GS[Part A] GO	Run program “Part A”, return and make a 10 unit incremental move.

GT Go to Program — syntax – GTi or GT[name]

Value: N/A
Units: N/A
Range: i=1–199 (1–400 with 30k memory option), [name] = any legal program name

For use within program only, not for hosted mode.

GT branches to the program number or name specified. All subsequent commands in the calling program are ignored. Nested loops and subroutine calls are cleared by a GT command.

Examples:

IF10 GT[PART A] EB	IF input 1 is on and input 2 is off, jump to program “Part A”
IF01 GT20 GT30 EB EN	IF input 1 is off and input 2 is on, run program 20. Program 30 is never run. Use the GS command if you want to return to this program and goto program 30.

IF If

syntax – IF (Mathematical expression)
IFxx (assumes first input is input 1)
IFi,xx ...

Value: N/A
Units: N/A
Range: i=starting input number 1–8
x=0; input high. X=1; input low (grounded). x=anything else; ignore input changes.
expression = any valid expression (see math and variables definitions)

For use within program only, not for hosted mode.

Allows the conditional execution of a block of commands based on the evaluation of an expression or input state. If the expression or input state is TRUE, the commands between the IF and the EB are executed. If FALSE, execution continues with the command following the EB. An IF statement should not be confused with a WT statement. An IF statement evaluates, true or false, based on the conditions at the instant the command is processed. A WT statement pauses command processing until the condition is true.

Note: An End of Block (EB) command must be used with every IF command.

IF blocks can be nested up to 16 levels deep. To increase flexibility (primarily with programmable logic controllers) the IF command allows you to use configured inputs with the command. To help prevent this added flexibility from causing programming confusion, you can specify any character as an input (x). This allows you to self document your IF statements. For example, assume you configure input #3 as a "JOG SPEED" input. Programming like "IF01J10" can help remind you that you are already using input #3 as "JOG SPEED".

Examples:

IF14,1 GO EB	If input 14 equals 1 Go
IF12,010 GO EB	If inputs 12–14 equal 010 Go
IF110 GO OT3,1 EB	If inputs 1–3 equal 110 Go and turn on Output #3
IF(TEMP) > 50 OT1 EB	If temperature variable > 50 turn on Output 1
IF(PARTS)=25 GS20 EB	If PARTS variable = 25 Gosub to Program 20

syntax – IVi,(variable),min,max

IV Input Variable

Value: N/A
Units: N/A
Range: i=1–40 display position characters
variable= any legal variable name
min=the minimum range value (optional); max=the maximum range value (optional)

For use within program only, not for hosted mode.

Allows operator input of variable information under program control. It is usually used with the message command (MS) to prompt for operator input of the variable specified in the IV statement. The cursor is placed at character position “i”. The program waits until a number is entered before continuing execution. You are not allowed to type past the end of either display line. Variables store 4 digits to the right of the decimal place.
When minimum and/or maximum range values are specified, the IV command will not accept inputs outside this range (one of the following messages is displayed on the keypad):

- Input below minimum, Press ESC to resume
- Input above maximum, Press ESC to resume

Variables can be used in a math equation, conditional expression or to set any command parameters (Example: DA, DC, VE, AC, LP, IF, TD, etc.). A variable can be used anywhere in a program a real number or integer is used. Use care when performing math on variables used in LP statements. LP will truncate the non–integer portion of the variable. For example: (COUNT)=25*.2 LP(COUNT) will only loop 4 times because (COUNT)=4.9999. A small offset can be added to variables used in LP statements to avoid this problem. (COUNT)=(COUNT)+.1 will guarantee that (COUNT) will be greater than 5, so the program will loop 5 times.

Examples:

MS1,“”	Clears the Display
MS1,“How many?: ”	Writes string beginning at character 1, top line
IV12,(PIECES),1,15	Waits at 12th character for the # of pieces in the range 1–15.

MS1,“”	Clears the Display
MS1,“How long?: ”	Writes string beginning at character 1, top line
IV12,(LENGTH)	Waits at 12th character for the # of pieces.
LP(PIECES)	Loops the number of pieces entered
DI(LENGTH)	Defines the desired move length/distance.
GO	Moves the length commanded
EB	Ends the loop.

LP Loop

syntax – LPi

Value: 0
Units: N/A
Range: N/A

For use within program only, not for hosted mode.

Causes all commands between LP and EB to be repeated “i” times. If LP is entered without a number following it or a 0, the loop will repeat continuously.
Note: An End of Block (EB) command must be used with every LP command.
Up to 16 nested loops (one inside the other) are allowed. Each LP command must have a corresponding EB command to end the block (loop). A GT command within a loop will terminate the loop, clear the loop stack and jump to a new program.

Example:

AC.09 DE.09 LP3 VE30 DI1 GO EB VE7 DI–3 GO EN
The motor will perform an incremental 1 unit move at speed 30 three times and then a 3 unit move at speed 7 in the other direction.

LU Loop Until Condition True

syntax – LUi,xx ..
LUxx ..
LU(Mathematical expression) or expressions (2)

Value: N/A
Units: N/A
Range: N/A

For use within program only, not for hosted mode.

The Loop Until (LU) command defines a loop block that can only terminate when a condition is true. Up to 16 nested loops (one inside the other) are allowed. LU checks the condition’s value at the end of the loop block. Therefore, the block is always executed at least once, even if the condition is true on the first iteration. (Loop While (LW) defines loops where the conditional is checked at the beginning of the loop.) A GT command within a LU loop will terminate the loop, clear the loop stack and jump to the new program.
i = starting input number, 1–8
x=0; input high. X=1; input low (grounded).
x=anything else; ignore input changes.
expression = any valid expression (see math and variables definitions)
Note: An End of Block (EB) command must be used with every LU command.

Examples:

(A)=0 LU(A)=10 DI10 GO (A)=(A)+1 EB
In this example, the loop is executed 11 times with a final position 110 distance units.

(A)=10 LU(A)<20 DI10 GO EB
In this example, the loop is executed once since the (A)<20 condition is true on the first iteration.

LUXX1X1 MS1,“Inputs 3 & 5 are off” EB GT[Inputs On]
In this example, the loop will continue to execute as long as inputs #3 and #5 are off.

LU4,1 MS1,“Input 4 is off” EB GT[Input On]
In this example, the loop will continue to execute as long as input #4 is off.

LW Loop While Condition True

syntax – LWi,xx ..
LWxx ..
LW(Mathematical expression) or expressions (2)

Value: N/A
Units: N/A
Range: N/A

For use within program only, not for hosted mode.

The Loop While (LW) command defines a loop block that can only terminate when a condition is false. Up to 16 nested loops (one inside the other) are allowed. Loop While loops check the condition's value at the beginning of the loop. Therefore if the condition is false on the first iteration, the block is immediately skipped. A GT command within a LW loop will terminate the loop, clear the loop stack and jump to the new program.
i = starting input number, 1–8
x=0; input high. X=1; input low (grounded).
x=anything else; ignore input changes.
expression = any valid expression (see math and variables definitions)
Note: An End of Block (EB) command must be used with every LW command.

Examples:

(A)=0 LW(A)<=10 DI10 GO (A)=(A)+1 EB
In this example, the loop is executed 11 times with a final position 110 distance units.

(A)=10 LW(A)>20 DI10 GO EB
In this example, the loop is immediately skipped since the (A)>20 condition is false.

LWXX1X1 MS1,"Inputs 3 & 5 are on" EB GT[Inputs Off]
In this example, the loop will continue to execute as long as inputs #3 and #5 are on.

LW4,1 MS1,"Input 4 is on" EB GT[Input Off]
In this example, the loop will continue to execute as long as input #4 is on.

syntax – MC+

MC Move Continuous

Value: N/A
Units: N/A
Range: N/A

Sets move profiles to “continuous move”, using the AC, DE and VE parameters. Move Continuous is enabled on an axis with the “+” sign. “MC+” enables the mode for axis one. DI, DA and DC commands reset the mode to distance.

Each MC+ segment must contain a GO command. Accelerations, Velocities, and Decelerations may be changed in any segment. If no change is specified, the last parameter value will be used. It is not valid to issue positional commands (DI, DA, DC, GH, SP) to an axis while it is moving in continuous mode. Any command is valid within an MC segment except Distance Commands (DA, DC, & DI).

The direction of the move is specified by the sign of the VE parameter. If the sign of the VE parameter changes between two segments, the control will automatically stop the motor (at the programmed deceleration rate) and change directions to the new speed. This makes changing directions based on variable inputs very easy to program using a scaled variable as the VE parameter.

When a MC+ segment is started, it will continue to move at the speed specified by VE until another VE is commanded, the ESC Key is pressed, or an End Of Travel, Kill Motion, Interrupt, or Stop Input is activated. A commanded velocity of zero (VE0) stops a Mode Continuous move. Motion will also stop if the edit, help, copy, or delete menus are opened.

After a continuous move segment has started, command processing will continue when constant velocity is reached. Other commands are then processed sequentially. This allows you to do things like:

- Have asynchronous inputs change the speed of an axis
- Make motion profile changes based on time delays or input states
- Manipulate I/O while moving as a function or time, distance, or input states
- Change speed based on analog inputs or variables
- Have an operator update the speed of an axis through the keypad
- Servo to an analog input
- Make a one axis joystick using analog inputs

If a motor is making a move when it comes to the end of a program, the motor will continue moving, even after the program ends. This allows you to:

- Put different MC moves in different programs and select different speeds by running different programs.
- Change speeds based on Binary or BCD program select lines
- Call MC+ moves as subroutines
- Run from “hosted” RS–232C mode, where the computer commands speed changes
- Run another program from the keypad that does not violate MC syntax. So you could run another program from the keypad to change speeds, manipulate I/O, interface with an operator or calculate arithmetic.

Move Continuous Continued

Examples:

1. Basic Move Continuous syntax. Demonstrates how to change speed and stop MC+ moves based on time delays and input conditions.

MC+	Enable Move Continuous on axis 1
AC.1 DE.2	Set the acceleration and deceleration rates
VE50	Set top speed to 50
GO	Start the Move Continuous move, command processing will continue when axis 1 reaches constant velocity
TD2	Delay for 2 seconds at speed
VE25 GO	Decel to 25
WT111	Wait for inputs 1,2, and 3 to go active
VE0 GO	Stop the move

2. Demonstrates how to prompt an operator for speed changes on a one axis LinStep+. The move is started after the initial velocity prompt. The velocity only changes when the operator enters a new value using the keypad. The move can be stopped by entering a velocity of zero, or when any of the stop conditions defined above exist.

[One Axis MC]	
MS1,“Enter the Velocity”	Prompt the operator
IV23,(V)	Put the operator input in variable (V)
MC+ AC1	
VE(V)	Use operator inputted variable (V) as new speed
GO	Change velocity of axis 1 to the new speed
GT[One Axis MC]	Repeat

3. Demonstrates the use of WT, OT and TD commands in continuous move.

MC+ AC3 VE3 GO	Start first segment
WT8,1 AC.1 VE10	GO Wait for input 8 and change speed
TD5 AC.3 VE.2 GO	Wait for 5 seconds and change speed
WT3,1 VE–10 GO	Wait for input 3 and change speed and direction
OT11	Turn on outputs 1 and 2
TD10 VE0 GO	Wait 10 seconds and stop the move

MS Message to Display

Value: N/A
Units: N/A
Range: n=1–40 characters (20 on each line)

syntax – MS,"" returns to the initial runtime display
MSn,""
MSn,"user text"
MSn,(variable)

For use within program only, not for hosted mode.

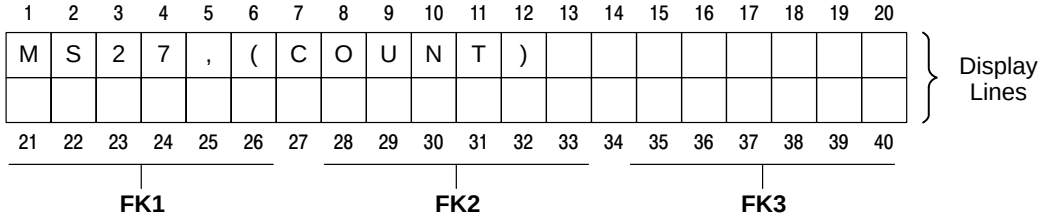
MS allows messages to be displayed on the keypad's display. Messages are usually to prompt for operator input, display function key prompts, or as a diagnostic tool.

MS,"" can be used to restore the initial axis position and I/O display during program execution. MSn,"" clears the display from the n th character to the end of message. MSn,"user text" prints user text beginning at the n th character. MSn,(variable) writes the value of the variable beginning at the n th character. These variations to MS all disable the initial position and I/O display until program execution stops. MS,"" can be used to restore the initial axis position and I/O display during program execution.

Examples:

MS1,""	Clears the Display.
MS3,"Part Count"	Writes string beginning at character 3, top line.
MS27,(COUNT)	Displays the value of the variable (COUNT), beginning at position number 27 (7th character, 2nd row).

This keypad programming template shows the relationship of the each character position to the location of the function keys. This will help you align messages on the display. A larger version is located in the back of this manual and can be photo copied for programming use.



ON On Condition (On Event)

syntax – ONn,GTx
ONn,GSx
ONn,0 Clear the event

n= On EOT Limit

Value: N/A
Units: N/A
Range: N/A

Allows conditional program execution based on an event. When the programmable event occurs, the current program and move are interrupted and program execution begins at the predefined interrupt program. The interrupt program can be defined as a GT or a GS. Defining the interrupt as a GS allows program execution to return to the exact point in the original program where the interrupt occurred. The ON command currently supports End–of–Travel (EOT) as an event conditional.
After an ON event is defined, it is active in all user programs and need not be redefined except to change the interrupt program type, number or name, or clear the event condition.

Example: Using ON to handle an EOT event.

```
[POWERUP]
ONL,GT[ON EOT]          [Goto [ON EOT] {on an End–of–Travel input}
GT[HOME]                {Home the machine}

[MAIN]
LP                      {Loop forever}
   VE5 DA20 GO          {Define move}
   DA0 GO               {Define move}
EB

[ON EOT]
IF(SA1)&256             {Check if EOT– switch was hit (bit #9 in SA)}
   VEI DA0.5 GO         {Move off EOT– switch}
   GT[HOME]             {Home the machine}
EB
VEI DA–0.5 GO           {Move off EOT+ switch}
GT[HOME]                {Home the machine}

[HOME]
GH3                     {Home}
GT[MAIN]                {Jump to main loop}
```

OT Output (turn outputs On/Off)

syntax – OTi,xx ..
OTxx (assumes first input is input 1)
i= starting output number 1–16
x=0; input high. X=1; input low (grounded).
x=anything else; ignore input changes.

Value: N/A
Units: N/A
Range: i=1 to 16

Sets both discrete and digital Opto output states. After an output is turned on (low), it remains on until changed by another output command, a reset input (software warm–boot), or power is cycled. All outputs are turned off (high) at power up or during a reset.

For flexibility, the OT command allows you to use configured outputs anytime. To help prevent programming confusion, you can use any character in the “don’t change” section of your output statement. This allows you to self–document your OT statements. For example, assume you configured output #3 as a “FAULT” output. Programming like “OT01F10” can help remind you that you are already using output #3.

Example:

OT4,1	Turn on Output 4.
OT2,0D1	Turn Outputs 2 off, leave 3 as is, and turn 4 on.
OT110	Turn Outputs 1 and 2 on, and 3 off.

“” Quote

syntax – Any ASCII Character

Value: N/A
Units: N/A
Range: N/A

The “” command transmits an ASCII string from the serial communications interface. A “” without any string transmits a carriage return character (ASCII 13).

Example:

“Move Complete”	Transmits string only out serial interface.
“”	Transmits a carriage return only.

syntax – RGr

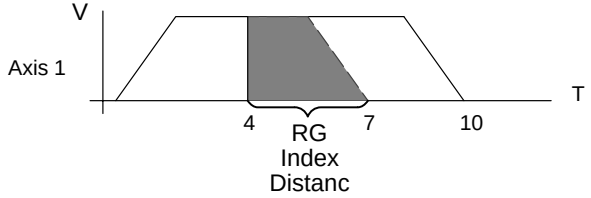
RG Registration

Value: N/A
Units: N/A
Range: N/A

The Registration command (RG) specifies a distance to be moved from the current position – as commanded by a specific input trigger. For example, in the following program of 10 user–units on axis #1, the input trigger is received at user–unit 4, to move 3 user–units from the point where the input trigger was received.

VE2 AC.1 DA10 RG3 GO

Assume the input is an optical sensor that triggered on a registration mark at a position of 4 user–units. The commanded move related to the registration move is as follows:



T=4 RG Input Trigger
T=7 RG End of Move
T=10 Ends at Commanded Position

Accompanying the programmable Registration Command is the configurable Registration Input: G (also G in Serial Setup Commands). To configure a Registration input from the keypad, choose EDIT > SETUP > I/O > INPUTS. An input configured as a Registration Input is designated by a G on the keypad input status display. The RG Command will only function if the corresponding input has been configured as a Registration Input.

Note: Registration Input is only configurable on Input #1 for Axis #1.

syntax – SP±r

SP Set Position

Value: 0
Units: set in EDIT > SETUP > MECH
Range: varies with distance units

Sets the current absolute position to “n”. This command is typically used to adjust or shift a coordinate system. It is often done after a series of incremental moves to reset the absolute coordinate frame.

Example:

MC+ GO WT1,1 VE0 GO SP10.5 After the move is complete, sets the current position of axis 1 to 10.5.

syntax – SQr,(var)

SQ Square Root

Value: N/A
Units: N/A
Range: 0.0001 – 214748.3645

The SQ command calculates the square root of a number and returns the result in a user defined variable. The n parameter in the syntax can be a number or a variable parameter, however, the second parameter must be a previously defined variable for which the square root result is stored. If the second parameter is not a defined variable, you will get a Bad Variable Name error. Following mathematical convention, SQ will produce an Invalid Parameter error for negative "r" values. The return value is accurate to the 0.01 place.

Example: This program calculates the square root of 27.96 and stores the value in the user defined variable (SQRESULT).

(SQRESULT)=0 SQ27.96,(SQRESULT) The returned value in (SQRESULT) is 5.28.

syntax – STn (or ST#n)

ST Stop Move on Input

Value: N/A
Units: N/A
Range: 0–8 inputs

ST stops move execution upon activating the input specified by n.
ST0 disables (turns off) the STn command.
ST#1 stops the move on axis #1. (Allows program command conditional program termination).

After ST is executed, the specified input is monitored during every "move profile". If the input is activated, the current "move in progress" is terminated, stopping all motion until the input is deactivated or an ST0 is processed. LinStep+ will process and calculate commands, but will wait at the next GO command until the ST input changes.

The motor is stopped at the deceleration rate specified in the Stop Decel Rate parameter. Once issued, Stop on Input remains active until it is turned off by the ST0 command, a reset is issued, or power is cycled.

Example:
ST1 AC1 DE1 VE25 DA6 GO VE50 DA0 GO EN
Move to absolute position 6 units. If input 1 is activated while moving, Stop Motion. When the input is deactivated, immediately execute the next move profile which is to move to the absolute zero position. If input 1 is not activated while moving, the motor would complete its 6 unit move before executing the move back to absolute zero.

syntax – TDr

TD Time Delay

Value: N/A
Units: seconds
Range: r=0.01 – 99999.99

Delay r seconds before executing the next command.

Example:
VE50 DI4 GO OT11 TD.5 OT00
Move 4 units, turn outputs 1 and 2 on, delay .5 seconds, and turn outputs 1 and 2 off.

syntax – VEr

VE Velocity

Value:
Units: in/sec set in EDIT > SETUP > MECH > VEL
Range: varies with velocity units

Sets the maximum velocity during a move profile. If the acceleration rate is too slow or the move distance is too short, the motor may make a triangular move (velocity vs. time) and the motor may never reach the specified speed. When VE is specified, the value is used in all subsequent moves until re–defined.

Example:

AC.1 DE.2 VE50 DA4 GO Move to absolute position 4 units with a top speed of 50 in/sec.

WT Wait

syntax – WTi,xx...
WTxx... (assumes first input is input 1)
WTexpression
WT#1, (synchronized the program after a GI move)

Value: N/A
Units: N/A
Range: 1–16

For use within program only, not for hosted mode.

Causes LinStep+ to wait for the specified condition to be true before continuing program execution. Either digital or analog input conditions may be used. The WT command allows use of configured inputs in the expression. To help prevent this from causing programming confusion, you can use any character as an input (x). This allows you to self document WT statements. For example, assume input #3 is set to “JOG SPEED”. Programming like “WT01J10” can help remind you that you are already using input # 3.
i = starting input number, 1–16
x=0; input high. X=1; input low (grounded).
x=anything else; ignore input changes.
expression = any valid expression (see math and variables definitions)

Example:

WT4,1 GO Wait for input 4 to equal 1 before moving.
WT2,010 GO Wait for inputs 2–4 to equal 010 before moving.
WT110 GO Wait for inputs 1–3 to equal 110 before moving.
WT#1 Causes program execution to halt (wait) until GI move is complete.

Table 3-2 Summary of Expressions, Operators and Functions

[ ]	Name Program
( )	Name Variable
&&	Logical AND
||	Logical OR
!	Logical NOT
!=	Not Equal
+	Add
–	Subtract
*	Multiply
/	Divide
=	Equal
>	Greater Than
>=	Greater Than or equal to
<	Less Than
<=	Less Than or equal to
&	Bitwise Boolean AND
|	Bitwise Boolean OR
++	Increment Variable
+=	Increment by n
– –	Decrement Variable
–=	Decrement by n
<<	Shift Left
>>	Shift Right

Helpful Hints

Programming your application

This section provides additional information that may be helpful to begin programming the LinStep+. Also, several practical examples are given that can be used or modified. More program examples are available on the Intelliware disk set.

Programming Overview

First, you must decide how the LinStep+ fits into the overall machine control hierarchy. Generally there are three ways that the LinStep+ may be used:

1. Stand–alone mode – controls all the Inputs/Outputs and motion.
2. With a PLC – the PLC runs the machine and calls on the LinStep+ via program select lines for motion.
3. In a “hosted” mode – the PC sends serial commands to the LinStep+ for execution. The LinStep+ uses a sequential, interpretive command processor. This means that commands in a program are executed one at a time, and that one command must be completed before the next command is processed.

`[Move] VE4 DI10 OT01 GO OT10` **Example of "Hosted" Mode Program**

In the program [Move], the maximum move velocity is set to 4, the command incremental distance is set to 10, output 1 and output 2 are turned off and on simultaneously, axis one then moves 10 units. After axis one stops moving, output 1 is turned on and output 2 is turned off. These changes of outputs 1 and 2 occur at the same time.

The flow of the program is controlled with WT (wait for an event or condition to occur), TD (wait for a pre–set amount of time to elapse), and IF (if a certain condition is true at this instant, then execute a block of commands) statements. External controllers such as PLC and computers can be coordinated by the digital outputs and ASCII serial commands. The commands that can be entered from the keypad and used in a program are listed in Table NO TAG.

Variables

Memory space allows for up to 100 variables. All variables are stored as fixed point numbers. All variables are global. All standard variables are volatile, though non–volatile variables are available as well. Variables can be used in many parts of the program, such as:

- Arithmetic
- Conditional Expressions
- Loop Counts
- Distance and velocity commands
- Set values
- Set command values or parameters
- Set analog signals
- Read analog or temperature input
- Display information such as position or velocity
- Any place that a number can be used, a variable can be used

Variable Names

Descriptive variable names can be assigned, instead of V1, V2, etc. Variable names can be up to 14 characters, but the first 10 characters must be unique. A name can contain other printable ASCII characters, such as numbers, underscores, exclamation points, even spaces. Upper and lower case characters are supported within variable names, and these variable names are case sensitive. ASCII control characters such as LF and CR are not supported. All variables must be enclosed in parentheses, (variable name). Parentheses are not legal variable characters.

Built–in Variables

Some variable names are pre–defined. They can be used in expressions, to set voltages, to test conditions, or to display information to the keypad display or an external serial device.

Variable Name	Description of Built–in Variable	Type
(AI1) to (AI6)	Analog Inputs 1 through 6	Read Only
(AROWREL)	Current status of any of the four arrow keys	Read Only
(CPOS1) (CPOS2)	Commanded position of axis 1 or 2	Read Only
(EPOS1) (EPOS2)	Encoder position of axis 1 or 2	Read Only
(VEL1) (VEL2)	Commanded velocity of axis 1 or 2	Read Only
(POS1) (POS2)	Current Position of axis 1 or 2	Read Only
(#F1) thru (#F50)	Non–volatile, limited use, user system variables	Read/Limited Write
(FKEY)	Value of Function Key pressed	Read Only
(LASTKEY)	Value of last Function key pressed	Read Only
(TERM)	Sends variable out RS–232 port	Write Only
(1TW)	Scans inputs 1–4 for BCD Digit	Read Only
(2TW)	Scans inputs 1–8 for BCD Digit	Read Only
(TIME)	Elapsed Time (ms) since power up or since reset	Read Only
(CRCS)	Value of the EEPROM setup checksum	Read Only
(CRCP)	Value of the EEPROM program checksum	Read Only
(SA1) (SA2)	Integer value of the status of axis 1 or 2 (SeeRS232 command SA)	Read Only
(SD1) (SD2)	Integer value of the drive status of axis 1 or 2 (See RS232 command SD)	Read Only
(SS)	Integer value of the system status (See RS232 command SS)	Read Only
(INT98CTRL)	Enables/disables (ARM INT98) trigger option	Read and Write
(ARM INT98)	Enables/disables INT98 input if (INT98CTRL) is enabled	Read and Write

Examples of how to use Built–in Variables

(PIECES)=10	Assigns 10 to variable
(SPEED)=(AI12)*(VEL SCALE)	Speed = analog input times a scalar
MS21,“Enter Length” IV32,(LENGTH)	Prompts user and gets feed length
VE(SPEED)	Sets velocity to value in variable
MS1,(POS2)	Displays current position of axis 2 on keypad screen
(TERM)=(POS1)	Sends the current position of axis 1 out the RS–232 port
(TEMPERATURE)=(AI9)	Reads in temperature from an analog input
(AO15)=4012	Sets the analog output to 4012
(#F1)=(PIECES)	Stores the value of Pieces in FLASH memory variable #F1

Using Built–in Variable (AROWREL)

(AROWREL) is a built–in Boolean read only variable that determines the status of any of the four arrow keys. When used with (FKEY), the program can detect if an arrow key is being held down. (AROWREL) will only return the status of the four arrow keys. If a different key is pressed, (AROWREL) will return the value 0. (AROWREL) will return one of these values:

(AROWREL)=0 One of the arrow keys is being held down.

(AROWREL)=1 The arrow key has been released.

Example JOG application using (AROWREL) and (FKEY):

```
[MAIN]                      {Program #1}
FK12,13                     {Wait for a Left or Right arrow key}
GT(FKEY)                    {Jump to arrow key program #12 or #13}

[LEFT ARROW]                {Program #12}
MC+                         {Enable MC mode}
AC.1                        {Start MC move}
VE1                         {Move in positive direction}
GO
LP
    IF(AROWREL)=1           {Check status of arrow key}
        VE0                 {Stop MC move on key release}
        GO
        GT1                 {Return to main program}
    EB
EB

[RIGHT ARROW]               {Program #13}
MC+                         {Enable MC mode}
AC.1                        {Start MC move}
VE–1                        {Move in negative direction}
GO
LP
    IF(AROWREL)=1           {Check status of arrow key}
        VE0                 {Stop MC move on key release}
        GO
        GT1                 {Return to main program}
    EB
EB
```

Non–Volatile Variables

(#F1) through (#F50) are fifty user variables stored in non–volatile flash memory so they retain their values through power cycles, warm boots, and system resets. Standard user variable are lost at power down or reset. When one of these variables is changed (i.e. used on the left side of a equal (=) sign, the new value is written to, and stored in the user non–volatile flash.

Note: Flash memory has a limited read/ write lifetime (100,000 writes before failure), variable values that change frequently should not be stored as these variables. Examples include loop count variables, and POS1 and POS 2 variables. LinStep+ will allow only 1,000 FLASH writes between power cycles. This limit is set to prevent damage to non–volatile memory due to a simple programming mistake or misunderstanding. When this write limit is exceeded, all programs will stop running, an error message will be displayed, and the appropriate status bits will be set.

Example: At the start of each part run, a program called [Set–up] is used to initialize a number of variable part parameters. During production the program called [PARTS] is run. This program reads from the FLASH variables, but does not generate any writes to the FLASH, so the lifetime of the FLASH is not compromised.

[Set–up]	{Program #1}
MS1,“Feed length?: ”	Writes string beginning at character 1, top line
IV12,(LENGTH),1,15	Loads the part length to variable (LENGTH)
MS1,“Feed Speed?: ”	Writes string beginning at character 1, top line
IV12,(SPEED),.05, 5	Loads the speed into volatile variable (SPEED)
(#F1)=(LENGTH)	Loads the length into non–volatile variable (#F1)
(#F2)=(SPEED)	Loads the speed into non–volatile variable (#F2)
EN	
[PARTS]	[PARTS] runs on power up, unless new parameters are entered.
(LENGTH)=(#F1)	Load part specific variable from non–volatile #F1.
(SPEED)=(#F2)	Load part specific variable from non–volatile #F2.
LP(NUMBER)	Loop (NUMBER) of times
DI(LENGTH)	Move (LENGTH)
VE(SPEED)	at (SPEED) velocity
GO	
OT1 TD.1 OT0	Change output to indicate part done
EB	End the loop Block

Arithmetic Operands and Equations

Addition (+), subtraction (–), multiplication (*), and division (/) are easily performed. Expressions may only contain one operand. Complex equations require multiple statements. Variables and fixed point numbers may be mixed in arithmetic equations. All user arithmetic and variable storage uses 32 bit integer and fractional representation. The + and – symbols have a dedicated button on the keypad. Pressing the button will change between the two. The *, /, and = are accessed with the Alpha+0+ ... keystrokes.

Example:

```
(X)=(Y)*10
(AO15=(VOLTAGE) + (ERROR)
```

Examples of incorrect use:

(X)=1+2–3	This statement is incorrect because it has more than one operand.
(Length)=(Total)*0.3125	This statement is incorrect because it has more than four decimal places.

Instead, you should use:

(X)=1	
(X)=(X)+2	or (X)+=2
(X)=(X)–3	or (X)–=3

and

(Length)=(Total)*3.125	Multiply the significant figures
(Length)=(Length)*.01	Then move the decimal place
or	
(Length)=(Total)/32	32 bit storage of fractional decimal number

Boolean Operators

Boolean operators & and | perform their respective bitwise Boolean functions on immediate or variable parameters. As an example, isolate a specific bit from an SD response to determine if axis #1 drive is enabled. This corresponds to a bit #5 (10000) Binary, (16) Integer in the SD response. The example program segment is written as follows:

```
(DRIVE STAT)=(SD1)&16 IF(DRIVE STAT)=16 MS,1“Drive Enabled” EB
```

The 16 corresponds to an integer weight of bit #5 (10000).

Logical Operators

Conditional commands (IF,WT, LU, LW) support logical operations of AND (&&) and OR (||). Two expressions may be logically AND'd or OR'd within one conditional command. For example:

(A)=5 (B)=2.5 IF(A)>2&&(B)=2.5 MS1, "True Statement" EB

In this program, the message "True Statement" appears since BOTH conditional statements are true.

Increment/Decrement Variables

Four syntaxes are supported by variables: ++ (Single Increment), += (Value Increment), — (Single Decrement), –= (Value Decrement). These operators will initialize any uninitialized variable to zero before incrementing or decrementing it for the first time.

(Variable Name)++	Increments a variable value by 1
(Variable Name)+=n	Increments a variable value by n
(Variable Name)– –	Decrements a variable value by 1
(Variable Name)–=n	Decrements a variable value by n

Expressions

Five conditional expressions are supported: less than (<), equal to (=), greater than (>), less than or equal to (<=), and greater than or equal to (>=). The IF and WT commands can use these expressions to direct program flow or wait for an analog input to meet a condition. The > and < symbols are entered into the keypad editor with the ALPHA+↑+↑...↑

Examples:

IF (X)>10 GS20 EB	If X is greater than 10 gosub to program #20.
WT(AI12)<(MAX TEMP)	Wait for the temperature to go below the maximum before continuing command processing.

Other Programming Samples

These sample programs provide an idea of how to solve simple programming tasks. To aid your program documentation, comments are placed in {brackets}. These comments are stripped from the program as it is downloaded for execution to help conserve memory. Files should be saved before downloading save the comments.

Example:

DI10,2 GO	Moves to load position
DI15,15 GO	Moves to unload position

Create a Message and Read an Input Variable

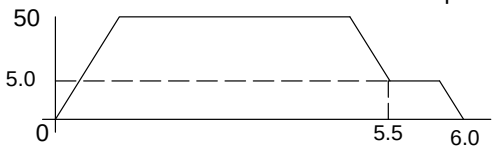
[GET PARTS]	Name the subroutine
MS1,“”	Clears the Display
MS1,“How many?: ”	Writes string beginning at character 1, top line
IV12,(PIECES)	Waits at 12th character for the # of pieces.
MS1,“”	Clears the Display
MS1,“How long?:: ”	Writes string beginning at character 1, top line
IV12,(LENGTH)	Waits at 12th character for the length.
LP(PIECES)	Loops the number of pieces entered
DI(LENGTH)	Moves the length entered.
GO EB	

Create a Menu (menu display on keypad display for operator)

MS1,“”	Clears keypad screen
MS21,“PART1 PART2 PART3”	Writes a message above function keys.
FK1,2,3	Waits for a Function Key to be pressed
(FKEY)=(FKEY)+50	Add an offset to FKEY
GT(FKEY)	Goto program #51, 52, or 53. (50 + 1, 2, or 3)

Fast In, Slow Feed Move (Using the Distance to Change (DC) command)

AC.05	Set acceleration
DE.09	Set deceleration
VE50	Set first velocity
DA6	Set total move distance
DC5.5	Set point where you want to change speed
VE5	Set second speed
GO	Start the move profile



Setting an Output=On (on–the–fly)

AC.05	Set acceleration
VE10	Set velocity
DA4	Set total move distance
DC1	Set point to turn on...
OT1	Output 1
DC2	Set point to turn on...
OT2,1	Output 2
DC3	Set point to turn on...
OT3,1	Output 3
GO	

Read a 4 Digit BCD number, 2 Digits at a time

[GET 4 BCDS]	Returns value of 4 digit BCD number
OT01	Connect ground of first two BCD digits
(4 DIGIT BCD)=(2TW)*100	Make value of first two digits the MSB
OT10	Connect ground of 2nd two BCD digits
(4 DIGIT BCD)=(4 DIGIT BCD)+(2TW)	Add value of 2nd two to 1st two * 100

Reading an Analog Input Value

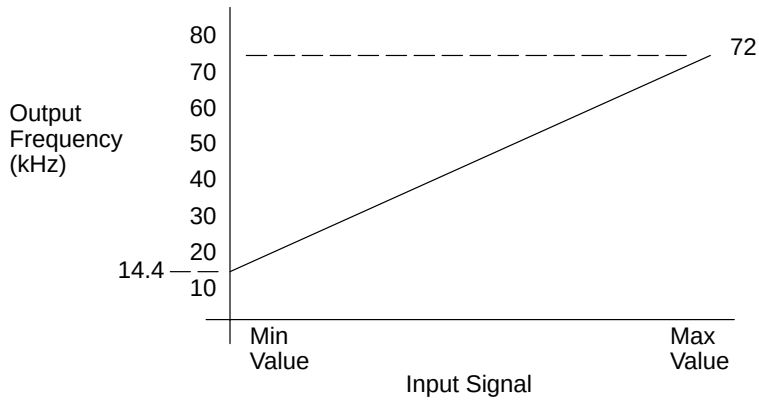
The value of the analog system variables (Al I –AI6) are scaled from 14,400 to 72,000 Hz. This value is actually a scaled frequency read from the OPTO module representing the analog signal. These input values are updated every 16 milliseconds. If your program needs to display this value in units such as VOLTS, you will need to scale the value to VOLTS in your program. The scaling factor depends upon the type of OPTO module used. For example: a ”J ” thermocouple uses a different factor than a ”K” thermocouple. Due to slight variances in the output frequency from module to module, it is recommended that the OPTO be calibrated by querying the corresponding AIx value with no input signal connected to the OPTO. This value should be used as the zero input reference frequency.

Example: Use a 0–10VDC analog input. 0V=14,400; 10V=72,000 or 5.760 Hz/volt.

(VOLT)=(AI2)	Read the value of analog input #2 into variable volts
(VOLT)=(VOLT)–14400	Remove frequency offset
(VOLT)=(VOLT)*1.736	Scaling factor multiplied. by 10,000
(VOLT=(VOLT)*.0001	Scaling back to volts

The variable (VOLT) is now in units of volts. If you are waiting for a condition to occur or doing a comparison, (see below) there is no need to go through the conversion process.

(TEMP)=(AI9)	Read in temperature from analog input
WT(AI12)<45000 GO	Wait for analog input 12 <45000 (<5.3 VDC using the previous example) before moving
IF(AI12)<45000 GO EB	Go if analog input 12 < 45000



Section 4
Serial Communications

Programming with Serial Communications

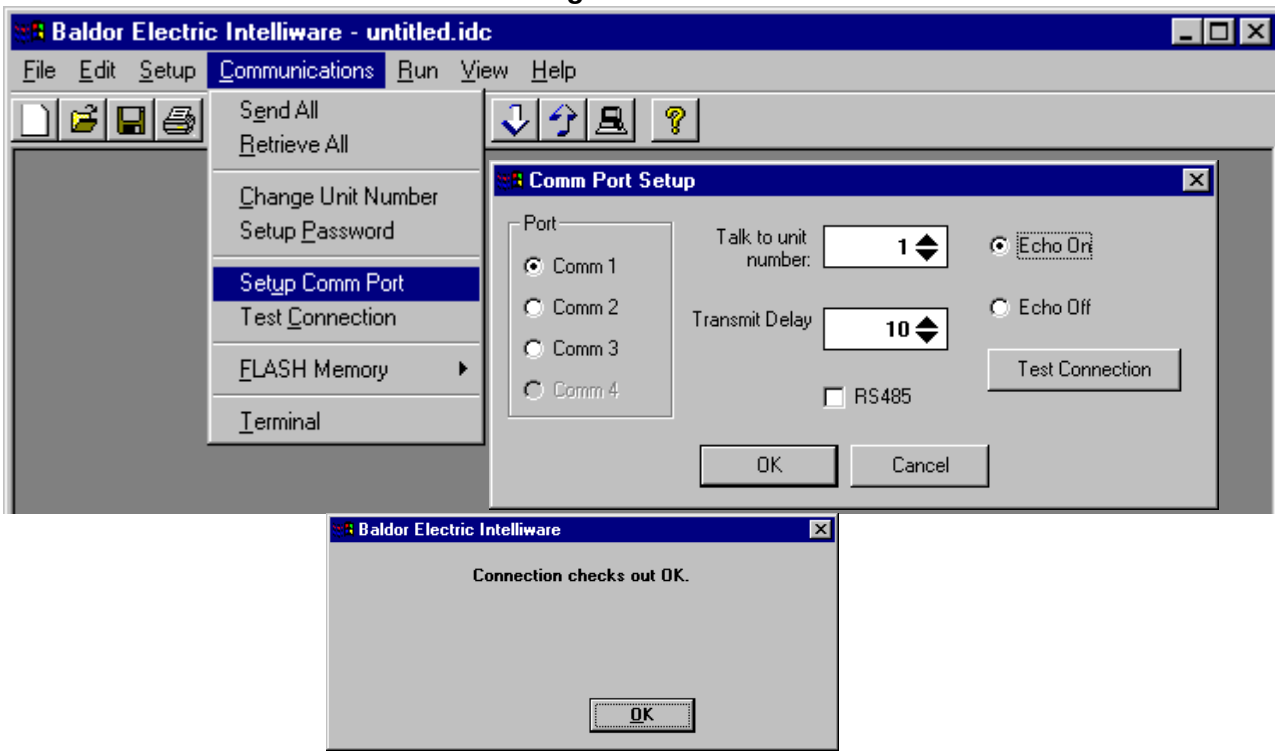
An RS–232C terminal, PC, or RS–232C–equipped PLC can be used to configure, program, and operate the LinStep+. Baldor strongly recommends the use of the optional Windows–based Intelliware software tools for configuration and programming. If you choose not to use Intelliware, all of the serial programming and setup commands are listed alphabetically in this appendix (see Command Reference).

Intelliware provides a graphical control configuration environment, a program development editor, and a terminal communication package. Intelliware also provides application upload and download utilities, and an on–line help utility.

RS232 Communication

Be sure the RS232 connections are made as described in Section 3 of this manual. Install Intelliware (or you serial communications software if you are not using Intelliware). Configure the COM port and set Echo ON. Test the connection.

Figure 4-1



A configuration file is edited and the values are saved or downloaded at run time. A configuration file has an extension of ".idc". An example configuration file is shown in Figure 4-1.

Figure 4-1 Partial Configuration File Example

```
{ Baldor Electric Intelliware, v1.00 }
{ untitled.idc, 04-03-2001, 08:08:31 }

. . .

{Setup Parameters}
MT:10                {Motor type: Stepper}
MR:10                {Motor resolution: 36000}
AU:0                 {Acceleration units: USER units/sec¿
VU:0                 {Velocity units: USER units/sec}
DU:2                 {Distance units: inches}
GR:1:2               {Gear ratio}
OP:IIIIIIII          {OPTO definition: I=Input}
ID:UUUUUUUU          {Input definition: U=Unassigned}
OD:PPPPPPPP          {Output definition: P=Programmable}
OE:P,00000000        {Output state on power-up}
OE:F,XXXXXXXX        {Output state on fault}
OE:S,XXXXXXXX        {Output state on stop}
ET:1                 {End of travel polarity}
JA:50                {Jog acceleration}
JL:2                 {Jog-low velocity}
JH:5                 {Jog-high velocity}
JE:1                 {Jog enable}
HE:0                 {Home edge}
HF:1                 {Home final direction}
HS:1                 {Home level}
HO:0                 {Home Offset}
PU:0                 {Power up program: None}
SN:1111111           {Stop scan on-settings: }
DY:100               {Scan delay (ms)}
EM:0                 {Encoder mode: Open loop}
ER:5000              {Encoder resolution}
FE:1500              {Following error}
IR:25                {In Range}
IT:-10               {In Range Time}
HM:0                 {Homing Method}
PG:10                {PM Gain}
PV:1                 {PM Vmax}
BK:0                 {Backlash}
DF:1,0,4,5           {Display format: }
SR:500               {Stop deceleration rate}
MD:0                 {Motor direction}
MV:50                {Maximum velocity}
AM:500               {Accel limit}
AH:0                 {Anti hunt}
AW:0                 {Anti-inertial windup}
FA:0                 {Accel feed-forward gain}
FV:0                 {Velocity feed-forward gain}
KI:0                 {Integral gain factor}
KP:0                 {Position gain factor}
KV:0                 {Velocity gain factor}
EL:0                 {Motor enable polarity}
FL:1                 {Motor fault polarity}
BV:5                 {Clamp break velocity}
CU:0                 {Clamp current units}
AR:0                 {Anti-Res}
MH:L                 {Motor inductance}
MI:0                 {Motor current}
WA:0                 {Waveform}
RE:0                 {Rest 0:disabled 1:enabled}
IL:0                 {Idle 0:disabled 1:enabled}
WC:0                 {}
WG:0                 {}
WP:0                 {}
```

Command Syntax
All commands use two letter UPPER CASE ASCII characters. Command delimiters can be a carriage return (<cr>) or space (<sp>) character. A brief command listing is intended to help programmers with command syntax. Table 4-1 describes the abbreviations and command syntax definitions.

Table 4-1

Letter or Symbol	Description
<n>	Unit address number is optional with RS–232C, and the command is sent to all units if no address is specified. All Status commands require an address. RS–485 users must address all commands.
,	Field separator.
a	Alphabetic character.
h	Hexadecimal number.
i	Decimal integer number.
r	Decimal real number (up to 4 places to the right of the decimal).
:	A colon (:) is a neutral character. It can be used in a command to make it more readable to the programmer. For example OP:OOOOIIII is easier to understand than OPOOOOIIII. Note: The colon is not neutral, and is required in GR (Unit Ratio) command. Example GR4:1

***Asterisk** – Some commands request a response from the control. Responses will always be preceded by an asterisk (*) which notifies the other controls on a daisy chain to ignore the subsequent response characters preceding the next command delimiter. For example, the Input State (IS) command might return *AF09<cr>. Your program will need to mask the asterisk before decoding the value returned.

{Brackets} – You can document your programs by placing comments between brackets. For example: {this is a comment}. To maximize program storage space, the control strips off these comments when a program file is downloaded.

Serial Setup Commands

This section provides an alphabetical listing and a detailed description of each command. Commands may be sent for execution over an RS232 serial connection or written to a program file and downloaded to the LinStep+ non–volatile memory for later execution.

(This category of commands is identical to the equivalent keypad commands described in the LinStep+ manual.)

AM Acceleration Maximum — syntax – <n>AMr

Value: 0.002 sec
Range: Unit scaling dependent

Sets the maximum acceleration and deceleration limit for programmed move profiles. Programmed accelerations and decelerations for moves will be limited by this parameter (like VMAX for velocity).

AR Anti–Resonance — syntax – <n>ARi

Value: N/A
Range: N/A

Sets the anti–resonance value.

AU Acceleration Units — syntax – <n>AUi

Value: N/A
Range: 0=Units/Second (units set by DU), 1=ips^2, 2=seconds

Sets the acceleration units.
Example: AU0 {axis one Units/sec^2, Units defined by DU}

DF Display Format — syntax – <n>DFi,i,i,i
<n>DF“text”,i,i,i,i

Value: N/A
Range: 0=Units/Second (units set by DU), 1=ips^2, 2=seconds

Configures the four run time display quadrants of the keypad display. DF takes four parameters where i is an integer representing a display data type per quadrant. User defined text is limited to 10 characters per field.

i=0	Blank	i=8	Not used
i=1	POS1	i=9	Inputs
i=2	POS2	i=10	Outputs
i=3	POS1+Unit	i=11	Opto’s
i=4	POS2+Unit	i=12	SA–Status1
i=5	VEL1	i=13	SA–Status2
i=6	VEL2	i=14	SA–Status
i=7	Not used	i=“ ”	User text in quotes

DU Distance Unit Label syntax – <n>DUi

Value: N/A
Range: 0=Units/Second (units set by DU), 1=ips^2, 2=seconds

Configures the four run time display quadrants of the keypad display. DF takes four parameters where i is an integer representing a display data type per quadrant. User defined text is limited to 10 characters per field.

i=0	steps	i=9=	degrees
i=1	rev	i=10	radian
i=2	inch	i=11	grad
i=3	mil	i=12	arcsec
i=4	meter	i=13	arcmin
i=5	cm	i=14	percent
i=6	mm	i=15	index
i=7	yard	i=16	□m
i=8	foot		

DY Scan Delay syntax – <n>DYi

Value: 100ms
Range: 2ms

Sets the amount of time required for the program select inputs (BCD or Binary) to remain stable before they are valid. The minimum time is 2 ms. If program select inputs are not stable for a time equal to or greater than the specified delay, the program will not be executed. Use the numeric keys to enter a value in ms.
Note: See Data Valid Input Configuration for an alternate approach.
Example: DY500 {500ms delay}

EL Enable Line Polarity syntax – <n>EL0

Value: Active Low
Range: N/A

This parameter value is set to Active Low and cannot be changed.

EM Encoder Mode syntax – <n>EMi

Value: Open Loop
Range: Open Loop, Open–Stall, Closed Loop, Servo–Closed Loop, Closed Loop–PM

0 Open Loop The OPEN LOOP position will be displayed on the keypad.
1 Open–stall The OPEN LOOP position will be displayed on the keypad, and the encoder will be used for stall detection. (See Following Error)
2 Closed Loop The actual encoder position is displayed on the screen. All subsequent moves are calculated from this actual position. All moves are based on encoder pulses. Stall detection is enabled. Positioning resolution will equal the resolution of your encoder.
3 Servo–closed Displays actual encoder position, but moves are based only on LOOP commanded OPEN LOOP position. Stalls are detected in this mode.
4 Closed Loop–PM Functionally identical to CLOSED LOOP, with the addition of post–move position maintenance of the last commanded position. Provides “pseudo– servo” operation to stepper systems.

Use PM GAIN, PM VMAX, and In–range Window parameters to specify position maintenance tuning parameters.

Application Notes:
Following–error is still active while in CLOSED LOOP–PM mode. A following–error will occur when the number of correction steps exceeds the following error value. This allows the unit to signal a fault when the displacement cannot be corrected, i.e. an obstruction.
Closed Loop–PM will not attempt to correct position while navigating menus with the keypad.
Example: EM2 {axis one Closed Loop}

ER Encoder Resolution syntax – <n>ERi

Value: 2000 pulses/in
Range: 0–36000

This option sets the encoder resolution. The resolution is specified in encoder pulses per in of travel, post–quadrature. To prevent end–of–move dither, an encoder resolution of 8000 or less is recommended.
Example: ER2000 {axis one 2000 counts/in}

ET End of Travel Switch Polarity syntax – <n>ETi

Value: NORM CLOSED
Range: NORM OPEN, NORM CLOSED

Allows selection of Normally Open or Normally Closed polarity of the End of Travel switch (EOT).
0 Normally Open
1 Normally Closed

FE Following Error Limit — syntax – <n>FEi

Value: 750 Motor Steps
Range: 0–999,999 motor steps (0 = Off)

If a Following Error occurs, the control will enter a fault state where:

- Any motion or program being executed is immediately terminated.
- The LCD Display will indicate “Following Error”, along with an explanation.
- A fault output will be generated if defined as a “Stall” or Fault output.
- The fault must be cleared before motion can occur. A Stop or Kill, by programmable inputs or serial command, the ESC key or a RESET will clear a Following Error fault
- Bit 9 of SS response is set to 1
- Bit 1 of SD response is set to 1

Example: FE1000 {axis one 1000 steps}

FL Fault Line Polarity — syntax – <n>FL0

Value: Active Low
Range: N/A

This parameter value is set to Active Low and cannot be changed.

HE Home Edge — syntax – <n>HEi

Value: NEGATIVE
Range: NEGATIVE, POSITIVE

Sets the home switch active on the negative edge or positive edge of the encoder index channel.

HF Home Final Driection syntax – <n>HFi

Value: NEGATIVE
Range: NEGATIVE, POSITIVE

Sets the direction for the Go Home (GH) move. This is the direction used to search for the encoder index mark (Z channel) after the appropriate home switch edge is found.

0 Negative Edge
1 Positive Edge

Example: HF1 {axis one positive direction}

HM Homing Mode syntax – <n>HMi

Value: Switch Only
Range: Switch Only, Switch Then Z Channel, Z Channel Only

Sets how a Go Home (GH) command will execute. There are three modes:

0 Switch Only– The control will only search for the appropriate edge of a switch.
1 Switch Then Z Channel – The control first waits for a home switch edge, aligns to the edge, and then slowly moves until an encoder Z pulse is found.
2 Z Channel Only – Motor operation continues at a slow speed until an encoder Z pulse is found. The sign of the velocity parameter determines the low speed direction. The magnitude of GH velocity parameter is ignored. The control does not search for a home switch.

HO Homing Offset syntax – <n>HO±r

Value: 0.0 {Distance Units}
Range:

Sets the offset from home position for the “real” home position. After a successful homing move, the home position (the factory set home position is +0.0000) is set to the offset value. Use the numeric keys to enter a value.

This allows multiple systems with identical programs. All that is changed is the home offset value for each machine.

Example: HO1.0 {axis one 1.0 distance units}

HS Home Switch syntax – <n>HSi

Value: Norm Open
Range: Norm Open, Norm Closed

Selects the type of switch used for the home input.

0 Normally Closed
1 Normally Open

Example: HS1 {axis one uses a normally open home switch}

ID Input Definition syntax – <n>IDaaaaaaaa

Value: Norm Open
Range: Norm Open, Norm Closed

Each input is easily configured using the keypad as described in Table 4-2.

Table 4-2

Char	Input Character Description
B	Binary Program Select – allows remote program selection and execution from a PLC, switches, or outputs from a PC. Up to 255 programs may be selected using 8 binary inputs. The lowest numbered input becomes the least significant selection bit (i.e., input #1 is less significant than input #2). The act of configuring an input as a program select input also enables binary program select mode.
C	BCD Program Select – allows remote program selection and execution using a PLC, switches, or outputs from a computer. Up to 99 programs may be selected using BCD inputs. The lowest numbered input becomes the least significant selection bit (i.e., input #1 is less significant than input #2).
c	Clear Command Buffer – Clears the terminal input buffer and buffered command buffer.
D	Disable Keypad – When activated, the keypad is disabled allowing NO user access. The keypad resumes normal operation, subject to the DIP switch pattern, when the input is released.
E (axis1) e (axis2)	Extend Jog – When activated, the motor will Jog in the Extend (+) direction. When the input is released, motion stops at the Jog Accel rate. If an End of Travel limit is hit while jogging, the motor will stop at the Stop Rate (see Edit–Setup–Misc.). Before the motor can be removed from the limit, a Stop or Kill input must be activated to clear the fault generated by the End of Limit switch. Serial commands S or K will also clear the fault. The velocity is determined by the Jog Speed Input and the Jog Low and High setup parameters. When the input is off, the speed is low. If no input is configured for Jog Speed, the motor will jog at the Jog Low setting.
G	Registration – Input #1 must be configured as a Registration input – no other inputs will work. See the RG command for more details.

Table 4-2 Continued

Char	Input Character Description

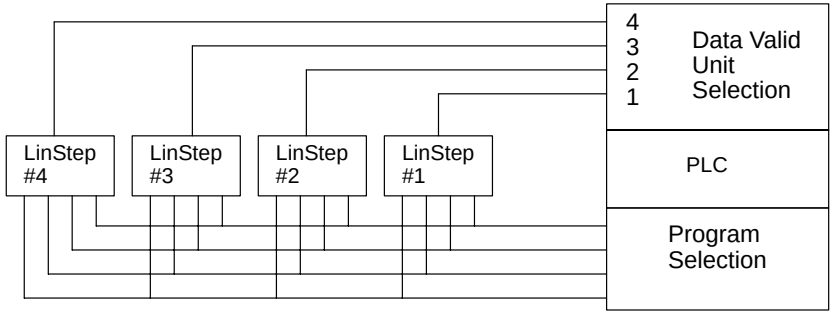
I	Interrupt (Run 98) – When activated, motion on all axes stop at the stop–rate (see Edit–Setup–Misc–Stop–Rate). The current program is stopped, and processing continues with the first command in program 98. If no program is running when the input is activated, program 98 will run. This input is ignored while the keypad is in Edit mode. This is a positive edge triggered input, rather than a level sensitive input. If multiple inputs are configured as Interrupts, only the first edge of the first activated input will be seen. If subsequent Interrupt inputs go active while the first Interrupt input is active, no additional interrupts will be seen.
	Advanced Interrupt handling can be achieved using the INT98CTRL and ARM INT98 variables. The INT98CTRL variable determines whether Interrupts can be disabled or not. The ARM INT98 variable allows you to arm and disarm the Interrupt as desired.
	After power–up, INT98CTRL is initialized to 0. In this mode, every interrupt results in an immediate jump to program 98, even if you just entered program 98. The value of (ARM INT98) is ignored.
	When INT98CTRL=1, Interrupts can be disabled with the ARM INT98 variable. INT98CTRL=1 also initializes ARM INT98 to 1. This means the control is watching for interrupts. When INT98CTRL is set to 1 an interrupt causes the program to jump to program 98 and sets ARM INT98=0, disabling any further interrupts until you re–enable them by setting ARM INT98=1. This allows you to control when you want to re–enable Interrupts in your interrupt service routine (program 98).
	To summarize, when (INT98CTRL)=1:
	If (ARM INT98)=0, Interrupts are ignored. (ARM INT98) is internally set to 0 on the first edge if the previous (ARM INT98) value was 1. Interrupt processing will be suspended until (ARM INT98) is reset to 1. This allows for input debounce and controlling the ability of program 98 to interrupt itself.
	If (ARM INT98)=1, The system is awaiting the first INT98 input assert edge. Once the interrupt is seen the control will go to program 98. Subsequent interrupts are ignored until (ARM INT98) is set to 1.
	INT98CTRL and ARM INT98 are reset to factory values on power–up.
	Note: There is a space in ARM INT98.
	When activated, any executing program or functional operation is terminated and program I98 (interrupt program) is immediately executed. If a move is executing when the interrupt is activated, the move is terminated (decelerated at a rate determined by the Stop Deceleration rate setup parameter). The unit will go into Run mode once program I98 is completed

J (axis1) j (axis2)	Jog Speed – When a jog input is activated, the control checks the state of this input to determine the jog speed. If the input is OFF, the system will jog at the Jog Low speed. If the input is ON it will jog at the Jog High speed. If the input is not configured the jog inputs will jog at the Jog Low speed. This input works along with the Extend Jog and Retract Jog.
K	Kill Motion – Causes the control to abruptly stop commanding further motion and terminates program execution. No deceleration ramp is used. Caution: instantaneous deceleration could cause damage to mechanics. The Stop input provides a more controlled halt.
M (axis1) m (axis2)	Motor Shutdown – May be activated when the control is not running a program and the motor is idle. Selecting shutdown (M, m) will disconnect power to the motor, which removes current (torque) and allows the motor to spin freely.

Table 4-2 Continued

Char	Input Character Description
P	Pause/Continue – When this input is grounded, program execution is stopped. Moves are not interrupted when the Pause input goes active. Command execution will pause at the end of the move, and continue when the input goes high. See the ST and RG commands for interrupting moves in progress.
R (axis1) r (axis2)	Retract Jog – When activated, the motor will Jog in the Retract (–) direction. When the input is released, motion stops at the Jog Accel rate. If an End of Travel limit is hit while jogging, the motor will stop at the Stop Rate. (see Edit–Setup–Misc.) Before the motor can be removed from the limit, a Stop or Kill input must be activated to clear the fault generated by the End of Limit switch. Serial commands S or K will also clear the fault. The velocity is determined by the Jog Speed (J) input and the Jog Low and Jog High setup parameters. When the input is off, the speed is low. If no input is configured for Jog Speed, the motor will jog at the Jog Low setting.
S	Stop – When activated, any program execution or functional operation is immediately stopped. This includes any motion, time delays, loops, and faults. Moves will decelerate at the stop deceleration rate. New programs will not execute until the stop input goes inactive. See the SCAN setup parameter for more information on stopping program execution. See the ST command for more information on stopping moves without halting command execution.
U	Unassigned – An Unassigned input acts like a programmable input, and can be used in IF and WT statements just like any of the dedicated function inputs.
V	Data Valid – When configured, it determines if the Binary/BCD program select inputs are processed or ignored. If the input is active, program select inputs are processed, otherwise they are ignored. This allows applications to be wired in a pseudo–bus architecture fashion with each unit sharing the same program select lines, and the data valid inputs determining which units should listen. Configuring this output can greatly reduce panel wiring. In the example shown in Figure 4-2, using the Data Valid input reduced the number of wires by one–half.
W	Warm Boot (System Reset) – Clears the RAM Buffer, and resets the LinStep+ to its power–up state. Programs and setup parameters are not erased. This is typically used to restart the system when a fault condition occurs. The power–up program, if defined, will start.

Figure 4-2



IL Idle Mode syntax – <n>ILi

Value: OFF
Range: ON or OFF

Selects the type of switch used for the home input.
0 Off
1 On

IR Position Maintenance In–Range Deadband syntax – <n>IRi

Value: 25 Encoder Steps
Range:

In–Range Window specifies the position maintenance deadband or region surrounding the set–point position. The “window” is specified in post–quadrature (4 x # of lines) encoder steps. The window is the region surrounding the commanded position in which the motor shaft can reside and not be considered “out of position.” The control will try to correct the position if the motor is outside this window. (No variables).

JA Jog Acceleration syntax – <n>JAr

Value: 25 (units defined by AU)
Range:

Sets the acceleration and deceleration used during a jog move. Use the numeric keys to enter a value (units were selected in the SETUP > MECH > ACCEL menu).
Example: JA.01 {0.01 and units are defined by AU}

JE Jog Enable syntax – <n>JEi

Value: Disabled
Range: Disabled, Enabled

Enables or disables JOG features. When disabled, an error message is displayed when the jog buttons are pressed. JOG is often disabled after installation to prevent access.
0 Disabled
1 Enabled
Example: JA.01 {0.01 and units are defined by AU}

JH Jog High Velocity syntax – <n>JHr

Value: 2.0 (units defined by VU)
Range:

Sets the high speed JOG velocity.

JL Jog Low Velocity syntax – <n>JHr

Value: 0.5 (units defined by VU)
Range:

Sets the low speed JOG velocity.

MD Motor Direction syntax – <n>MDi

Value: Positive
Range: Positive or Negative

Provides a convenient way to change the motor direction for a positive distance command. When POSITIVE is selected as the motor direction, the EOT+ limit switch should be wired so that moves in the plus direction will activate the switch. When NEGATIVE is selected, the EOT+ limit switch should be wired so that moves in the negative direction will activate the switch.

0 Positive Direction
1 Negative Direction

MH Motor Inductance syntax – <n>MHa

Value: High
Range: High or Low

L Low Inductance
H High Inductance

MI Motor Current — syntax – <n>MIn

Value: 0.0 Amps
Range: 0.0 – 8.0 Amps

This parameter sets the motor current for your stepper motor. Entering a value outside the “Range” will reset the motor current to 0.0 Amps.

MR Motor Resolution — syntax – <n>MRi

Value: 36000 Steps/In (fixed)
Range: N/A

The Motor Resolution is fixed at 36,000 as shown.

MT Motor Type — syntax – <n>MT10

Value: 10
Range: N/A

The Motor is fixed.

MV Maximum Velocity — syntax – <n>MVr

Value: 50.0 in/s
Range: N/A

Sets the top motor speed. (Limits the speed to prevent accidental damage to your mechanics.)

OD Output Definition syntax – <n>ODaaaaaaaa

Value: PPPPPPPP
Range: See Table 4-3.

Each input is easily configured using the keypad as described in Table 4-3. The function of each input channel is indicated by a letter at the bottom of the display.
Example: OD:PPPPPPPP {All outputs are defined as programmable}
Note: Even if only 1 output is used, all 8 must be defined.
Note: Lower case Input Characters (b, d, h, k and m) appear on the Keypad but are not used.

Table 4-3

Char	Keypad Display	Input Character Description
A	AMP FAULT	Amplifier Fault – Output goes low on any amplifier fault. An amplifier fault may be due to temperature, motor short–circuits, excessive following error, over–voltage and excessive regeneration conditions. Note: This is not an all–inclusive fault output. Use F–Fault for this.
B (axis1) b (axis2)	BRAKE	It is often advisable that applications using a ballscrew type actuator with a vertical load use a brake to prevent the load from falling in the event of a fault. The Brake output is normally disengaged, which is actually an ON condition. When a fault occurs, power to the brake is removed and the brake is engaged. This is a “fail–safe” type of brake, controlled by an OPTO module, and it requires a customer supplied, 120VAC power supply, or 24 VDC with B Motors.
C	OVER CURRENT	Not Used.
D (axis1) d (axis2)	DIRECTION	The output remains set until motion is commanded in the reverse direction.
F	FAULT	The fault output acts as an all–inclusive fail–safe output. Under normal operation the output is low (ON) and goes high (OFF) when any type of fault occurs. A fault can occur from any amplifier fault condition (A) as well as for the following general faults: • BMA (Board Monitor Alarm) time–out • Error finding Home – both limits were hit. The exact cause of the fault can be determined a number of ways: • Shown on keypad display • RS–232C using the SS, SD, and SA status commands (see Appendix A) • Other outputs can be configured to show more specific fault states
H (axis1) h (axis2)	AT HOME	The output goes high as long as the axis is at home.
L	LIMIT ERROR	The output goes low if a limit switch is hit during a normal move, or if both limits are hit during a Go Home move.
M (axis1) m (axis2)	MOVE COMPLETE	The output goes high as soon as an axis move is started and goes low when a move is completed.
P	PROGRAMMABLE	Unassigned outputs are set to Programmable and can be used with OT commands.
S	STALL	The output goes low if the control detects a motor stall.

OE Output State on Event (event set by OD) syntax – <n>OEa,iiiiiiii

Value: No Change
Range: On, Off, No Change

The OD command sets the output event definition. The OE command sets the output state when that event occurs. Each output is configured to On, Off state or No Change when an event occurs. The Power Up state, Fault state and Stop/Kill state for each output condition are individually set.

a=P Power up
a=F Fault
a=S Stop/Kill **Note:** Only defines outputs 1–8 (defined by OP command).

i=o Off
i=1 On
i=X No Change

Example: `OEF,0001XX01XXXXXXXX`

OP Opto Input/Output Configuration syntax – <n>OPIIIIOOOO

Value: IIIIOOOO
Range: Input or Output

Defines each opto position as an input or an output. Optional Optos are on terminals 9–16.

I Input
O Output

Example: `OP:IIII0000` {Assigns first four optos (9–12) as inputs and the last four (13–16) as outputs}.

PG Position Maintenance Gain syntax – <n>PGi

Value: 10
Range: 1–32767

PM Gain specifies a gain value used to determine correction velocity. The correction velocity is calculated as “displacement* correction gain” in units of steps/in. Therefore, the larger the displacement, the more quickly position maintenance gain will attempt to correct position. For example, if the correction gain is set to 3 and an active displacement of 3200 steps occurs, the correction velocity will be (3 * 3200) = 9600 steps/sec.

PU Power up Program syntax – <n>PUi

Value: 0
Range: 0–400

Sets the program to run at power–up. The selected program is executed (run) when power is applied or after a reset. If a value of 0 is entered in this menu, or if the specified program does not exist, no program is run. Use numeric keys to enter program number.

PV Position Maintenance Maximum Velocity syntax – <n>PVi

Value: 1.0 (units defined by VU command)
Range: 0.005–9,999,999

Limits the velocity of a position maintenance correction. Regardless of the magnitude of displacement of correction gain, the correction velocity will never exceed this maximum velocity setting.

PW Passwords syntax – <n>PWoooo,aaaa

Value: None
Range: N/A

Sets the Operator (oooo) and Administrator (aaaa) passwords. Passwords allow you to restrict access to the RUN, EDIT, COPY, DEL and keypad DIP switches. Select Operator or Administrator (see Table 4-4 for description).
Enter a password, use ← → and DEL keys to edit the password.

General Password Rules:

- Passwords are 4 characters maximum, a–z, A–Z, 0–9, in any combination.
- If no password is entered, there is no restriction.
- Entering the wrong password or pressing ESC at the password prompt will return the keypad to the standard run–time display.
- Select EDIT > SETUP > MISC > PASWRD > CLEAR to delete all passwords.

Note: Subsequent attempts to RUN or EDIT a program do not require a password. You are prompted to: Use Last (F1) or Reset (F3). Select Use Last to run or edit another program. Select Reset to require the next user to enter a password.

Example:

1	PW4FT,Q12h	Set the OPRATR password to 4FT and the ADMIN password to Q12h
2	PW,New	Changes ADMIN password to New; leaves OPRATR password unchanged
3	PW–,–	Clears both the OPRATR and ADMIN passwords

Table 4-4

Password Type	Description	Gives access to these menus
OPRATR	Operator only	RUN, EDIT, COPY, DEL
ADMIN	Administrator only	RUN, EDIT, COPY, DEL
OPRATR + ADMIN	Operator and Administrator	ADMIN = RUN, EDIT, COPY, DEL OPRATR = RUN only (All RUN functions except TEST)

RE Rest Mode — syntax – <n>REi

Value: OFF
Range: ON or OFF

When Rest Mode is enabled, motor current is reduced to 1 Amp if no motion occurs for 12 minutes. Full current is restored when the next move starts. Enabling REST (ON) improves fan life in applications where the machine is powered–up, but may not run for extended periods of time, e.g. a machine that is operated two shifts, but is left on 24 hours a day.

0 Disable rest mode
1 Enable rest mode

SN Scan Conditions — syntax – <n>SNiiiiiii

Value: 00000000
Range: 0 or 1

Selects the events that cause the control to stop scanning program–select configured inputs. If a stop–scan event is enabled, the system will stop scanning the inputs for program numbers when that condition occurs. To resume scanning, a reset (Warm Boot input or cycle power Off then On) must be given. This option has no effect if the inputs are not configured as program select inputs (either BCD or Binary). Seven events are represented: ESCape, STOP, LIMIT+, LIMIT–, KILL, FAULT or INTerrupt.

0 Continue program select scanning
1 Stop program select scanning

SR Stop Deceleration Units — syntax – <n>SRr,r

Value: 0.1 in/s^2 (units are always in/s^2 and cannot be changed)
Range:

Set the deceleration rate used whenever a configurable stop input is activated, or when the ESC key is pressed during a move. Normally set to the fastest controllable deceleration rate possible with mechanics in your application.
Example: SR100 {100 in/s^2}

UN Unit Number — syntax – <n>UNi

Value: 1
Range: 1–99

Set the unit's address. Each unit in an RS–232C serial daisy chain of multiple units must have a unique Unit Number.
Example: UN5 {sets unit address to 5}

VU Velocity Units — syntax – <n>VUi

Value: 0
Range: 0–3

Sets the velocity units. All velocity values will be expressed in these units.

0	units/sec (where units is defined by the DU command)
1	units/min (where units is defined by the DU command)
2	RPS
3	RPM

WA Waveform — syntax – <n>WAi

Value: Sinusoid
Range: Sinusoid or –4% 3rd Harmonic

Depending on motor design and the current level at which it is being driven, it may be advantageous to distort the sinusoidal waveform to achieve better low speed smoothness and step–to–step accuracy. With the motor running at low speed, alternately select between SINUSOID and –4% 3rd to determine which setting produces the smoothest running condition.

Serial Immediate Status Commands

Serial Immediate Status commands are processed immediately upon receipt, rather than waiting in the buffer for previous commands to finish. They can be issued while a program is running, or while motion is in progress. These commands are not used within a program. These commands will interrupt the LinStep+ and generate a return. They do not affect operation of the LinStep+.

Using Immediate Status Commands

Serial Immediate Status commands serve two purposes. The first is to allow a query in real time, for system, position, and I/O status. In a typical hosted–mode application, all machine operations and decisions are performed by a high level device. Motion commands are generated and downloaded to the LinStep+ by this host device. Immediate commands are provided so the host can verify the LinStep+ status before commanding motion. The system status (SS) command returns overall system information and general faults. The axis and drive (SA and SD) commands can then be used for more detailed, axis–specific information.

The second is to allow in–depth troubleshooting over the RS–232 lines. Since they are immediate commands, they will generate a response during a move, while waiting for an input condition to become true, etc. Checking the system status and the I/O status provides enough information to determine what the LinStep+ is doing. If a fault is indicated, the drive status and axis status commands provides axis–specific information.

CB Clear Command Buffer — syntax – <n>CB

Value: N/A
Range: N/A

Clears the terminal input buffer and buffered command buffer

IS Tell Input States — syntax – <n>IS

Value: N/A
Range: N/A

Returns the current state (on or off) of the 8 inputs. The status is returned as a four digit hexadecimal number, preceded by an asterisk. The least significant digit represents the binary value of inputs 4–1. Example: IS returns *00F6<cr> with the input conditions shown. Your computer program must decode the hexadecimal number to determine the state of each input.

n/a								Inputs							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
off	off	off	off	off	off	off	off	on	on	on	on	off	on	on	off
0				0				F				6			

K Kill — syntax – <n>K

Value: N/A
Range: N/A

Issuing the K command causes the control to abruptly stop. All motion and program execution is terminjated. No deceleration ramp is used with this command. Caution should be used in issuing this command because of the damage instantaneous deceleration may cause to sytems mechanics. (Stop provides a halt with controlled deceleration.)

MN Model Number — syntax – <n>MN

Value: N/A
Range: N/A

Issuing the MN command causes the control to return its model number.

OS Tell Output States — syntax – <n>OS

Value: N/A
Range: N/A

Returns the current state (on or off) of the 8 Outputs and any of the Optos that are configured as digital Outputs. The status is returned as a four digit hexadecimal number, preceded by an asterisk.
Example: OS returns *00F6<cr> with the Output conditions shown. Your computer program must decode the hexadecimal number to determine the state of each input.

n/a								Inputs							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
off	off	off	off	off	off	off	off	on	on	on	on	off	on	on	off
0				0				F				6			

PA Tell Absolute Position — syntax – <n>PA1

Value: N/A
Range: N/A

Reports current position in user units based on encoder mode selected. Can report commanded or actual encoder position when PAa,n is used.
Example: PA1 returns *+1.000 (the position of axis one)

RS Reset System — syntax – <n>RS

Value: N/A
Range: N/A

Initialize (warm boot) the control software to its power–up state. Initialization takes about 10 seconds to complete. Programs and configuration settings are not erased. This command is the equivalent of cycling power.

S Stop — syntax – <n>S

Value: N/A
Range: N/A

Terminates program execution and immediately decelerates each motor to a halt (at a rate set by the SR command). Functions the same as the pressing ESC key on the IDC keypad or activating an input defined as a Stop input.

SA Tell Axis Status syntax – <n>SA1

Value: N/A
Range: N/A

Returns the current axis status as a four digit hexadecimal number, preceded by an asterisk. Your controller program will decode the hexadecimal number to determine the axis status. See Table 4-5.

Example: SA1 returns *002A<cr>. This means Axis 1 is not moving, the last move completed successfully, and the home switch is on.

n/a								Inputs							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
off	off	off	off	off	off	off	off	off	off	on	off	on	off	on	off
0				0				2				A			

Table 4-5

Description	bit #	Parameter Definition
Not Moving/Moving	1	1=Steps being sent to the amplifier 0= No steps being sent
At Velocity	2	1= Stepping at a constant rate (includes zero velocity) 0= Step rate is changing
In Range	3	Not Used.
Move Command Complete (Same as Move Done Output)	4	1=The correct number of steps were sent without an amp fault, following error, or hitting an End of Travel limit. 0=Reset to zero at the beginning of each move.
Home Successful	5	1= The last homing move was successful 0= At power up, reset to zero at the start of the next jog, GO, or GH.
Home Switch Status	6	Hardware status of home switch. 0=off, 1= on
– Limit Switch Status	7	Hardware status of limit switch 0=off, 1= on, limits require a NC switch
+ Limit Switch Status	8	Hardware status of limit switch 0=off, 1= on, limits require a NC switch
– Limit Switch Latched	9	1= Set when a move is terminated by a limit in the –direction. Cleared at the start of a move in the +direction. 0= At power up or reset, even if on the – limit.
+ Limit Switch Latched	10	1= Set when a move is terminated by a limit in the +direction. Cleared at the start of a move in the –direction. 0= At power up or reset, even if on the + limit
RESERVED	11	State undefined, should be masked
RESERVED	12	State undefined, should be masked
RESERVED	13	State undefined, should be masked
RESERVED	14	State undefined, should be masked
RESERVED	15	State undefined, should be masked
RESERVED	16	State undefined, should be masked

SD Tell Drive Status syntax – <n>SD1

Value: N/A
Range: N/A

Returns the current drive status as a four digit hexadecimal number, preceded by an asterisk. Your controller program decodes the hexadecimal number to determine the drive status. See Table 4-6.
Example: SD1 returns *0010<cr>. This means Axis 1 is enabled, in position mode, and no faults.

n/a								Inputs							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
off	off	off	off	off	off	off	off	off	off	off	on	off	off	off	off
0				0				1				0			

Table 4-6

Description	bit #	Parameter Definition
Following Error	1	1= Following error occurred 0=At power up and reset. Set to zero at the start of the next move.
Over–current	2	1= Over Current (requires reset to clear) 0= At power up and after reset.
Thermal Fault	3	1= Thermal fault in the motor or drive (requires reset to clear) 0= At power up and after reset.
RMS Over–current	4	Not Used.
Drive Enabled	5	1= Enable Drive (see also EA1) 0= Disable Drive (see also EA0)
RESERVED	6	State undefined, should be masked
RESERVED	7	State undefined, should be masked
Torque/Position	8	Not Used.
Amplifier Fault	9	1= The amplifier is faulted. Requires a power cycle to reset. 0= At power up or reset
RESERVED	10	State undefined, should be masked
RESERVED	11	State undefined, should be masked
RESERVED	12	State undefined, should be masked
RESERVED	13	State undefined, should be masked
RESERVED	14	State undefined, should be masked
RESERVED	15	State undefined, should be masked
RESERVED	16	State undefined, should be masked

SS Tell System Status syntax – <n>SS

Value: N/A
Range: N/A

Returns the current system status as a four–digit hexadecimal number, preceded by an asterisk. Your controller program decodes the hexadecimal number to determine the system status. See Table 4-7. Example: SS returns *0001<cr> means there are no amplifier faults, and no programs running – LinStep+ is ready to process any buffered RS–232C command.

n/a								Inputs							
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
off	off	off	off	off	off	off	off	off	off	off	off	off	off	off	on
0				0				0				1			

Table 4-7

Description	bit #	Parameter Definition
Ready to Buffer RS–232C Commands	1	1= Ready to buffer RS–232C commands 0= Initializing from a power–up or reset, or unchecked errors exist. All buffered commands sent will be discarded.
Flash Memory Error	2	1= Non–volatile memory checksum error, all programs deleted at power up. 0= Non–volatile memory checksum OK
Program Running	3	1= Running a program 0= Not running a pre–defined program.
FK Active	4	1= Paused waiting for a function key. 0= Not waiting at a FK command.
WT Active	5	1= Paused waiting for a input condition. 0= Not waiting at a WT command
TD Active	6	1= Paused at a time delay. 0= Not waiting at a TD command
Waiting for IV	7	1= Paused, waiting a variable input. 0= Not waiting at a IV command
Buffer Full	8	1= RS–232C buffer 75% full Total Capacity: 2k 0= RS–232C buffer less than 60% full.
Axis #1 Fault	9	1= Amp fault, following error, move stopped by limit switch (see SAi and SDi for more detailed fault information) 0= No faults
Axis #2 Fault	10	1= Amp fault, following error, move stopped by limit switch (see SAi and SDi for more detailed fault information) 0= No faults
RESERVED	11	State undefined, should be masked
Program Select Scanning	12	1=BCD and Binary program select scanning enabled. 0=A Stop Scan condition has occurred or no inputs are configured as program select lines.
Data Download Status	13	1=Data Transfer failed (program memory overflow) 0=Data successfully received
RESERVED	14	State undefined, should be masked
RESERVED	15	State undefined, should be masked
RESERVED	16	State undefined, should be masked

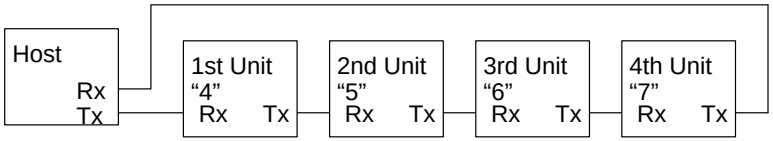
Serial Supervisory Commands

These commands control the uploading, downloading, deleting and execution of the setup and program parameters.

AA Auto Address — syntax – <n>AA or AAi

Value: N/A
Range: 1–99

Auto Address
Automatically addresses units in a daisy chain. It assigns an address to each unit on the daisy chain. This allows the units to be wired in a daisy chain without setting each unit's address manually. The AA command parameter n indicates the beginning value the address sequence.



In this example, the Host issues an AA4 and the units are assigned addresses 4, 5, 6, 7. This offers the convenience of adding a new unit anywhere in the daisy chain without manually re–addressing all the other units. Just connect the new unit, issue an AA command from the new unit with the address of the new unit as the AA parameter , i.e. AAi.

DP Delete Program — syntax – <n>DPi

Value: N/A
Range: 1–400

Erases a program from memory, where i is the program number. This is equivalent to pressing the delete key on the keypad and entering the program number. Range: 1–400
Example: DP99 (deletes program number 99).

DR Download a Program to RAM — syntax – <n>DRi

Value: N/A
Range: 1–400

(also see the PR command)
Downloads a program from the host to the LinStep+'s RAM, rather than non–volatile memory. These programs will be lost at reset or power cycle. The program string must end in EP. The commands between DR and EP do not need a device address.
The DR command is typically used when operated exclusively with a host controller which constantly downloads and executes programs. This increases the usable life of the FLASH. Range: 1–400 (w/30kB)
Example: `1DR50 AC4 DE4 VE30 LP6 DI10.5 GO EN 1EP RN50`
Downloads program #50 to Unit #1 RAM, then runs program #50

EC Echo Enable/Disable — syntax – <n>ECi

Value: N/A
Range: 1–400

RS–232/485 Echo Enable/Disable 0 = echo Disabled, 1 = echo Enabled.
Example: EC0 (echo off). Echo must be enabled for daisy chain.

EP End Program Definition — syntax – <n>EP

Value: N/A
Range: N/A

Defines the end of a program. All program definitions must begin with nPRi or nDRi and end with EP.
Example: `PR15 [part A] AC4 VE30 DI10.5 GO EP`

EX End Upload All or Load All — syntax – <n>EX

Value: N/A
Range: N/A

EX is sent by the LinStep+ to the host after completing a UA.
EX is sent by the host to the LinStep+ to terminate a LA.

LA Load All — syntax – <n>LA

Value: N/A
Range: N/A

Sent to the LinStep+ before downloading a long list of setup parameters and programs. This command disables the non–addressed units so that each setup parameter doesn't need an address. Must be followed by an EX to re–establish the daisy chain communications.

LS List Programs — syntax – <n>LS

Value: N/A
Range: N/A

Lists number of programs, memory usage, and available memory. Like Edit/List command from the keypad.

OC Original Configuration — syntax – <n>OC

Value: N/A
Range: N/A

Returns the FLASH contents to factory settings. The command buffer is cleared, all programs are erased, and all configuration settings are returned to factory settings.

PR Define Program — syntax – <n>PRi

Value: N/A
Range: 1–400

Starts a program definition, like the DR command, but writes to LinStep+ non–volatile EEPROM memory.
Example: `PR25 AC.1 VE5 DI10 GO EP` (uses program number).
Example: `PR25 [P/N 170–001] AC.1 VE5 DI10 GO EP` (uses optional program name).

RN Run Program — syntax – <n>RNi

Value: N/A
Range: 1–400

Commands LinStep+ to run a program, by number only. The RN command does not support the optional program names.
Example: RN25

SW Tell Software Version — syntax – <n>SW

Value: N/A
Range: 1–400

LinStep+ returns its software revision.
Example: `1SW` returns `*V1.40 <cr>`

UA Upload All — syntax – <n>UA

Value: N/A
Range: N/A

Uploads all setup parameters and programs from unit n. LinStep+ sends an EX to terminate upload.

UL Upload Program — syntax – <n>ULi

Value: N/A
Range: 1–400

Uploads program number i to the host – LinStep+ adds brackets.
Example: `1UL2` Uploads program 2 from unit #1. Response: `{ [part A] AC4 VE30 DI10.5 GO }`

Refer to Section 3 for Expressions, Operators, Functions and Programming hints.

BALDOR ELECTRIC COMPANY
P.O. Box 2400
Ft. Smith, AR 72902–2400
(501) 646–4711
Fax (501) 648–5792
www.baldor.com

CH TEL: +41 52 647 4700 FAX: +41 52 659 2394	D TEL: +49 89 90 50 80 FAX: +49 89 90 50 8491	UK TEL: +44 1454 850000 FAX: +44 1454 850001	F TEL: +33 145 10 7902 FAX: +33 145 09 0864
I TEL: +39 11 562 4440 FAX: +39 11 562 5660	AU TEL: +61 29674 5455 FAX: +61 29674 2495	CC TEL: +65 744 2572 FAX: +65 747 1708	MX TEL: +52 47 61 2030 FAX: +52 47 61 2010

MN1855

Printed in USA
7/01 C&J 1000