Asymtek
A NORDSON COMPANY

Automove®
Control Language (ACL)
Version 3.61

Programming Reference Manual

P/N 76-CS02-01, Revision A 6/96

NOTICE

All information contained in or disclosed by this document is considered proprietary by Asymtek. By accepting this material the recipient agrees that this material and the information contained therein are held in confidence and in trust and will not be used, reproduced in whole or in part, nor its contents revealed to others, except to meet the purpose for which it was delivered. It is understood that no right is conveyed to reproduce or have reproduced any item herein disclosed without express permission from Asymtek.

Asymtek provides this manual "as is," without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Asymtek may make improvements or changes in the product(s) or programs described in this manual at any time. These changes will be incorporated in new editions of this publication.

Asymtek assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of this manual, nor for any problem that might arise from the use of the information in this manual.

Asymtek

International Headquarters	**2762 Loker Avenue West** **Carlsbad, CA 92008-6603**
Toll Free Line:	**1-800-ASYMTEK**
Tel:	**619-431-1919**
Fax:	**619-431-2678**
Email:	**info@asymtek.com**
Web site:	**http://www.asymtek.com**
European Office	**The CadCam Centre** **Bighton, Alresford, Hamshire** **SO24 9RE England**
Tel:	**44 (0) 1962 73566**
Fax:	**44 (0) 1962 735734**

Technical Support **(U.S. and Canada) 1-800-ASYMTEK**

Millennium Series is a trademark of Asymtek.

TABLE OF CONTENTS

Table of Tables

1 Introduction

ACL and This Manual

The Automove® Control Language (ACL) is a simple language for controlling Asymtek Automove Series of X-Y tables and motor controllers. A host computer is connected to the Automove hardware via an RS-232C cable. The host sends ACL commands, in ASCII (American Standard Code for Information Interchange), to the Automove hardware. Each command causes the Automove System to perform some simple action such as moving the carriage or actuating the Digital Outputs.

This manual is intended to serve as a reference for the person writing a host computer program to control the Automove System via ACL. Unless specifically noted, all information in this manual applies equally to all models of the Asymtek Automove System. For further information about system installation and operation, please see the Automove Operations Manual.

A good way to experiment with ACL and get a feel for how the commands work is to connect the Automove System to a terminal or a computer emulating a terminal. As you type the commands they are executed immediately. When the Automove System detects an error in a command it sends a "?" back; you can type OE or ESC.E to find out what went wrong.

There is a simple example program in Appendix G which shows how to use the BASIC language to control an Automove System. Appendix H gives a few suggestions for debugging the system when something doesn't work right.

Manual Conventions

- ACL commands are shown in Arial font, i.e. MR, OA, SO.
- Terms described enclosed in a < > in Arial font are variables to be entered into the command. For example: SR[<rate>] means that SR is the ACL command and <rate> is the variable entered for command execution, such as SR[65535].

In the examples the command parameters are shown separated by spaces. These spaces may be sent and will not change the effect of the commands, except possibly to slow things down a bit due to the transmission of the extra characters over the RS-232C interface.

Input Buffer

As they are received, ACL commands are put into a 256-byte input buffer. After the machine finishes processing one command, it fetches the next from the buffer. In order to prevent the host from sending characters too fast and thus overflowing the buffer, one of four input buffer handshakes must be implemented within the host. These are: Hardwired DTR, Xon/Xoff, Enq/Ack, and Software Checking. The host sets up the desired handshake by sending escape sequences to the Automove System. For more information about handshakes, which handshake to use, and how to set up a handshake, see Chapter 5 RS-232C Communications, Chapter 6 Escape Sequences, and Appendix F Choosing a Handshake.

Numeric Coordinate Systems

The Automove X-Y tables have a carriage which moves above a flat surface called a platen. If you are using the stand-alone controller with your own mechanical system, your physical system may not have anything resembling a carriage or platen. However, for convenience in this document we talk about moving the **carriage** above the **platen**.

The physical coordinate system is based on physical motor counts, or microsteps, which for the Automove X-Y tables are nominally 0.001 inch. Calibration factors can be entered via the CF command (see Chapter 3) so that the host software can work in true 0.001 inch Calibrated Units or any other convenient units. At power up the System designates the current position to be the Home position, or (0,0) microsteps.

The FH ("Find Home Switches") command causes the carriage to move to the lower-left corner of the platen. The point where both Home switches have closed is then defined to be the Home position, or (0,0) microsteps. Another way to find the Home switches is by a front-panel manual re-reference sequence; see Chapter 4. Typically, an FH command or manual re-reference should occur before any ACL motion commands or other front panel interactions are attempted. If the Home switches are not found, the Automove System will not "know" where the carriage is with respect to the physical boundaries, and the TL travel limits will not work correctly. Also, since vectors and arcs to negative physical coordinates are prohibited, a portion of the platen would only be reachable via the Arrow buttons.

The origin of the Calibrated Units coordinate system (the "Origin position") can be offset from the Home position via the front-panel SET ORIGIN button, the SO command, or the BP command. Also, the BP command can cause the entire coordinate system to be rotated by a specified angle.

See the BP, CF, CR, MA, and SO commands in Chapter 3 for more information about calibration factors, origin offsetting, pattern rotation, and linearity correction.

Within this document the term **Calibrated Units** always refers to this "corrected" coordinate system. The term **microsteps** always refers to an actual number of physical microsteps at the motors. If you want your computer program to deal in units other than microsteps, the calibration factors can be changed to suit your needs. For example, values near 1.0000 can be used to compensate for small mechanical variations, or larger values can be used so that a Calibrated Unit is one inch or one millimeter. (This does not change the physical size of a microstep, only the number of microsteps per calibrated unit.) Use caution, though, since the calibration factors affect vectors but not the shape of arcs.

Microstep Size, Resolution, and Maximum Travel

The Automove X-Y tables are shipped with 1.8-degree stepping motors. This means that the motors have 200 full steps per revolution. (There are 50 physical poles on the rotor; four full steps cause a pole to rotate to the position formerly occupied by its neighbor.) By default the Automove System divides each full step into eight microsteps, giving 1600 microsteps per revolution. Each microstep on an Automove table produces nominally 0.001 inch of carriage travel.

The System allows a total travel of 32766 microsteps in each axis, or about 32 inches in the default configuration. If this produces too little overall travel for your application, or if eight microsteps per full step is too coarse, you can change the size of a microstep via the RE ("Resolution") command. A microstep can be as large as one full step, or as small as one thirty-second of a full step. This gives a range of 200 through 6400 microsteps per revolution for 1.8-degree motors.

Whenever the microstep size is changed the System loses its Home position reference. Thus if the RE command is to be used it should be the first command executed after power up, and should be followed by an FH ("Find Home") command or a manual re-reference. Alternatively, you can change the power-up default resolution via the PE ("Personality") command.

Another way to increase travel is via the SP ("Set Position Counters") command. You can move from one end of the travel to the other, and then reset the position counters to their original values. This effectively moves the entire coordinate system to a different place, giving increased travel with no sacrifice in resolution.

Motor Direction

The motors should be physically and electrically connected so that travel toward lower-numbered coordinates produces motion towards the Home switch in each axis. If your mechanical setup or wiring is such that the motor travel is backwards you can reverse the direction within the Automove controller. See personality parameters 29, 30, and 31 in Chapter 7 Changing the Personality.

Command Syntax

Each ACL command begins with a two-letter mnemonic; for example, MA for "Move Absolute". The mnemonic may be either upper case or lower case letters. In some commands the second character of the mnemonic is one of the following:

< > = + - * / & ! |

Depending on the command, the mnemonic may be followed by one or more numeric parameters The parameters must be separated from each other by at least one comma, space, or sign ("+" or "-") character. The end of the command is delimited by a semicolon (";") or by the mnemonic of the following command.

As soon as the last parameter has been delimited, the Automove System begins to execute the command. (It is a good idea always to send the semicolon; otherwise a command with parameters will not execute until the following command comes along to delimit the last parameter. This could be a long time.) <Carriage Return> (character code 13) and <Linefeed> (character code 10) are ignored in ACL; therefore they will NOT serve as parameter delimiters or command terminators.

Any number of commas and spaces may occur before, between, or after parameters; they will serve as delimiters but are otherwise ignored. They may not appear between the digits of a given parameter. If a sign character ("+" or "-") is to be used, no characters may occur between the sign and the first digit character.

Certain commands allow their parameters to be omitted, in which case the corresponding System variables are given default values. These are called **optional** parameters. A semicolon (or the mnemonic of the following command) anywhere in the parameter list causes all remaining unspecified parameters to be defaulted, if the command allows it.

For example, the following command sequences are equivalent:

```
MA 300, 400;  AB 0;  OS;

ma 00300  400ab0os

MA,+300+400.00;;;ab;OS;
```

While processing ACL commands, the System ignores control characters (i.e. those with ASCII codes below 32 decimal, including Carriage Return and Linefeed) and the characters ", %, ', (,), :, ?, [, \,], _, `, {, }, ~, and DEL. These characters have no effect if they appear in the parameter list of an ACL command, or between commands. However, they may not appear in the parameter list of an escape sequence. (See Chapter 6 Escape Sequences.)

Out-of-range values typically have the following effect: If any bad parameter is detected during the processing of a command, an error is logged but otherwise the command is ignored. Even if other parameters were OK, they do not take effect. (For an exception to this rule see the MA command.)

If too many parameters are received the command executes with the required number of parameters, an error is logged, and the remaining parameters are ignored.

Any ACL error causes the System to ignore all characters until the next semicolon ";" or letter "A" through "Z".

Parameter Types

Some commands, for example ES and OU, have string parameters. The syntax of string parameters is described with those commands in Chapter 3.

Numeric parameters fall into two categories: fractional and whole numbers. Fractional parameters typically can be in the range -32768.0000 through +32767.9999. Only the first four digits after the decimal point are significant; if more are given, the excess digits are ignored. If no sign ("+" or "-") is given the parameter is assumed to be positive.

Whole number parameters are typically treated as integers in the range 0 through 65535. If there are digits after the decimal point, the number is rounded to the nearest integer. Values in the range 32768 through 65535 may be sent in two different forms: either as themselves or as negative numbers in the range -32768 through -1, respectively; i.e., as the desired value minus 65536. For example, the following two commands are equivalent:

```
XD 3, 65532;

XD 3, -4;
```

Where a numeric parameter is expected, the value of an ACL variable can be substituted. For more information about variables, please see Chapter 10.

Getting Status Information

Several ACL commands and escape sequences cause the Automove System to send one or more numbers back to the host. When this occurs, the numbers are always separated by commas; after the last one, the System sends the Output Terminator sequence. At power up this sequence defaults to Carriage Return and Linefeed. It can be changed via the ESC.M command; see Chapter 6.

When the System sends fractional numbers it suppresses trailing zeros after the decimal point, and suppresses the decimal point itself if the fractional part is zero. See, for example, the OC command.

When the System sends whole numbers back to the host, it always uses the "normal" representation: 0 through 65535. No decimal point is sent. See, for example, the OZ command.

See **Outputs to the Host** in Chapter 5.

Personality Parameters

Many hidden aspects of the Automove System's personality can be changed via the PE ("Personality") command. The personality parameters are stored in the built in non-volatile memory and so are "remembered" even when you turn the power off.

You can change such things as: arrow button speed and direction; Find Home speed; power-up values of calibration factors, resolution, travel limits, acceleration, and step rate; direction of motor rotation. In addition, you can set up the eight Digital Inputs to trigger Download Sequences or to toggle the Digital Outputs; and you can establish a Download Sequence which is to be automatically executed at power up.

For a complete description of the personality parameters and how to change them, see Chapter 7 Changing the Personality. A summary of the personality parameters is in Appendix D.

Continuous Path

Sequences of straight-line and curved moves can be performed without stopping. See the BC ("Begin Continuous Path") command in Chapter 3.

Downloading

Any sequence of ACL commands can be downloaded into the Automove System's non-volatile memory for later execution. Up to 255 independent Download Sequences can be defined and they can invoke each other, with any number of executions triggered by a single invocation.

Download Sequences can be invoked from the front panel or via a voltage change on one of the Digital Inputs. The speed or flow of execution can be changed as the result of external inputs. For more information see the BD ("Begin Download") and WN ("Wait For Digital Inputs") commands in Chapter 3, as well as Chapter 8.

ACL Variables

The Automove System can store numeric values in 128 separate ACL variables. Each variable retains its value until it is explicitly changed by the sequence of ACL commands. The values are stored in non-volatile memory, and so are retained even when the power is turned off.

Note: If you have Release 3.48 or above, you may have 384 variables.

A move sequence can perform arithmetic and logical operations on variables and can test their values in order to control the flow of execution.

For more information about variables see Chapter 10.

2 Summary of ACL Commands

Table 1 - X-Y Commands

Command	Description	Page No.
AA	Arc Absolute	3-2
AB	Antibacklash Vectors	3-3
AC	Acceleration	3-4
AR	Arc Relative	3-6
BP	Begin Pattern	3-17
CF	Calibration Factors	3-21
CP	Clear Patterns	3-22
CR	Correction	3-22
EP	End Pattern	3-26
FH	Find Home Switches	3-28
MA	Move Absolute	3-36
MR	Move Relative	3-45
MT	Move To Taught Point	3-46
OA	Output Actual Position	3-47
OC	Output Commanded Position	3-47
OF	Output Calibration Factors	3-49
OG	Output Angle	3-49
OL	Output Travel Limits	3-49
OO	Output Origin	3-50
OR	Output Correction	3-51
OT	Output Taught Point	3-52
PM	Power Level of Motors	3-57
RE	Resolution	3-59
SC	Scale	3-61
SO	Set Origin	3-61
SP	Set Position Counters	3-62
SR	Step Rate	3-63
TL	Travel Limits	3-65
VM	Vector Mode	3-68

Table 2 - Third Axis Commands

Command	Description	Page No.
AZ	Absolute Z Move	3-7
CZ	Configure Z Axis	3-24
FZ	Find Z Home Switch	3-31
MZ	Move Z (relative)	3-46
OZ	Output Z Position	3-54
PZ	Set Power Level of Z Motor	3-58
SZ	Set Z Position Counter	3-64

Table 3 - Flow of Execution Commands

Command	Description	Page No.
XD	Execute Download Sequence	3-76
XI	Execute If (conditionally execute a Download Sequence)	3-77
XU	Execute Until (repeat a Download Sequence Until Condition)	3-78
XW	Execute While (repeat a Download Sequence While Condition)	3-78

Table 4 - ACL Variables Commands

Command	Description	Page No.
OV	Output Variable	3-54
VA	Compute Vector Angle	3-67
VC	Variable Capture	3-67
VL	Vector Length	3-67
VR	Vector Rotate	3-69
VS	Variable Set	3-69
VT	Variable Test	3-70
V<	Variable Less Than	3-71
V=	Variable Equal To	3-71
V>	Variable Greater Than	3-71
V+	Variable Add	3-71
V-	Variable Subtract/Negate	3-72
V*	Variable Multiply	3-72
V/	Variable Divide	3-72
V&	Variable AND	3-72
V|	Variable OR	3-72
V!	Variable NOT/Exclusive-OR	3-72

Table 5 - Digital Outputs Commands

Command	Description	Page No.
AP	Auxiliary Digital Port Select	3-6
CD	Change Digital Outputs	3-20
GD	Lower Retractable Height Sensor	3-33
GU	Raise Retractable Height Sensor	3-33
MD	Mid-move Digital Outputs Changes	3-36
MM	Multiple Mid-move Digital Outputs Changes	3-39
OD	Output Digital Outputs State	3-48
PD	Pre-move Digital Outputs Changes	3-55
TD	Toggle Digital Outputs	3-65
WD	Wait after Digital Outputs Changes	3-74

Table 6 - Digital Inputs

Command	Description	Page No.
MN	Mid-move Digital Inputs Response	3-42
ON	Output Digital Inputs State	3-49
WN	Wait For Digital Inputs	3-74

Table 7 - Configuration and Miscellaneous Commands

Command	Description	Page No.
AD	Abort Download Sequence Execution	3-5
AM	Arrow Mode	3-5
BC	Begin Continuous Path	3-8
BD	Begin Download	3-13
CS	Clear Stop	3-23
EC	End Continuous Path	3-26
ED	End Download	3-26
ES	Execute Escape Sequence	3-26
FP	Front Panel Lockout	3-30
IN	Initialize	3-34
OB	Output Buttons	3-47
OE	Output Error Code	3-48
OI	Output Identification	3-49
OP	Output Personality Parameter	3-50
OQ	Output Qualities	3-50
OS	Output Status	3-52
OU	Output Literal String	3-53

3 Detailed Descriptions of ACL Commands

In the following descriptions, square brackets ("[" and "]") enclose optional parameters (i.e. those which may be omitted so as to receive default values). Angle brackets ("<" and ">") enclose the name of each parameter. The notation ESC.X stands for an escape sequence.

The command parameters are shown separated by spaces. These spaces may be sent and will not change the effect of the commands, except possibly to slow things down a bit due to the transmission of the extra characters over the RS-232C interface.

In all of the examples, unless otherwise specified, it is assumed that the calibration factors are 1.0000 in each axis and the resolution is 8.0000 microsteps per full step in each axis.

The ACL commands are presented in alphabetical order.

AA <x center>, <y center>, <angle> **Arc Absolute**

The Automove System moves the carriage in a smooth circular arc. The center of the arc is specified by <x center> and <y center>, in absolute Calibrated Units as affected by the SO, CF, and CR commands.

The range of <x center> and <y center> is -32768.0000 through 32767.9999, but they must be limited such that the radius of the arc does not exceed 32700 microsteps and such that the arc center lies within the range -32768 through 32767 microsteps in each axis. (Further constraints are in effect if the BC command is active.) Certain microstep resolutions may further restrict the radius; see below.

The arc is circular on the microstep grid; in other words, its shape is not affected by the CF and CR commands.

The arc begins at the current carriage position and proceeds for <angle> degrees in the counterclockwise direction. If <angle> is negative the arc sweeps clockwise. <angle> must be in the range -360.0000 through 360.0000.

The carriage accelerates smoothly from zero velocity to the speed given by the SR command, up to a maximum of about 17000 microsteps per second. If the SR command's parameter was greater than 17000, arcs are performed at 17000 microsteps/second but vectors are performed at the higher speed. The rate of acceleration is as specified by the AC command. Small radii may cause slew speed to be limited in order to limit radial acceleration; see the SR and AC commands. Continuous Path and certain microstep resolutions may also cause the slew speed to be limited; see the BC and RE commands.

The travel limits are applied as follows. The arc center is not required to be within the travel limits. If any portion of the arc itself would lie outside the travel limits or if the arc radius exceeds 32700, no arc is generated. Instead, a Position Overflow error is logged and the system behaves as though an MA (Move Absolute) to the final position had been received. (If the final position is within the travel limits, the carriage moves in a straight line to this position. Otherwise, it moves to a position on the edge of the travel limits. See the MA command.)

If mid-move Digital Outputs changes have been established via the MD or MM command, they will be executed as though the arc were "unrolled" into a straight line. In other words, the MD command's <count1> and <count2> parameters (or the MM command's <count> parameter) are treated as distances along the arc, in microsteps. If the arc length exceeds 32767 there will be a region near the end of the arc which is unreachable by positive <count> values, and another near the beginning which is unreachable by negative values. If arc length exceeds 65535 there will be a region in the middle of the arc which is unavailable for mid-move Digital Output changes; the Digital Outputs will not change in this region.

Several conditions may force the Automove System to skip some intermediate microstep positions during the generation of the arc. This skipping is done in a regular, smooth pattern and does not affect the overall shape of the arc, but it does affect the dynamic resolution and the accuracy with which the "ideal" endpoint can be reached. Also, available motor torque may be reduced.

Skipping occurs if either: the specified slew speed exceeds approximately 2200 microsteps per second; or the circumference of the arc (i.e. the actual distance traveled) exceeds 65535 microsteps. Note that other internal errors may cause small endpoint discrepancies even though none of the above conditions is exceeded and thus no microsteps are skipped.

The actual final carriage position as given by the OA command may differ from the "ideal" position by several microsteps. The Commanded Position, as given by the OC command, will not reflect this difference. Any discrepancy will disappear at the next MA or MR command. (For an exception, see the BC command.)

The AA command is ignored while the machine is in the Emergency Stopped state. If the STOP button is pressed (or an external Emergency Stop switch closes) during the arc, the motors stop immediately and the Home position reference is lost. See the STOP button in Chapter 4 and the FH command in this chapter.

The behavior of the AA command can be altered by changing the Vector Mode. See the VM command.

Example: the sequence MA 2000, 2000; AA 3000, 2000, -360; moves the carriage to (2000,2000) and then performs a clockwise circle with a radius of 1000 microsteps, centered about (3000,2000). The carriage is left at approximately (2000,2000).

Example: the sequence MA 1000, 0; AA 1000, 6000, 45; produces an arc which ends at approximately (5243,1757). But the sequence MA 1000, 0; AA 1000, 6000, -45; produces Error 6 and a vector to (0,1757), because a portion of the arc lies outside the travel limits.

AB [<flag>] **Antibacklash Vectors**

If <flag> is nonzero, antibacklash vectors are enabled. When antibacklash vectors are enabled, each move performed by the MA, MR, and MT commands and GO TO ORIGIN button is divided into two vectors. The first vector moves to a point 15 microsteps to the left of and 15 microsteps below the Commanded position. The second vector moves to the Commanded position. The effect of this extra vector is to improve endpoint repeatability by applying a consistent bias to the backlash of the mechanical system.

If mid-move Digital Outputs changes have been established via the MD or MM command, they are performed during the first vector; the Digital Outputs do not change between the two vectors or during the second vector.

The <flag> parameter is optional. If is zero or is omitted, and also at IN and power up, antibacklash vectors are disabled. They can also be temporarily disabled via the VM command.

The size of the antibacklash vector (15 by 15) can be changed via the PE command; it can be made any length or direction, or can be restricted to one axis.

The AA and AR commands do not produce antibacklash vectors. If it is necessary to be exactly at the "right" physical position following an arc, send MR 0,0; after the arc command; this generates a pair of vectors which wind up at the right spot.

Antibacklash vectors are temporarily disabled during Continuous Path motion; see the BC command.

AC [<accel>] Acceleration

The value <accel> becomes the acceleration (in thousands of microsteps per second per second) used for subsequent AA, AR, MA, MR, and MT commands, as well as Continuous Path motion, the GO TO ORIGIN vector, and the FH backoff vectors. Usable values are in the range 10 through 65530. If a microstep is 0.001 inch, these correspond to accelerations in the range 0.026 G (Gravities) through 169.8 G. (One Gravity is 386.04 inches/second/second.)

If <accel> is omitted, and also at power up and at IN, acceleration defaults to 193 thousand inches/second/second, corresponding to 0.5 G at 0.001 inch per microstep. The default value 193 itself can be changed via the PE command, so that the System "wakes up" with the desired acceleration at power up.

Acceleration is applied to **vectors** as follows:

The carriage speed increases smoothly (i.e. linearly) from zero to the step rate specified in the SR command. The carriage travels at this "slew" step rate for a while, then decelerates smoothly to zero velocity at the commanded endpoint. If the vector is diagonal the specified <accel> is the diagonal acceleration along the vector. Some vectors may be too short to reach the slew phase before they must begin to decelerate.

Acceleration is applied to **arcs** as follows:

The carriage accelerates smoothly until it reaches the specified step rate along the arc (i.e. tangent to the arc). It slews for a while, then decelerates.

Circular motion has two components of acceleration: radial (or centripetal) acceleration which is directed towards the center of the arc, and tangential acceleration which is directed along the path of motion. The Automove System budgets 0.707 of the <accel> value to each of these two components of acceleration. Since they are always at right angles to each other, the total peak acceleration of the load in any direction will never exceed the specified <accel> value. It follows that neither motor will be required to deliver more than this value of peak acceleration to the load.

Radial acceleration depends upon speed and radius. At small radii the Automove System automatically decreases the slew velocity to prevent the radial acceleration from exceeding its budgeted amount. For example, if AC 386; SR 10000; has been executed, slew rate will begin to be limited at a radius of 366 microsteps (or 0.366 inch on a typical Automove X-Y table). But if AC 39; SR 10000; is executed, the slew rate will begin to decrease at a radius of 3.66 inches. The crucial radius is the square of the step rate divided by the budgeted radial acceleration.

Acceleration is applied to a **Continuous Path move** as follows: The carriage accelerates, slews, and decelerates once for the entire Continuous Path move. In order to control radial acceleration, the slew rate is limited according to the smallest arc radius of the entire BC...EC sequence. If there are no arcs, the slew rate is as specified by the SR command. See the BC command.

AD — Abort Download Sequence Execution

This command aborts any currently-executing Download Sequence(s), leaving the machine in the idle state. The System forgets any partially-executed Download Sequences which may have invoked other sequences via the XD, XI, XU, or XW commands, or may have been interrupted via the front panel or the Digital Inputs. Any remaining repeat counts are ignored.

Normally, Download Sequences do not stop executing until the main sequence is finished. (The main sequence is the one invoked from the front panel, the Digital Inputs, or the host via a "live" XD or XI command.) The AD command is a way to preempt this normal termination.

The AD command only works if it is part of a Download Sequence. If the host computer were to send AD while a Download Sequence was executing, nothing would happen because the Automove System does not process incoming ACL commands until after the Download Sequence has finished executing. After this the AD command would execute but would have no effect.

Note, however, that the host can abort an executing Download Sequence immediately by sending the ESC.!5 command; see Chapter 6.

In most applications the AD command should be preceded by a CP if the application is using patterns, and by an EC if the application is using Continuous Path.

AM [<mode>] — Arrow Mode

This command changes the behavior of the four Arrow buttons and other buttons on the Automove front panel. If the optional joystick is connected its behavior is affected also.

The <mode> parameter is optional. If it is omitted or is zero then the Arrows control the positions of the X and Y motors. If a nonzero value is specified for <mode> then the Up and Down Arrow buttons control the Z axis motor. Other buttons are affected, too; see "Z AXIS" in Chapter 4.

You can also change the Z Arrows mode manually, by pressing the Z AXIS button. The lights in the Up and Down Arrow buttons will illuminate while the machine is in Z Arrow mode. The host computer can determine the current status (Z vs. XY) via the OS command.

The speed and direction of Z motor travel are affected by personality parameters 40 through 42; see Chapter 7. Also, the default <mode>, zero, can itself be changed via the PE command so that the System "wakes up" with the desired Arrow mode at power up.

The Z axis full/half step mode is as specified by the CZ ("Configure Z Axis") command. For example, when an Arrow button is pressed briefly to cause a single motor step, the step will be either a full step or half step according to the most recent CZ command.

The Arrow buttons cannot move the Z motor outside the range 0 through <mod> - 1 motor counts, where <mod> is the Z axis modulus as specified by the CZ command. In other words, there is no position counter wraparound as with the MZ ("Move Z Axis") command. In this respect the Arrow buttons are more like the AZ ("Absolute Z Axis Move") command.

You can teach the Z axis position to the host computer in the same way that the OT ("Output Taught Point") command teaches X and Y. There are two techniques. In the first one, the host computer sends an AM 1; then the user adjusts the Z position with the Arrows, then he strikes a key on the host computer's keyboard. The host sends AM 0; OZ ("Output Z Position") and records the result as the taught Z point.

The second technique is similar: the host sends AM 1; OT. The user adjusts the Z position and presses the TEACH button. The host reads and discards the OT coordinates, then sends AM 0; OZ to get the Z position and restore normal XY Arrows.

AP [<flag>] **Auxiliary Digital Port Select**

The AP command is used to select an auxiliary I/O port for subsequent digital inputs and outputs. The auxiliary port has two input bits (5 and 6) and two output bits (3 and 4) available for additional functions. AP1**;** is used to select the auxiliary port and then each CD, OD, ON, XI, XU, XW and VC followed by ON will operate on the bits of the auxiliary port. At power up, and on receipt of an AP; AP0; or IN; command the AP mode will be reset. Fluidmove and Easymove can use normal digital outputs and inputs after the AP mode is set to utilize the extra I/O bits. Inputs from the auxiliary port will include internal machine bits such as the PS-10 BCD thumbwheel. However, outputs to the auxiliary port are masked in such a way that internal machine functions on the other bits cannot be affected.

AR <dx center>, <dy center>, <angle> **Arc Relative**

This command is identical to the AA (Arc Absolute) command in all respects, except that the arc center location is specified as a relative offset from the current Commanded Position instead of as an absolute position.

<dx center> and <dy center> are in Calibrated Units in the range -32700.0000 through 32700.0000. The System adds them to the current Commanded Position, then multiplies by the calibration factors and applies linearity correction to give the location of the arc center. The intermediate and final results must all be in the range -32768.0000 through 32767.9999, and the arc radius must not exceed 32700 microsteps. If the arc is part of a BC...EC sequence the circumference (i.e. total distance traveled) must not exceed 65535 microsteps.

The arc begins at the current carriage position and proceeds for <angle> degrees in the counterclockwise direction. If <angle> is negative the arc sweeps clockwise. <angle> must be in the range -360.0000 through 360.0000.

For information about acceleration, step rate, travel limits, Digital Outputs changes, microstep skipping, and Emergency Stop, see the AA command.

AS[<flag>] Auxiliary Speed

This command selects an alternate servo routine which outputs "bump" changes on the AUX1 output bit each time either the X or Y stepper motors step.

AT[<flag>] Arm Timer

This command is used to determine if the machine is kept busy at all times in a benchmark test. If, during the test run, the buffer empties, a counter will be incremented once per parser cycle, indicating that a program was running too slowly to keep the machine constantly busy. AT1; will turn on the feature and zero an internal variable that is incremented each time a service loop finds the buffer empty. This variable is output after AT0; (which also turns off the feature) and if it is zero then no delays occurred. If it contained 1200 then it means that 1200 loops occurred during the run when the buffer was found to be empty. Programmers could then optimize their software and run the test again. If the number is reduced it would mean that they made their software more efficient. AT; will turn off the timer feature with no timer counter output.

AZ <z> Absolute Z Axis Move

This command moves the Z axis motor to a specified absolute location, <z>. The motion uses the acceleration, step rate, and half/full-step mode established by the CZ command. <z> can be in the range 0 through 32766.

The motor is left at full power after the move.

The AZ command uses the Z axis travel limits. The lower limit is always 0. The upper limit is <mod> - 1, where <mod> is the Z axis modulus as specified by the CZ command. If the <z> parameter is outside the range 0 through <mod> - 1 then an error is logged and the Z axis moves to the nearest limit (i.e. it moves as far as possible in the desired direction).

If the STOP button is pressed (or an external Emergency Stop switch closes) while the Z motor is moving, the motor stops immediately. Its physical position reference remains accurate unless the sudden deceleration causes the motor to slip. However, the No Z Reference status bit is set regardless. (See the OS command.)

If the machine is already in the Emergency Stopped state when the AZ command is received, the command is ignored. See the STOP button in Chapter 4 and the FZ command in this chapter.

The Z axis has no calibration factor, origin offsetting, or antibacklash vectors.

BC [<re-use>] Begin Continuous Path

This command denotes the beginning of a Continuous Path sequence. All ACL commands following this command, up until the next EC ("End Continuous Path") are part of this Continuous Path sequence. Any XY motion commands within the Continuous Path sequence are performed as a single move with one acceleration phase, possibly one slew phase, and one deceleration phase. The motion does not begin until the EC command is received.

The following commands may occur between BC and EC:

Physical action commands:

AA, AR, CD, MA, MR, SP, TD.

Non-action commands:

BP, CF, EP, IN, OC, OE, OF, OI, OL, OO, OP, OR, OS, OU, PS, SO, XD.

A maximum of 200 physical action commands may occur in one BC...EC sequence. The number of non-action commands is unlimited.

The IN command cancels the BC and all intervening commands.

The output commands OC through OU execute immediately and do not become part of the BC...EC motion sequence. The same holds true of the PS command; the machine pauses after executing that command, but before the continuous motion has begun.

Repeating The Sequence

Normally there is a precomputation delay as the physical action commands are processed, before motion begins. (See "Delays, Emergency Stop, ...", below.) If the same BC...EC sequence is to be repeated exactly, this delay can be eliminated by specifying a nonzero value for the <re-use> parameter. In this case, no EC is required. The sequence may be repeated any number of times (e.g., via BC1; BC1; BC1; ...) as long as the following conditions are met.

The previous BC...EC sequence can be re-used only if: the carriage is in exactly the same location it occupied at the previous BC; and no errors occurred in the previous sequence. Note that arcs may introduce small position errors (see "Position Errors", below); therefore, it may be necessary to put an MA command before the repeated BC to ensure exactly the right carriage location. (Note that the MA won't help if it is within the BC...EC.)

A repeated BC...EC sequence is performed at the same speed and acceleration as the original sequence, regardless of any intervening AC, SR, or VM commands.

If the <re-use> parameter is zero or is omitted then the BC is "normal" and must be followed by motion commands and EC.

Errors

If a command not on the above list is encountered; or if an otherwise legal command causes an error such as a bad parameter, wrong number of parameters, or a travel limits violation; or if the 201st physical action command is encountered: the bad command is not executed. Instead, the error is logged and all remaining physical action commands until the EC are ignored. Then the System executes those physical action commands which preceded the bad command, plus all non-action commands.

Acceleration and Step Rate

The System uses the acceleration specified by the most recent AC command, and the step rate specified by the most recent SR command. (Any AC or SR command must occur ***before*** the BC command in order to affect Continuous Path motion.)

However, the VM command may cause the System to use the alternate acceleration and step rate. (Such a VM must occur ***before*** the BC command.) See personality parameters 34 and 35 in Chapter 7.

If a given BC...EC sequence contains any arcs, the acceleration is set to 0.707 of the specified value, in order to "budget" radial and tangential acceleration. Also, the slew speed may be decreased according to the smallest arc radius in the sequence, in order to limit radial acceleration. See the discussion under the AA command. The maximum achievable step rate for a Continuous Path with arcs is approximately 16000 microsteps per second.

If the BC...EC sequence contains no arcs, the specified acceleration and step rate values are used, as-is. The maximum step rate without arcs can be as high as 65535 microsteps per second under some conditions; see below.

The step rate is limited to lower values if certain microstep factors have been specified; see the RE command. If the SR command specifies a higher step rate, the System uses the highest achievable rate.

If the step rate has been set higher than about 2100 microsteps per second, or if the overall length of the BC...EC sequence is more than 65535 microsteps, the System automatically skips some intermediate microstep positions during the Continuous Path motion. This skipping is done in a regular, smooth pattern and does not affect the overall motion. However, it does affect the dynamic resolution and the accuracy with which the "ideal" endpoint can be reached. Available motor torque may be reduced.

The sequence should not contain more than one or two CD, TD, or SP commands between any given two vectors or arcs. The System takes a small amount of time to execute the CD, TD, or SP; if too many are executed all at once, there will be an extra-large delay between two microsteps, and the motors may slip.

Path Smoothness

At high speeds the move sequence path must be smooth. In other words, the vectors and arcs must be arranged so that there are no sudden changes of direction. Where the direction must change substantially, an arc should be used. If this rule is violated (for example, a square with sharp corners) the stepping motors may slip, with a loss of the position reference.

If path smoothness is maintained, the maximum torque demanded of the stepping motors will be related to the specified acceleration value; see the discussion of radial and tangential acceleration under the AA command.

At low speeds the stepping motors have more torque, so sudden changes of direction may be allowable. The speed limit depends on how sharp the angles are, how much mass is attached to the carriage, and how much shaking or jarring of the entire mechanical system is allowable.

Length Limitations

The maximum length of any arc resulting from a single AA or AR command during a Continuous Path sequence is 65534 microsteps. The arc length is the distance traveled along the arc, which is the radius times the sweep angle (in degrees) times pi/180. Thus, the maximum radius of a single-command 360-degree arc is 10430 microsteps. (Longer arcs can be produced by dividing them into several AA or AR commands.)

The total length (i.e., the sum of all the vector and arc lengths) of a given Continuous Path move cannot exceed a certain number of microsteps. This number is proportional to the smaller of the X and Y microstep factors (see the RE command), and is limited to 507888 microsteps if the sequence contains any arcs.

The following table shows a few representative limit values:

Table 8 - Arc Length Limitations

Smaller RE Microstep Factor	Longest Sequence With Arcs	Longest Sequence Without Arcs
1.0000	65534 microsteps	65534 microsteps
2.0000	131068	131068
4.0000	262136	262136
8.0000 (default)	507888	524272
16.0000	507888	1048544
32.0000	507888	2097088

Digital Inputs And Outputs

The entire Continuous Path sequence is treated as though it were a single vector or arc, for the purposes of the following commands: MD, MM, MN, PD, and VM.

For example, suppose an MD command has set up a Digital Outputs change which is to occur 1000 microsteps after the beginning of the move, plus another which is to occur 200 microsteps before the end. (The command might be: MD 1000, 2, 2, -200, 0, 2;.) The Digital Outputs will change 1000 steps after the beginning of the BC...EC sequence, no matter how many vectors or arcs have already been executed. Then the Outputs change again, 200 steps from the end of the entire sequence. It is as though the Continuous Path sequence were "unrolled" into a straight line.

The CD and TD commands are executed wherever they appear within the BC...EC sequence. For example, the sequence:

```
BC;  MR 100,0;  CD 255;  MR 100,0;  EC;
```

produces a move of 200 microsteps to the right. Exactly in the middle of the move, all of the Digital Outputs are turned on.

If a given Continuous Path sequence is executed with MD or MM active and it also contains CD or TD commands, both kinds of Digital Outputs changes occur as specified.

Interrupts and Front Panel

For the purposes of interrupts caused by the Digital Inputs, the entire BC...EC sequence is treated as a single move. In other words, no interrupt can occur from when the BC is executed until after the Continuous Path motion is complete. See Chapter 8 Using the Digital Outputs and Inputs.

Once the EC has executed and motion has begun, no front panel button except STOP, RESET, or PAUSE will be acted upon until the BC...EC sequence is complete. The PAUSE button is "remembered" and the machine pauses upon completing the Continuous Path move. STOP and RESET occur immediately.

If any front panel XY motion occurs between BC and EC (due to the Arrow buttons or GO TO ORIGIN) an error is logged and no Continuous Path motion is performed.

Delays, Emergency Stop, Antibacklash, Host Interrogation

As the physical action commands are processed there is a delay before motion begins. The amount of delay depends on how many vectors and arcs are contained in the sequence; the delay is about 28 seconds for 200 downloaded, rotated MA vectors and about 16 seconds for 200 downloaded AA arcs. (If MR or AR is used, or if the commands are being sent from the host, the delay will be even longer.) This delay can be eliminated if the same Continuous Path sequence is to be repeated; see "Repeating The Sequence", above.

The entire BC...EC sequence is ignored if it occurs while the machine is in the Emergency Stopped state. If the STOP button is pressed (or an external Emergency Stop switch closes) during the motion, the motors stop immediately and the Home position reference becomes inaccurate. See the STOP button in Chapter 4 and the FH command in this chapter.

Antibacklash vectors are temporarily disabled during a Continuous Path sequence. If they were enabled at the time the BC was executed, they will automatically become re-enabled after the EC.

The host computer can inquire how many physical action commands have been processed since the BC command, and how many are allowable. See the ESC.S6: and ESC.S7: commands in Chapter 6.

Position Errors

As with single (non-Continuous Path) arcs, each arc may leave the carriage several microsteps away from the "ideal" arc endpoint. These position discrepancies are cumulative during a single Continuous Path sequence; in other words, the position error may grow larger as more arcs are executed.

Continuous Path vectors do not increase the position error, but neither do they eliminate it. The position error is eliminated at the first vector after the EC command. The most straightforward way to be sure the carriage is at the "ideal" point is to execute an MR 0, 0; after the EC. This may be unnecessary, since any subsequent vector will eliminate the position error.

Examples

The following ACL sequence generates a "race track" -- an oval- shaped closed curve:

```
MA 3000, 2000;  BC;  MR 2000, 0;  AR 0, 1000, 180;

MR -2000, 0;  AR 0, -1000, 180;  EC;
```

There are four moves between BC and EC -- two vectors (MR) and two arcs (AR). The carriage moves to 3000, 2000 and stops; then it moves in a nonstop counterclockwise oval, ending up where it started.

The following example shows how Continuous Path can be used to produce a move of great length. Suppose your application uses a leadscrew table and, for high endpoint resolution, you are using high microstep factors (via RE 32, 32;). Normally, the overall travel limit of the Automove System is 32766 microsteps in each axis. Suppose, however, that you need to move 80000 microsteps from the Home switches in order to reach the "active area" of your application. The following sequence finds the Home switches and then moves 80000 microsteps away in each axis, in a single move:

```
FH;  SO 0, 0;  CF 1, 1;  BC;  MA 30000, 30000;  SP 0, 0;

MA 30000, 30000;  SP 0, 0;  MA 20000, 20000;  EC;
```

(The SO and CF have been included to simplify the application.) The first MA leaves the carriage at 30000, 30000 microsteps. Then the SP "fools" the System into thinking it is at 0, 0 microsteps. Without stopping, the carriage moves another 30000 microsteps in each axis, then another 20000.

The next example shows how to use patterns and Download Sequences to repeat a shape numerous times within one Continuous Path sequence. We assume that a fluid dispensing valve is connected to Digital Output bit 1.

```
BD 0;  MA 1000, 4000;  SR 1000;  PD .05, 2, 2;

MD -50, 0, 2;  BC;  XD 100, 20;  EC;  ED;

BD 100;  BP;  MA 100, 100;  MA 300, -100;  MA 500, 0;

EP;  ED;
```

In the first Download Sequence we move to 1000, 4000 and set the speed low (SR 1000;) so that the machine can make sharp corners without slipping the stepping motors. Then we set up pre-move and mid-move Digital Outputs changes to turn the valve on 0.05 seconds before motion begins and turn it off 50 microsteps before motion ends. The BC indicates the beginning of nonstop motion. Without stopping, we execute the second Download Sequence (i.d. number 100), repeating it 20 times.

Download Sequence 100 makes a zigzag move to the right, whose overall X-axis travel is 500 microsteps. A pattern (BP...EP) is used so that the absolute moves will be referenced to a new Origin each time the move is executed. When this pattern has been repeated 20 times, the carriage will have moved a total of 10000 microsteps to the right, and a total of 60 physical action commands will have been executed; i.e., 3 MA's per pattern.

To execute this downloaded example, simply send XD 0; to the Automove System; press the Execute button with the thumbwheel switch set to 0; or press FAST/PAUSE on the front panel.

To repeat the entire sequence in the above example without any precomputation delay:

```
VM 15;  MA 1000, 4000;  VM 0;  BC 1;
```

The VM 15; prevents the dispensing valve from turning on during the move back to the starting point, (1000, 4000). Then VM 0; allows dispensing to resume. The BC 1; need not be followed by an EC;.

BD[<id> [, <rept>]] **Begin Download**

This command puts the System into the Downloading mode. All ACL commands following this command, up until the next ED ("End Download") command, are stored in the System's non-volatile download memory instead of being executed immediately. The only ACL commands that cannot be stored in a Download Sequence are BD and ED.

Be sure to flip the backpanel Write Protect switch down before downloading. If you don't the System logs an error and ignores all ACL commands up until the next ED command.

<id> is the identifying number of the Download Sequence to be defined; it must be an integer in the range 0 through 255, and it identifies which Download Sequence is to be created. <id> is optional; if omitted it defaults to zero (i.e. Download Sequence 0 will be defined). If <id> is outside the range 0 through 255 an error is logged and Downloading mode is not entered.

Any existing Download Sequence stored under the specified <id> number is erased when the BD command is executed. Thus it is not necessary to erase an old sequence before sending a new one.

The Downloading mode is canceled, and the Download Sequence is delimited, by executing an ED command or sending ESC.!5:. After the Downloading mode has been canceled all ACL commands are executed normally as soon as they are received. However, if you reset or turn off the power without sending ED or ESC.!5: then download memory is left in a corrupt state and the Automove System will detect this at the next power up. (See Appendix A Power Up Actions.) You can usually fix this by simply sending an ED, then cycling the power again.

If a second BD command is sent without sending an ED for the first Download Sequence, the first sequence is delimited (i.e. terminated) and the second one is begun.

If the IN command is sent during Downloading mode it does not cancel Downloading mode but instead is stored as part of the Download Sequence.

Invoking

After it has been stored via BD...ED the Download Sequence can be invoked, or executed, by the XD ("Execute Download Sequence") command. When a Download Sequence is invoked the stored ACL commands are executed as though they had been received through the RS-232C port. Download Sequences can call each other via XD up to a maximum nesting depth of 12 sequences.

Several other commands can be used to invoke Download Sequences. They are: XI ("Execute If"), XU ("Execute Until"), and XW ("Execute While").

The first ten sequences, 0 through 9, can be invoked from the front panel. Download Sequence 0 is invoked by pressing the PAUSE button while holding the FAST button depressed; sequence 1 is invoked by pressing TEACH while holding the FAST button. (The FAST button is the unlabeled button in the center of the four Arrow buttons.) Any sequence from 0 through 9 can be invoked by setting the Program Select thumbwheel switch to the desired i.d., then pressing the GO button. Download sequences can also be invoked via the Digital Inputs -- see personality parameters 36 through 38; also see Chapter 8 Using the Digital Outputs and Inputs.

All front panel buttons work normally during the execution of a Download Sequence. For example, PAUSE causes execution to be suspended until PAUSE is pressed again; STOP causes all motion to cease until a CS command is executed. Any ACL output commands (e.g. OA, OC, OS) send their outputs to the host, as usual. (Of course, if no host is connected, the outputs go into the "bit bucket".) ACL errors in the downloaded commands are logged as usual. When the erroneous command is executed the System immediately sends a "?" to the host. (Some errors are caught during downloading and thus do not occur while the sequence is executing. See the **Storage** section, later in the chapter.)

Repeating

<rept> is the repetition count, an integer in the range 0 through 65535. It is optional but cannot be specified unless <id> is also specified. If <rept> is omitted or has the value 1 the download sequence executes once each time it is invoked. If <rept> is greater than one the sequence executes the specified number of times. If <rept> is zero the sequence repeats forever until the power is turned off or the machine is reset, or ESC.!5: is sent from the host.

The BD command's <rept> parameter can be overridden by the XD, XU, and XW commands.

External Control

Escape sequences (see Chapter 6) cannot be stored as part of a Download Sequence. If an escape sequence is received while the System is either in Downloading mode or is executing a Download Sequence, the escape sequence is executed immediately.

If ACL commands are received from the host while a Download Sequence is executing they are stored in the input buffer and are executed upon completion of the Download Sequence. If the Download Sequence is repeating infinitely the ACL commands will never be executed, unless the Download Sequence is halted.

The execution of a Download Sequence can be prematurely halted four ways: by resetting the machine or turning off the power; by executing an AD command which is contained in the Download Sequence; by sending the ESC.!5: command (see Chapter 6); or by exceeding the maximum nesting depth (see the XD command).

Storage

In the 16K non-volatile memory 13894 bytes are available for the storage of Download Sequences. The rest of the 16K bytes is consumed by overhead, by the personality parameters, by the XY Linearity Correction Table, and by the ACL variables.

The available download memory can be divided in any proportion between the Download Sequences, as long as the total space used does not exceed the amount available. If this happens an error is logged and any further ACL commands are ignored, up until the next ED command.

The amount of download memory not in use can be determined via the ESC.S2: command. The amount consumed by any particular Download Sequence can be determined via ESC.S5;n:, where n is the id. See Chapter 6.

Each two-letter ACL mnemonic consumes one byte of download memory. The parameters to the AA and MA commands consume two bytes each if they are integers in the range 0 through 16383, and otherwise five bytes each. All other parameters consume two bytes each if they are integers in the range -8192 to 8191, and otherwise five bytes each. Two bytes are allocated to overhead in each Download Sequence.

For example, assume a Download Sequence is to consist only of MR commands with small parameter values. Then each command consumes five bytes (one for the mnemonic and two each for the X and Y parameters). If the Download Sequence contains five MR commands it will need 2 + (5 x 5) = 27 bytes of download memory.

During downloading if an illegal ACL mnemonic (i.e. an unrecognized command) occurs an error is logged and the command and its parameters are ignored. The System does not check for the correct number of parameters with each ACL command, nor does it check the ranges of the parameters.

If any parameter is above 65535.9999 it is stored as 65535.9999, and if below -32768.0000 it is stored as -32768.0000. No error is logged at the time the parameter is stored, but if the stored value is outside the allowable range for the particular command an ACL error will occur when the Download Sequence is executed. (Some of these errors may be missed -- see below.) Similarly, if too few or too many parameters are stored, an ACL error occurs when the sequence is executed.

Stored values can be interpreted in two ways, depending on what the ACL command is expecting. If the command expects -32768 through 32767 then values in the range 32768.0000 through 65535.9999 are interpreted as the specified value minus 65536. If the command expects 0 to 65535 then values below zero are interpreted as the specified value plus 65536. Because of this relaxed range checking it is a good idea to thoroughly debug your ACL sequences before downloading them.

To erase a Download Sequence and make its memory available for re-use, simply send the BD command and immediately follow it with the ED command. For example: BD 13; ED; erases Download Sequence 13. If the new Download Sequence is to have the same number as the old, it is unnecessary to first erase the old.

If the Automove System is reset or turned off or the power fails while a sequence is being downloaded, the download memory will be left in an invalid state. The next time you turn on the System it will enter the Emergency Stopped state. If you clear the Stop and execute the Download Sequence anyway, an ACL error may occur when the sequence is invoked, or the last command of the sequence may behave strangely, or part of the download memory may become unavailable. The first two problems can be corrected by simply re-downloading the sequence that was interrupted.

If you believe you have erased all of the existing Download Sequences but the ESC.S command still shows that some download memory is in use, then either: one or more sequences are still defined; or, the BD command was interrupted by a reset or power failure. To erase the entire download memory and reclaim the lost space, use the ESC.!7: command; see Chapter 6.

Erasing a single Download Sequence can take as long as 1.5 seconds if the download memory is nearly full. The fastest way to re- download all of the sequences in the machine is to first erase all old sequences via ESC.!7:, then send the new versions of the sequences.

Where long Download Sequences are involved, after the ED command your program should send an OE ("Output Error Code"), read the result back, and display it on the screen. This serves two purposes: it tells you that the downloading is complete so it is OK to turn off the power; and it tells you whether your sequence was too large for the available memory. (If it was, you will get a nonzero error code.)

Example: the command sequence BD; FH; MA 2000, 2000; AR 1000, 0, 360; ED; sets up Download Sequence 0 to be a Find Home, a Move Absolute, and an Arc Relative. After sending this sequence, press the FAST button and, while holding it, press the PAUSE button. The sequence will execute once. Disconnect the host computer or turn the Automove System's power off, then back on. Press FAST/PAUSE again, and the sequence will execute again.

BP [<angle>] **Begin Pattern**

This command denotes the beginning of a pattern. All ACL commands following this command, up until the next EP ("End Pattern") command, are part of this pattern.

The <angle> parameter is optional. If specified, it sets a rotation angle, in degrees, to be applied to all XY motion commands within the pattern. The range of <angle> is -32768.0000 through 32767.9999. Rotations less than zero or greater than 359.9999 degrees are reduced modulo 360 to the corresponding angle in the range 0 through 359.9999 degrees.

The rotation is centered at the current Origin. If you do not execute an SO ("Set Origin") command or press the Set Origin button while the pattern is executing then the center of rotation is wherever the carriage was when the BP was executed.

The rotation angle is relative to any current rotation. This means that nested patterns may use nested rotations. At the next EP ("End Pattern") command the System reinstates whatever rotation angle was in effect before the BP was executed.

The rotation is performed in Calibrated Units space; i.e., before calibration factors or linearity correction are applied.

Why Use Patterns?

Patterns are useful for step-and-repeat operations where a sequence of moves is repeated identically in several places. Step-and- repeat could be done using relative moves (i.e. MR and AR commands) but it is easier to create, debug, and modify a motion sequence using absolute moves (MA and AA commands).

Patterns allow you to create a move sequence using absolute moves (for example, by teaching the points) and then later translate the move sequence to a different location without changing the parameters of the MA and AA commands. This can be very useful in downloaded sequences.

The SO ("Set Origin") command can also be used for translating a set of absolute moves, instead of BP. However, you may wish to use the "main" origin for other purposes; for example, to register the workpiece location on the platen, or account for different tool offsets.

Patterns are also useful for performing translate-and-rotate operations to account for misregistration and rotation of the workpiece. With nested patterns, for example, separate translations and rotations can be applied to a panel of substrates, an individual substrate, and an individual component attached to a substrate.

Another use for patterns is when platen-relative and workpiece-relative motions must be interleaved, as shown in the final example.

How It's Done

When the BP command is executed the System saves the location of the Calibrated Units Origin and establishes a new Origin at the present carriage location, as though the SET ORIGIN button had been pressed. The old pattern rotation angle is saved and the specified <angle> parameter, if any, is added to the old angle. The Origin Change status bit is cleared. All subsequent AA, MA, OC, OT, and SP commands are referenced to this new Origin position. The rotation angle affects these commands, plus MR and AR.

When an EP command is executed the System restores the saved Origin location and rotation angle, so that absolute XY commands will again be referenced to the original Origin.

Nesting

Patterns may be nested up to a depth of 12. In other words, one pattern may contain other sub-patterns within itself, which may contain sub-sub-patterns, etc. The allowable nesting depth is exceeded, for example, if more than twelve BP commands occur with no intervening EP commands. When this happens the thirteenth and subsequent BP commands cause an error to be logged but the Origin location does not change.

On the other hand, one pattern may contain any number of sub-patterns within itself as long as they are not nested to a depth greater than 12.

Example:
BP...BP...BP...BP...BP...BP...BP...BP...BP...BP...BP...BP...EP...EP...EP...EP...
EP...EP...EP...EP... EP...EP...EP...EP...EP is illegal because the nesting depth is thirteen. But BP...EP...BP...EP... can go on forever because the sub-patterns are not nested.

Clearing Things Up

The CP ("Clear Patterns") command cancels all active patterns, such that the "main" Origin position is active and the pattern rotation is zero. This action is also forced by power up and by the IN command.

If an SO ("Set Origin") command or the front-panel SET ORIGIN button changes the origin while a pattern is active, the new Origin remains in effect only until the pattern is terminated by an EP. Also, if the GO TO ORIGIN button is pressed while a pattern is active the carriage moves to the pattern's Origin, not the "main" Origin.

The host can find out the current level of pattern nesting by sending the ESC.!S4: command; see Chapter 6. The current rotation angle can be interrogated via the OG ("Output Angle") command.

Examples

Example: the sequence MA 2000, 2000; BP; MA 500, 1000; MA 1000, 0; MA 0, 0; EP; moves the carriage to 2000, 2000 and then executes a small triangle. If you change the first command to MA 4000, 2000; the System executes the same triangle at a different location.

Example: the sequence BD 0; BP; MA 500, 1000; MA 1000, 0; MA 0, 0; EP; ED; sets up Download Sequence 0 to contain the triangle from the first example. If you use the Arrow buttons to move the carriage to the starting point, then press FAST/PAUSE, the triangle will be executed at the present carriage location.

Example: MA 3000, 1000; BP; MA 1000, 0; BP; MA 0, -200; MA 0, 200; MA 0, 0; EP; MA 2000, 0; BP; MA 0, -200; MA 0 200; MA 0, 0; EP; EP; moves the carriage horizontally to the right from 3000, 1000, pausing twice to jog below and above the main motion.

Example: suppose your application uses a tool holder bolted to the Automove platen. In the middle of the move sequence you desire to fetch a tool from the tool holder. However, the origin has been moved manually to account for the workpiece location on the platen, so you cannot simply use an MA to move to the tool holder. Use the following sequence: BP; SO; MA x, y; ... EP; where x,y is the absolute location of the tool holder and "..." represents whatever commands are necessary to fetch the tool. The SO; sets the Origin to the Home Position (which is fixed with respect to the platen) and the EP reinstates the original Origin.

Example: MA 0, 0; BP 10; MR 1000, 0; EP; MR 1000, 0; moves to the Origin, then sets up a 10-degree rotation. A 1000-unit-long rotated vector is performed; this leaves the carriage approximately 985 microsteps in X and 174 microsteps in Y from the Origin. The EP; reinstates zero rotation, so that the second MR 1000, 0; moves directly to the right. The carriage is now at approximately 1985 microsteps in X and 174 microsteps in Y from the Origin.

Example: for an example of how to set a new pattern Origin without first moving the carriage to that location, see the SC ("Scale") command.

CD [<new> [, <which>]] **Change Digital Outputs**

This command causes an immediate change in any or all of the 8 Digital Outputs. See also the TD ("Toggle Digital Outputs") command.

<new> is interpreted as an 8-bit number, with a useful range of 0 through 255. The number is written immediately to the Digital Outputs. Each bit corresponds to one Digital Output; if the bit is 1, the Digital Output is set True; if zero, it is set False. The correspondence with Digital Outputs is as follows:

Table 9 - Changing Digital Outputs

Bit	Decimal Value	Digital Output
0	1	0
1	2	1
2	4	2
3	8	3
4	16	4
5	32	5
6	64	6
7	128	7

For example, to turn on Digital Outputs 3 and 4 and turn off all the others, set <new> to 24, which is the sum of 8 and 16.

The <which> parameter is also optional, but may only be specified if <new> is specified. <which> controls which Digital Outputs are affected by the value of <new>. If a particular bit of <which> is 1, the corresponding Digital Output is immediately set to the corresponding bit of <new>, and the old value of that Digital Output is forgotten. If a bit of <which> is 0, the corresponding Digital Output retains its old level and the corresponding bit in <new> is ignored.

If <which> is omitted it defaults to 255 (i.e. binary 11111111), meaning that all 8 Digital Outputs are affected by <new>.

After the Digital Outputs have been given the new command, the machine waits a certain amount of time before processing the next command. This delay is specified via the WD command; it should be great enough to allow any associated external hardware to complete its action, if necessary. (Alternatively, the WA command can be used to produce a unique value of delay after each CD.)

If <new> is omitted, and also at Emergency Stop, IN, and power up, the Digital Outputs are all set to the False state.

While the machine is in the Emergency Stopped state the CD command is ignored; all Digital Outputs remain in the False state.

If you need a precise delay between a Digital Outputs change and the beginning of the next vector or arc, use the PD command. Also, the Digital Outputs can be made to change in the middle of an arc or a vector; see the MD and MM commands.

If <new> or <which> is above 255, the least significant 8 bits are used.

The current value being applied to the Digital Outputs can be determined via the OD command.

For more information about the Digital Outputs see Chapter 8 Using the Digital Outputs and Inputs.

Example: CD 5; turns on Digital Outputs 0 and 2, while turning off all others. (The integer 5, or binary 00000101, has bits 0 and 2 set, all others cleared).

Example: assume the Digital Outputs are at 5, as above. Then execute a CD 0,4; (i.e. <new> is binary 00000000 and <which> is binary 00000100). The effect is to set the Digital Outputs to binary 00000001. Only bit 2 changes since only bit 2 of <which> was set.

CF [<xcal>, <ycal>] **Calibration Factors**

The values of <xcal> and <ycal> become calibration factors for the X and Y axes, respectively. A calibration factor is the number of microsteps the machine actually moves when commanded to move one Calibrated Unit; for example, the parameters of the MR command are multiplied by the calibration factor to determine how many microsteps the carriage is to move. The allowable range for <xcal> and <ycal> is 0.0000 through 32767.9999.

At power up and at IN, the calibration factors are set to the default values, as explained below. Omitting <xcal> and <ycal> from the CF command establishes the same defaults. <xcal> and <ycal> must be either both specified or both omitted.

The default values are chosen as follows: if the controller is shipped with an X-Y table, the table travel is measured and the calibration factors are set to values near 1.000, such that a calibrated unit is exactly 0.001 inch. If the controller is shipped by itself the default calibration factors are set to 1.0000; hence Calibrated Units are interpreted directly as microstep values, subject to being offset by the Origin location, the linearity correction, and pattern rotation. The default values themselves can be changed via the PE command, so that at power up the machine "wakes up" with any desired value.

The calibration factors affect only the AA, AR, MA, MR, OC, OT, and SP commands. They do not affect the travel limits, the Origin position, the MT command, the acceleration, the step rate, or mid- move Digital Outputs changes; these are all set directly in microsteps. See the TL, SO, MT, AC, SR, MD, and MM commands.

The execution of a CF command does not affect the existing microstep locations of the carriage, the origin, the travel limits, and the taught point. However, the CF command affects the values reported by OC after front-panel Arrow button or GO TO ORIGIN motion since such motion causes the Commanded position to be recomputed.

Very small values of <xcal> and <ycal>, say in the range 0.0001 through 0.01, do not produce accurate results. For example, the sequence CF 0.001, 0.001; MA 10000, 10000; OC; produces the result 10000,10000 as expected. But then if the Arrow buttons are used to move the carriage away one microstep and then back to the original position, an OC will give 9929.697,9929.697.

This result differs from the first one because front-panel carriage motion forces the Commanded Position to be recomputed from the Actual Position by dividing a microstep value by the calibration factor. The magnitude of the error is related to the way the calibration factors are represented internally; larger values of calibration factor give proportionately smaller errors.

The current calibration factors can be read via the OF command.

CP — Clear Patterns

This command cancels all active patterns and reinstates the "main" Origin position; i.e., the Origin that was in effect when the first BP ("Begin Pattern") command was executed. The pattern rotation angle is set to zero.

The action of CP is performed automatically at power up and at IN.

CR <x index>, <y index>, <x correction>, <y correction> — Correction

This command installs a pair of correction offsets into the Automove System's XY Linearity Correction Table. This table is stored in the non-volatile memory so that it retains its values even when the power is turned off. Be sure to flip the backpanel write protect switch down before executing the CR command.

The Automove System consults the Linearity Correction Table whenever a position command is received via an AA, AR, MA, MR, or SP. The commanded XY position is corrected by a small amount to produce an accurate physical motion. The table is used "in reverse" by the OC and OT commands to convert a physical position into the corresponding Calibrated Unit position.

The Linearity Correction Table is used in conjunction with the calibration factors as established by the CF command. In an MA command, for example, the XY Calibrated Units position is first rotated by the pattern angle. This rotated position is then multiplied by the calibration factors and the (reverse-corrected) Origin is added. Then the Correction Table is applied. Thus, the correction is with respect to the Home switches, not the Origin. This means that the corrections can be measured over the travel of the physical system (e.g. Automove XY table) and will be appropriately applied even though the Origin has moved.

For more information about the CR command and its parameters please see Chapter 9 Linearity Correction.

The current values stored in the Linearity Correction Table can be read via the OR command. The entire table can be set back to its factory default state (i.e. no corrections) via the ESC.!8: command; see Chapter 6.

CS Clear Stop

The CS command turns off the STOP light and clears the Emergency Stopped state and the Emergency Stopped status bit, thus enabling subsequent motor motion. The Motor Slipped status bit is also cleared and the SLIP light is turned off. Any STOP or slip that has been logged but not yet acted upon is forgotten. (See FP.)

The Emergency Stopped state is entered when the user presses the STOP button on the front panel or when an external Stop switch actuates. It can also be triggered by the optional Step Verify circuit or by a corrupt non-volatile memory (see below). During the Emergency Stopped state the front-panel STOP light blinks and the machine ignores all physical action commands (AA, AR, AZ, CD, FH, FZ, MA, MR, MT, MZ, TD, WA, WN, Arrow buttons, GO TO ORIGIN button, manual re-reference). The Paused state is canceled and prevented by the Emergency Stopped state.

The Motor Slipped state is entered when the optional Step Verify circuit detects a slip of the X or Y stepping motor. The front- panel SLIP light begins to blink and the machine typically executes an Emergency Stop. A Download Sequence may be automatically triggered. The CS command performs a half-second delay before clearing the Motor Slipped state. These behaviors can be modified via personality parameters 49, 50, and 51; see Chapter 7.

CS has no effect if it is executed while the machine is in neither the Emergency Stopped state nor the Motor Slipped state.

Also, CS is ignored if the Stop switch is still actuated. (The host can check this via ESC.O.) If your system contains a Stop switch which is actuated by motor travel and you need to back the motor away from the switch after an Emergency Stop occurs, you can use the following trick: temporarily disable Emergency Stop via the FP command, execute a CS, IN, or ESC.!2: to clear the Stop, move the motor away, and execute another FP to re-enable Emergency Stop.

Note: if the machine enters the Emergency Stopped state automatically at power up but the Stop switch is not actuated, this indicates that the non-volatile memory is corrupt. For more information see Appendix A Power Up Actions.

The host can detect the Emergency Stopped or Motor Slipped state via the OS command or the ESC.o: command. See "STOP" in Chapter 4.

CZ [<rate> [, <mod> [, <halfflag> [, <accel>]]]] **Configure Z Axis**

This command configures the Z axis speed, position range, step size, and acceleration to be used with the AZ and MZ commands. The step size also affects the FZ command and Z arrow motion; see the AM ("Arrow Mode") command.

<rate> becomes the Z axis slew step rate, in steps per second. Usable values are in the range 1 through 10000. Values above 10000 are treated as 10000. Zero is treated as 1. At power up and IN, and when no parameters are specified, <rate> defaults to 100 steps per second. See <halfflag>, below.

In AZ and MZ moves, the step rate increases smoothly from zero to the specified value, at an acceleration rate determined by the <accel> parameter. For short moves the speed may never reach the slew rate.

<mod> is optional; it may only be specified if <rate> is specified. <mod> sets the Z axis modulus. This number is one greater than the maximum value which will ever appear in the Z axis position counter. The allowable range for <mod> is 0 through 32767.

<mod> acts as a travel limit to the AZ command and to Z axis Arrow button motion. Where the MZ command is concerned, <mod> acts as a wraparound modulus.

For example, if the modulus is set to 1000, the Z axis counter will always be in the range 0 through 999. If the Z axis position counter is 900 and an MZ 200; command ("Move Z +200 counts") is received, the Z axis motor takes 200 steps in the positive direction and the position counter is 100. (In other words, an OZ command would give the value 100.)

On the other hand, suppose that instead of the MZ 200; you send AZ 1100; . A Position Overflow error is logged and the Z motor takes 99 steps in the positive direction. The position counter is 999.

If your system has infinite Z axis travel (for example, a conveyer belt) you can use the modulus to specify the number of steps per motor revolution, so that the counter always has the same value at a given point in the revolution. In this case you should use MZ, not AZ. In a system with finite (i.e. limited) Z axis travel you can use <mod> as a travel limit to prevent physical damage due to programming errors or out-of-range data points. In this case you should use AZ, not MZ.

<mod> defaults to 32767 if it is omitted, and also at IN and power up.

<halfflag> is also optional, and may only be specified if <rate> and <mod> are specified. If <halfflag> is nonzero, the Z axis motor will be half-stepped; if zero, full-stepped. (In half- stepping, intermediate positions occur where only one phase receives power.) A half step and a full step both change the Z axis position counter by one count, although a full step causes twice as much physical motion.

Note that full stepping can be done with power applied to either one phase at a time or two. The Automove System uses two-phase full stepping.

<halfflag> defaults to 1 (i.e. half step mode) if it is omitted, and also at IN and power up.

<halfflag> is ignored if the High Resolution Z Axis option is installed. With this option the step size is fixed. However, all other CZ parameters retain the same meanings as described here.

<accel> is optional, and may only be specified if all preceding parameters are specified. It becomes the Z axis acceleration rate, in thousands of steps per second per second. (Half steps and full steps are taken at the same rate.) Usable values are in the range 1 through 10000; the smallest achievable acceleration is actually 5000 steps per second per second, corresponding to a parameter value of 5. Acceleration is used only for AZ and MZ. Arrow button motion does not use acceleration and FZ uses the value specified in personality parameter 56.

At power up and IN, or if <accel> is omitted, it defaults to 40, corresponding to 40000 steps per second per second.

All of the default values for the CZ command can be changed via the PE command, so that the System "wakes up" with the desired values at power up. See personality parameters 44 through 47 in Chapter 7.

EC End Continuous Path

This command denotes the end of a Continuous Path sequence. When the EC command is encountered, the entire Continuous Path motion sequence begins to execute. See the BC command.

If no Continuous Path sequence is active when the EC command is executed then the command has no effect.

ED End Download

This command cancels the Downloading mode, thus specifying the end of a Download Sequence which was started with the BD command. See the BD command.

If the System is not in the Downloading mode when the ED command is received then the ED command has no effect.

The Downloading mode is also canceled by other events related to downloading; see the BD command. The IN command does not cancel the Downloading mode, but instead is stored as part of the Download Sequence.

If you have sent a BD you should be sure to send an ED before resetting the machine or turning off the power. Otherwise the download memory is left in a corrupt state, and at the next reset or power up the machine will automatically become Emergency Stopped. (You can use ESC.O to verify that this has happened.) In this case you must send ESC.!7: to erase the entire download memory, then re-download. See Appendix A Power Up Actions.

EP End Pattern

This command denotes the end of a pattern which was begun with a BP ("Begin Pattern") command. If no pattern is active when the EP command is executed then an error is logged.

All active patterns can be canceled via the CP ("Clear Patterns") command.

When the EP command is executed the System reinstates the Origin location and pattern rotation angle that were in effect when the matching BP command was executed. The Origin Change status bit is cleared.

ES <quotechar> [<char> [, <char> ...]] <quotechar>

Execute Escape Sequence

This command causes the Automove System to interpret a string of characters as an Automove escape sequence, as though the characters had been received over the RS-232C port following an ASCII Escape (ESC, decimal code 27). The Escape itself is omitted from the string of characters.

Note that, normally, escape sequences are executed as soon as they are received by the Automove, whether or not it is currently downloading a move sequence. The ES command allows an escape sequence to be executed at the time the download sequence is executed, even though the host computer is not connected.

The <quotechar> is any ASCII character other than NULL or Escape. (As with all ACL, each character used in an ES command must not have bit 7 set; i.e., the decimal equivalent must be less than 128.) You may choose any character to delimit the string, as long as that character does not appear within the string itself. The suggested character to use under normal conditions is the ASCII "double quote" itself ("). If a double quote must appear in the string, some other character (such as single quote or apostrophe) should be used to delimit the string. Blanks preceding the first <quotechar> are ignored.

When the ES command is executed, the System processes each of the <char> characters as part of an escape sequence. A maximum of 64 characters may appear between the two <quotechar> delimiters. If too many characters appear, an error is logged and the excess characters are treated as normal ACL commands.

Don't forget the closing <quotechar> -- if you don't send it, some of the subsequent ACL commands will be "eaten up" by the ES command.

If you wish to include more than one escape sequence in one ES command, the second and subsequent sequences must start with the Escape character, which can be represented as "^[" in the string. Don't forget to include any necessary colons (":") following the parameters of the escape sequences.

While the System is processing an ES command it may not be able to receive characters from the RS-232C port. Also, any escape sequences received via the RS-232C port are "mixed in" with the ES command's string. Furthermore, if the ES command initiates any outputs to the host (for example, via ESC.B) these outputs may conflict with outputs requested simultaneously by the host. (See the ESC.E command in Chapter 6.) Therefore, it is advisable to use the ES command only in downloaded move sequences, rather than attempting to execute an ES "real time" as part of the ACL command stream from the host.

FH [<only if needed> [, <z first>]] **Find Home Switches**

If no parameters are specified the machine finds the Y Home switch, then the X Home switch. The XY motors are left at full power afterwards. (See the PM command.) The point where both switches close becomes the new (0,0) microstep coordinate, or Home position. The No Reference status bit is cleared. (See the OS command.) If either switch is not found within 32767 microsteps then that motor stops, the appropriate status bit is set, an error is logged, and that motor's present position is taken to be zero microsteps for that axis.

Before either the X or Y motor travels toward its Home switch the first time, they both move 250 microsteps in the opposite direction. This is in case the carriage is already beyond the switch closure point.

The entire XY motion sequence is actually performed twice -- the first time at high speed, the second at a lower speed for better accuracy. During the second sequence the motors back away only 100 steps before seeking the switches.

The <only if needed> parameter is optional. If it is specified and is nonzero then the System checks the No Reference status bit before performing any motion. (See the OS command.) The XY switches are only sought if they have not already been found (i.e., if the No Reference status bit is set).

For example, if you send "FH 1;" at the beginning of your move sequence then the System finds home the first time the sequence is executed but not subsequent times. This prevents unnecessary motion and delay.

If the <only if needed> parameter is omitted it defaults to zero; i.e., the Find Home action is performed regardless of the No Reference status bit.

<z first> is also optional but may only be specified if <only if needed> is also specified. If <z first> is present and is nonzero then the Z axis seeks its home switch before X and Y. For a description of this motion see the FZ command. If you have a three-axis system you may wish to use <z first> to prevent collisions with tooling fixtures, etc., during the XY motion.

If <z first> is omitted it defaults to zero, meaning that only the X and Y switches are found. You can change this default via the PE command, for example, to prevent collisions during the manual re-reference operation. See below.

If both <only if needed> and <z first> are nonzero then the System checks both the No Reference and No Z Reference status bits. If both XY Home and Z Home have already been found then no motion occurs. Otherwise, if either Home has not been found, the Z Home is found.

All of the distances and speeds associated with the FH motion can be changed via the PE command. The speed and acceleration of the back-off vectors can be changed via the SR and AC commands. If FH causes the motors to move away from the switches instead of towards them you can swap the motor direction via the PE command.

If the STOP button is pressed (or an external Emergency Stop switch closes) while either motor is moving, the motors stop immediately and the present position becomes the Home position. If the machine is already in the Emergency Stopped state when the FH command is received the command is ignored.

The action of the FH command can also be caused by the manual re-reference operation (i.e. pressing the FAST and GO TO ORIGIN buttons simultaneously). See Chapter 4.

The Automove System does not automatically find the Home switches at power up. Until it receives an FH command or manual re- reference, it assumes that the Home position is wherever the carriage was at power up. If you want your system to automatically find home you can make an Autostart sequence containing an FH command. See personality parameter 39 in Chapter 7.

Immediately after power up there are no constraints on front panel Arrow button motion. The first FH command or manual re-reference establishes Arrow button travel limits. See the TL command and the Arrow buttons in Chapter 4. As far as the AA, AR, MA and MR commands are concerned, travel limits are always in effect, regardless of whether the Home switches have been found.

At power up the Home position is wherever the carriage is when motor power is first applied, so until an FH or manual re-reference occurs the Home and Origin positions will both be at unknown locations with respect to the platen. The travel limits will not work correctly. Also, since ACL motion commands can go only to positive coordinates, and since the power-up position defaults to (0,0), a portion of the platen would only be reachable via the Arrow buttons. Moral: always find home FIRST!

The value of <mask> becomes a front panel lockout mask. The useful range is 0 through 8191. Each bit of the mask disables a particular button; if the bit is "1" the button is disabled; if "0", enabled. The mask defaults to 0 (all buttons enabled) if <mask> is omitted, and also at power up and IN.

Table 10 - Enabling/Disabling Front Panel Buttons

Bit	Value	Decimal Button
0	1	Right Arrow
1	2	Left Arrow
2	4	Down Arrow
3	8	Up Arrow
4	16	TEACH
5	32	FAST
6	64	GO TO ORIGIN
7	128	SET ORIGIN
8	256	PAUSE
9	512	STOP
10	1024	Interrupts (see below)
11	2048	Step Verify option
12	4096	Z AXIS

Use caution when disabling the STOP button. The external Emergency Stop switch input also becomes disabled, so the only way to stop the motors in an emergency situation is to press RESET or turn off the power switch. (Under some circumstances you may need to temporarily disable STOP in order to automatically resume a move sequence. See the CS command for an example.)

Bit 10 locks out the interrupts (i.e. triggering of Download Sequences) that can be caused by FAST/PAUSE, FAST/TEACH, the GO button, and transitions on the Digital Inputs, as described in Chapter 8. Bit 11 locks out motor slip detection by the optional Step Verify circuit.

If the GO TO ORIGIN button is disabled the Manual Re-reference sequence (invoked via FAST/GO TO ORIGIN) will also not work. Similarly, if the PAUSE or TEACH button is disabled, the corresponding Download Sequence (0 or 1) cannot be invoked via FAST/PAUSE or FAST/TEACH.

In the case of bits 8 through 11 (PAUSE, STOP, interrupts, and Step Verify): if the button is pushed (or the event occurs) while disabled by FP then the event is logged ("remembered") and acted upon as soon as a subsequent FP command re-enables it. While the event is pending the corresponding front-panel light, if any, glows steadily; i.e., does not blink. When another FP re-enables the event the light begins to blink.

Thus, FP can be used to protect small critical sections of a move sequence from undesired interference, as shown in the example below.

In the case of all other buttons (bits 0 through 7 and 12): if the button is pressed and released while disabled by FP then there is no effect. If the button is pressed while disabled but is held until after it is re-enabled then its action is performed at that time.

Example: the command FP 96; disables the FAST and GO TO ORIGIN buttons, so the operator can only move the carriage via the slow Arrow buttons.

Example: suppose your application uses a drill motor or a heater controlled by Digital Output 0. You execute the following sequence: FP 256; CD 1,1; WA .5; CD 0,1; FP;. The first FP locks out the PAUSE button. Then the first CD turns on the drill or heater; the WA waits for a half second; and the second CD turns it back off. The final FP re-enables all buttons. If the PAUSE button is pushed while the drill or heater is active then the PAUSE light turns on, but the machine does not actually pause until after the drill motor or heater is turned off.

FZ [<only if needed>] **Find Z Home Switch**

The Z axis motor moves in the negative direction until the Z axis home switch closes. This establishes the Z Home position. The No Z Reference status bit is cleared. (See the OS command.)

The <only if needed> parameter is optional. If it is specified and is nonzero then the System checks the No Z Reference status bit before performing any motion. (See the OS command.) If the switch has already been found (i.e. the status bit is zero) then the FZ command performs no motion.

If <only if needed> is omitted it defaults to zero; i.e., the motion is performed regardless of the No Z Reference bit.

The FZ command actually finds the Z home switch twice -- once at high speed and a second time at low speed for greater accuracy. The motor backs away 100 steps before seeking the switch the second time.

In order to provide greater load-carrying capacity in systems where the Z motor must fight gravity to reach the home switch, the FZ command uses a smooth acceleration at startup rather than instantly jumping to the full seek speed. However, when the switch closes the motor stops instantly.

The values of acceleration, seek speed, and back off distance can be changed via personality parameters 52 through 56; see Chapter 7. The half/full-step mode is as specified in the CZ command or personality parameter 46.

After the switch closes, the Z axis position counter is set to zero. If the switch has not closed within 32767 steps, the motor stops moving, an error is logged, and the present position becomes zero. The 32767-step limit can be changed via personality parameter 16.

The motor is left at full power after the switch is found.

If the STOP button is pressed (or an external Emergency Stop switch closes) while the motor is moving, the motor stops immediately and the present position becomes zero. If the machine is already in the Emergency Stopped state when the FZ command is received, the command is ignored.

The action of the FZ command can also be performed by the Z manual re-reference operation or in conjunction with the FH command or the XY manual re-reference operation. See the <z first> parameter of the FH command, personality parameter 48 in Chapter 7, and the section entitled Manual Re-reference in Chapter 4.

If FZ causes your Z axis motor to move the wrong direction (i.e. away from the home switch) you can reverse the motor direction via personality parameter 31. See Chapter 7.

The Automove System does not automatically find the Z home switch at power up. Until it receives an FZ command it assumes that the Z Home position is wherever the Z motor was at power up.

At power up there are no constraints on Z axis Arrow button motion. (See the AM command.) The position counter "wraps around" between 0 and the Z axis modulus, minus 1. (See the CZ command.) As soon as the first FZ is executed Z axis Arrow travel is limited to the range 0 through <mod> - 1, where <mod> is the Z axis modulus. (In other words, Arrow travel stops when the motor reaches the end of the range.) As far as the AZ command is concerned, travel limits are always in effect. The MZ command never has travel limits.

GD [<flag>] Lower Retractable Height Sensor (Gear down)

The GD command is used to turn off a solenoid that lowers the retractable height sensor. The I/O bit used was previously the Z axis test bit line for the Z head assembly. Insure that your system is equipped with the necessary changes to wiring before using this command. At power up or reset, the GD command will be cancelled.

See also: GU command

GU [<flag>] Raise Retractable Height Sensor (Gear up)

The GU command is used to turn on a solenoid that lowers the retractable height sensor.

See also: GD command

The IN command is equivalent to executing the following sequence of ACL commands:

EC; FP; CS; AB; AC; AM; AP; CD; CF; CP; CZ; MD; MM; MN; PD;

SO; SR; TL; VM; WD;

Table 11 - IN Command Default Settings

EC	Continuous Path is canceled. Any pending commands since BC are ignored.
FP	All front panel buttons are enabled.
CS	Emergency Stopped state and Motor Slipped state are cleared.
AB	Antibacklash vectors are disabled.
AC	Acceleration is 193, corresponding to 193000 steps/sec/sec.
AM	Arrow mode is set to XY.
AP	Primary I/O port is selected.
CD	All Digital Outputs are off.
CF	Calibration factors are defaulted as explained in CF.
CP	All patterns are cleared; the Origin is the "main" Origin. The pattern rotation angle is zero.
CZ	Z axis step rate is 100 steps/sec.
	Z axis modulus is 32767.
.	Z axis is in Half Step mode
	Z axis acceleration is 40000 steps/sec/sec.
MD	Mid-move Digital Outputs changes are disabled.
MM	Multiple Mid-move Digital Outputs changes are disabled.
MN	Mid-move Digital Inputs response is disabled.
PD	Pre-move Digital Outputs changes are disabled.
SO	"Main" Origin position is (0,0).
SR	Step rate is 10000 steps/sec.
TL	Travel limits are defaulted; see the TL command.
VM	Vector Mode is 0.
WD	Implicit Digital Outputs delay is 0.

In addition, IN causes the following actions: The ACL error code is cleared. The following OS status bits are cleared: Position Change, Origin Change, Taught Point Available, Emergency Stopped, Error, Motor Slipped. The Initialized status bit is set. The commanded position is recomputed from the Actual position according to the new (default) values of calibration factors, pattern rotation angle, and Origin position. The Taught Point is set to 0,0. The machine forgets any VC, VT, PAUSE, STOP, motor slip, or interrupt which has been logged but not yet acted upon. (See the FP command.)

X-Y or Z Motor power is not turned off and the present physical position is not changed. If the Home references were established they are not lost. Microstep resolution and linearity correction are not changed. Arrow motion remains unconstrained if it was already. Communications parameters are not changed. See ESC.! and ESC.R in Chapter 6, and Appendix A Power Up Actions.

Note that the IN command is not processed if the Automove System is in the Paused state waiting to finish a previous ACL command. If necessary, the host computer can first remotely clear the Paused state. See the ESC.! command in Chapter 6.

To customize the personality of the Automove System, some of the default values shown above can be changed via the PE command. For example, the SR default could be changed to 5000 so that it receives this value at power up, at IN, and at an SR with no parameters. The above table gives the factory-set values.

MA <x>, <y> Move Absolute

The carriage moves in a straight line (a "vector") to the absolute Calibrated Unit coordinate given by <x> and <y>. This is the most common motion command used with the Automove System.

The range of <x> and <y> is -32768.0000 through 32767.9999. The specified values are rotated by the pattern rotation angle. The rotated coordinates are multiplied by the calibration factors, rounded to the nearest integer, and added to the Origin position to get a pair of microstep coordinates. These coordinates are adjusted for linearity correction, and the carriage then moves to this absolute location. See the BP, CF, CR, and SO commands.

An antibacklash vector is performed if it is currently enabled; see the AB command. The current values of acceleration and step rate are used; see AC and SR. The XY motors are left at full power following the move; see the PM command.

During the conversion from Calibrated Units to microsteps the intermediate computation results must be within the range -32768 through 32767 and the final results (before Linearity Correction) must be within the travel limits. If either restriction is not met an error is logged, the out-of-range coordinate is set to the value of the nearest travel limit, and a vector to the resulting point is performed. (This is NOT a straight-line clipping algorithm.)

If the step rate has been set higher than about 7400 microsteps per second the System automatically skips some intermediate microstep positions during the vector. This skipping is done in a regular, smooth pattern and does not affect the overall motion or the accuracy with which the vector reaches its endpoint. (Contrast this with arcs; see AA.) Available motor torque may be reduced. The step rate is automatically constrained for certain values of resolution; see the RE and SR commands.

The MA command is ignored while the machine is in the Emergency Stopped state. If the STOP button is pressed (or an external Emergency Stop switch closes) during the vector, the motors stop immediately and the Home position reference becomes inaccurate. See the STOP button in Chapter 4 and the FH command in this chapter.

The effects of the MA command can be altered by the Vector Mode; see the VM command. To perform more than one vector or arc without stopping, see the BC command.

If short-vector system throughput is important, see Appendix E Speed Considerations.

MD [<count1>, <new1> [, <which1> [, <count2>, <new2> [, <which2>]]]]

Mid-move Digital Outputs Changes

The MD command establishes one or two points at which the eight Digital Outputs are to change state during each subsequent AA, AR, MA, MR, MT, or Continuous Path move until another MD command is executed. This is the equivalent of executing one or two CD commands in the middle of a move. See also the MM and PD commands.

Each Digital Output can only have one value at a time, either True or False (one or zero). The CD and TD commands change the outputs immediately; the MD and MM commands causes the outputs to change later, when vectors or arcs are performed. Once an output has been changed either by a CD or TD or by a move after a PD, MD, or MM, the old value of that output is forgotten.

If the value of <count1> or <count2> is positive, it specifies the number of microsteps from the beginning of the move, at which the Digital Outputs are to change state. If negative, it specifies the distance before the end of the move. The range of <count1> and <count2> is -32768 through 32767. Zero is interpreted as the beginning of the move, before the first step is taken.

If the magnitude of <count1> or <count2> is bigger than the length of a particular move, the Digital Outputs will change at the beginning or end of the move, as appropriate. If a vector is diagonal the true diagonal distance, in microsteps, is used.

<count1> and <count2> are applied to an AA or AR arc or Continuous Path move as though the move were "unrolled" into a straight line. In other words, a positive number is an arc-length distance from the beginning of the move, and a negative number is a distance from the end. (See the BC command.)

Note that it is possible to execute moves whose total length exceeds 65535 microsteps. In this case the middle portion of the move will be unreachable by the MD command; the Digital Outputs will not change in this region. If the length is in the range 32768 through 65535, a portion near the end will be unreachable by positive <count> values and vice-versa.

The parameters <new1>, <new2>, <which1>, and <which2> determine the new values written to the Digital Outputs. Each of these parameters is optional, and may only be specified if all preceding parameters have been specified. If a <count> is specified, the corresponding <new> must be specified.

<new1> is modified by <which1> and is output at <count1>. <new2> is modified by <which2> and is output at <count2>. For more information about these parameters, see the <new> and <which> parameters of the CD command.

If a <which> is omitted it defaults to 255 (i.e. binary 11111111), meaning that all 8 Digital Outputs are affected by the corresponding <new>.

If <count2>, <new2>, and <which2> are omitted, the Digital Outputs will change only once during each move. <count2> and <new2> may be specified only if <count1>, <new1>, and <which1> are all specified.

After power up and IN, and if an MD or MM with no parameters is executed, mid-move Digital Outputs changes are disabled.

The MD command does not use the delay specified by the WD command. The values of <count1> and <count2> must be coordinated with current acceleration and step rate such that any hardware attached to the Digital Outputs has enough time to actuate. If you need a precise delay between the first change and the beginning of motion, use the PD ("Pre-move Digital Outputs Changes") command.

If <count1> and <count2> resolve to the same microstep position within a given move, the <new2> value is output approximately 9 microseconds after the <new1> value.

The <new1> value is always output before the <new2> value. If <count2> resolves to an earlier microstep position than <count1> within a given move, the arithmetic mean (i.e. average) of the two positions is computed, and <new1> and <new2> are both output at this intermediate position, spaced approximately 9 microseconds apart.

A zero-length move (for example, due to MR 0,0; two MA's to the same location, or an arc with zero radius or sweep angle) produces both Digital Outputs changes (if specified), separated by approximately 9 microseconds.

Only one MD command at a time can be active; an MD cancels the parameters of any preceding MD. The MD and MM commands should not be used together.

If the machine is in the Emergency Stopped state when an AA, AR, MA, MR, MT, or Continuous Path move occurs, the motion command is ignored; hence no MD-induced change occurs at the Digital Outputs. If an Emergency Stop occurs during a move, the Digital Outputs all go immediately to the False state.

The System precomputes the locations at which the Digital Outputs are to change before starting each vector, arc, or Continuous Path move. If the move is terminated by a Dynamic Deceleration (i.e. has come to a complete stop) before a particular Outputs change has occurred then that change will not occur during this move. See the MN command.

The effects of the MD command can be suspended for a single move or group of moves through use of the VM command.

The current value being applied to the Digital Outputs can be determined via the OD command.

For more information about the Digital Outputs see Chapter 8 Using the Digital Outputs and Inputs.

Example: the command sequence MD 1000,3; MR 4000,4000; produces the following effect. The MR begins to execute. After 1000 diagonal steps (i.e. 707 steps each in the X and Y axes), Digital Outputs 0 and 1 become True and all others become False, regardless of their previous states. (Note that 3 is binary 00000011, which has bits 0 and 1 set).

Example: the sequence MD 500, 255, 255, -200, 0, 7; AR 1000, 0, 360; results in the following actions. The AR begins to execute; after 500 total steps along the arc, all 8 Digital Outputs become True; 200 steps before the end, outputs 0, 1, and 2 become False; the outputs remain unchanged for the remaining 200 steps.

Example: the following two sequences both have the same effect upon vectors and arcs whose length is less than 32768: MD 0, 255, 255, 32767, 0, 255; and MD -32768, 255, 255, 32767, 0; . The effect is to set all 8 Digital Outputs True at the beginning of each move, and False at the end. (Note that if no move's length exceeds 32767 then the changes are "constrained" so they occur at the beginning or end of each vector). If the length is greater than 32767 the two sequences begin to produce different results.

MM [<count>, <new> [, <which>]]

Multiple Mid-move Digital Outputs Changes

The MM command establishes up to 32 points at which the eight Digital Outputs are to change state during each subsequent AA, AR, MA, MR, MT, or Continuous Path move. This is the equivalent of executing multiple CD commands in the middle of a move. The effects of a series of MM commands remain active until a subsequent MM command or commands cancel them. See also the MD and PD commands.

Each Digital Output can only have one value at a time, either True or False (one or zero). The CD and TD commands change the outputs immediately; the MM command causes the outputs to change later, when vectors or arcs are performed. Once an output has been changed either by a CD or TD or by a move after a PD, MD or MM, the old value of that output is forgotten.

An MM command with no parameters clears out all mid-move Digital Outputs changes established by any previous MD or MM commands. Each subsequent MM command with parameters establishes one mid-move Digital Outputs change; up to 32 MM commands with differing parameters can be specified. All of the mid-move Digital Outputs changes will be applied during each subsequent vector, arc, or Continuous Path move.

Interaction Between MD and MM

The MD command has the same effect as executing one or two MM commands with parameters. Any preceding MM commands are canceled by an MD command. MD commands and MM commands should not be used together. The MM command is capable of performing the same functions as the MD command but the MD command has been preserved to maintain user program compatibility with previous versions of the Automove System.

The MM Parameters

Again, an MM command with no parameters erases the effects of all preceding MM and MD commands.

If <count> is specified and is positive, it specifies the number of microsteps from the beginning of the move, at which the Digital Outputs are to change state. If negative, it specifies the distance before the end of the move. The range of <count> is -32768 through 32767. Zero is interpreted as the beginning of the move, before the first step is taken.

If the magnitude of <count> is greater than the length of a particular move, the Digital Outputs will change at the beginning or end of the move, as appropriate. If a vector is diagonal then the true diagonal distance, in microsteps, is used.

<count> is applied to an AA or AR arc or Continuous Path move as though the move were "unrolled" into a straight line. In other words, a positive number is an arc-length distance from the beginning of the move, and a negative number is a distance from the end.

Note that it is possible to execute moves whose total length exceeds 65535 microsteps. In this case the middle portion of the move will be unreachable by the MM command; the Digital Outputs will not change in this region. If the length is in the range 32768 through 65535, a portion near the end will be unreachable by positive <count> values and vice-versa.

The parameters <new> and <which> determine the new values written to the Digital Outputs. Each of these parameters is optional, and may only be specified if all preceding parameters have been specified. In any given MM command if <count> is specified, <new> must also be specified.

<new> is modified by <which> and is output at <count>. For more information about the range and meaning of these parameters, see the <new> and <which> parameters of the CD command.

If <which> is omitted it defaults to 255 (i.e. binary 11111111), meaning that all 8 Digital Outputs are affected by the corresponding <new>.

After power up and IN, and if an MM or MD with no parameters is executed, mid-move Digital Outputs changes are disabled.

The MM command does not use the delay specified by the WD command. The values of each <count> must be coordinated with current acceleration and step rate such that any hardware attached to the Digital Outputs has enough time to actuate. If you need a precise delay between the first change and the beginning of motion, use the PD ("Pre-move Digital Outputs Changes") command.

Unusual Conditions

Each MM command's <count> value, whether positive or negative, resolves to a particular distance from the beginning of a given move. If more than one <count> resolves to the same microstep position within a given move, the next <new> value is output approximately 135 microseconds after the preceding <new> value.

If any <count>'s distance is less than that of the immediately preceding MM's <count>, the following action is taken: the average of the two distances is computed, and both Digital Outputs changes are performed at this intermediate distance, although separated by approximately 135 microseconds.

The MM's will be output in the same order that they were originally specified, even though their <count> values may be out of order. Each MM becomes a candidate for output as soon as the move has gone the specified distance, but will not be output until all previously-specified MM's are output. All specified MM's will be output during each move. (For an exception with MN, see below).

A zero-length move (for example, due to MR 0,0; two MA's to the same location, or an arc with zero radius or sweep angle) produces all Digital Outputs changes specified, each one separated by approximately 20 microseconds from the preceding one.

A move which has a non zero length that is less than any or all of the MM <counts> will cause all unprocessed MM's to be output at the end of the move, separated by approximately 20 microseconds.

If the machine is in the Emergency Stopped state when an AA, AR, MA, MR or MT occurs, the motion command is ignored; hence no MM- induced change occurs at the Digital Outputs. If an Emergency Stop occurs during a move, the Digital Outputs all go immediately to the False state.

The System precomputes the locations at which the Digital Outputs are to change before starting each move. If the move is terminated by a Dynamic Deceleration (i.e. has come to a complete stop) before a particular Outputs change has occurred then that change will not occur during this move. See the MN command.

Miscellaneous Information And Examples

The effects of the MM command can be suspended for a single move or group of moves through use of the VM command.

The current value being applied to the Digital Outputs can be determined via the OD command.

Example: the command sequence MM; MM 1000,3; MR 4000,4000; produces the following effect. The MR begins to execute. After 1000 diagonal steps (i.e. 707 steps each in the X and Y axes), Digital Outputs 0 and 1 become True and all others become False, regardless of their previous states. (Note that 3 is binary 00000011, which has bits 0 and 1 set.)

Example: the sequence MM; MM 500, 255, 255; MM -200, 0, 7; AR 1000, 0, 360; results in the following actions. The AR begins to execute; after 500 total steps along the arc, all 8 Digital Outputs become True; 200 steps before the end, outputs 0, 1, and 2 become False; the outputs remain unchanged for the remaining 200 steps.

Example: the command sequence MM; MM 1000,3; MM 2000,2; MM 3000,1; MR 4000,4000; produces the following effect. The MR begins to execute. After 1000 diagonal steps (i.e. 707 steps each in the X and Y axes), Digital Outputs 0 and 1 become True and all others become False, regardless of their previous states.

After an additional 1000 diagonal steps, Digital Output 0 becomes false and after another additional 1000 steps Digital Output 0 becomes true again.

Example: the following two sequences both have the same effect upon vectors and arcs whose length is less than 32768: MM; MM 0, 255, 255; MM 32767, 0, 255; and MM; MM -32768, 255, 255; MM 32767, 0; . The effect is to set all 8 Digital Outputs True at the beginning of each move, and False at the end. (Note that if no move's length exceeds 32767 then the changes are "constrained" so they occur at the beginning or end of each vector.) If the length is greater than 32767 the two sequences begin to produce different results.

For more information about the Digital Inputs please see Chapter 8 Using the Digital Outputs and Inputs.

MN [<mode>, <value> [, <which>]] **Mid-move Digital Inputs Response**

The MN command sets up automatic actions which are to take place during subsequent AA, AR, AZ, MA, MR, MT, MZ, and Continuous Path moves until another MN command is executed. The actions will only take place if a specified condition occurs on any or all of the eight Digital Inputs before or during the move.

Each bit of the <mode> parameter controls one action, as follows:

Table 12 - <mode> Parameter Controls For Command MN

Bit	Decimal Value	Action
0	1	If this bit is set the System enables the Dynamic Teach mode: If, during a move, the Digital Inputs come to match <value> and <which> then the present XY position is immediately recorded as a Taught Point and the Taught Point Available status bit is set. (See the OS command.) Any old Taught Point is overwritten. After the move is finished the host computer can read the Taught Point via the OT command. You can move the carriage to this point via the MT command. If the Digital Inputs condition occurs during a Z-axis move the System records the XY position, not the Z position; thus, the major use of Dynamic Teach during Z moves is to simply record (in the OS command's Taught Point Available bit) the fact that the Dynamic Teach did occur. See also the TEACH button in Chapter 4.
1	2	If this bit is set the System enables the Dynamic Deceleration mode: If, during a move, the Digital Inputs match <value> and <which> then the move immediately enters the deceleration phase; i.e., it rapidly comes to a halt. The host computer can determine the actual stopping position via the OA or OZ command. If you want an "instantaneous" stop then use a low speed and a high acceleration; see AC, CZ, and SR. Be careful, not to set the acceleration so high that the motors slip due to the inertia of the load. If there are mid-move Digital Outputs changes set up by MD or MM and they have not yet occurred by the time the motors come to a stop then they will not occur during this particular move.
2	4	Use <value> and [<which>] bits as OR inputs for Dynamic Teach mode.
3	8	Use <value> and [<which>] bits as OR inputs for Dynamic Deceleration mode.

The other bits of <mode> are ignored. For compatibility with future enhancements they should be set to zero.

The <mode> defaults to zero at power up and at IN, or if <mode> is omitted from an MN command. This default means that all automatic actions are disabled. If an MN command specifies a zero <mode> then <value> and <which> are ignored. <mode> and <value> must be either both specified or both omitted.

The <value> and <which> parameters set up a Digital Inputs condition which is to trigger the automatic actions. The parameters are treated as 8-bit numbers and are similar to the <new> and <which> parameters of the CD command, respectively. That is, the System checks that each Digital Input matches the corresponding bit of <value>, where a True input matches a 1 bit and a False input matches a 0. However, the System only checks those Digital Inputs which correspond to a 1 bit in <which>. All other Digital Inputs are ignored and do not need to match.

The <which> parameter is optional but may only be specified if <mode> and <value> are also specified. If <which> is omitted it defaults to 255 (i.e. binary 11111111), meaning that all eight Digital Inputs must match <value>.

If the specified Digital Inputs condition already exists when a move command is executed then the System performs the automatic action before any motion occurs. If the Dynamic Deceleration feature is enabled then the command produces no motion.

Only the first occurrence of the <value>/<which> condition during any given move produces an automatic action. Subsequent occurrences during the remainder of that move are ignored, unless they persist until the next move, at which time they will be seen.

During a vector the Digital Inputs are tested approximately once every 135 microseconds (0.000135 second); during an arc or Continuous Path move, approximately once every 450 microseconds (0.000450 second); and during a Z move, approximately once every 100 microseconds (0.000100 second). You must ensure that the desired Digital Inputs condition lasts for at least, say, one millisecond (0.001 second) so that the Automove System will not miss it.

A zero-length move (for example, MR 0,0;) checks the Digital Inputs once. Therefore you can use MN to "teach" the present position, via the following sequence: MN 1,0,0; MR 0,0; MN; . The first MN has a zero <which> so that the MR "teaches" immediately regardless of the state of the Digital Inputs. This could be useful, for example, to "memorize" and later return (via MT) to a position that resulted from a complicated series of motions or an operator input via the Arrow buttons.

The automatic actions of MN do not take place during antibacklash vectors (see the AB command), during the Find Home motion, or during any front-panel motion such as GO TO ORIGIN or Arrow buttons. The Digital Inputs are ignored during these moves.

Note: after an XY Dynamic Deceleration the commanded position will be at the expected endpoint but the actual position will be wherever the carriage actually came to rest in the middle of the move. If you then execute an AA, AR, or MR the System will "think" it is at the commanded position and may behave in an unexpected way. (For example, an MR 0,0; will go to where the original move would have gone, had there been no Dynamic Deceleration.) It may be useful to perform an MT ("Move To Taught Point") before doing an AA, AR, or MR.

Note that no signal debouncing of the Digital Inputs is performed in conjunction with MN. (For more information about debouncing please see Chapter 8). Normally this will not cause any problems because only one response can occur during a given move. By the time the move finishes and another move starts any switch bounce will probably be over. However, if your switches are switching near the end of an already-decelerating move (or if high acceleration produces an extremely short deceleration phase) then you may need to insert a short delay between moves; see the WA command.

The signal sense (High True vs. Low True) of the Digital Inputs for the MN command is set by personality parameter 38.

The effects of MN can be suppressed for one move or a group of moves via the VM ("Vector Mode") command.

Caution: To prevent unexpected responses you should be sure that the MN features are only enabled during the particular move or group of moves where you desire a response. After you get the response you should either execute a disabling MN; or VM 8; or somehow reset or disable the external hardware so that the next move does not immediately generate the same response.

Also, be sure to disable antibacklash vectors (via the AB or VM command) while using MN. Otherwise, the MN action is performed during the first ("long") vector, but then the carriage moves to the original MA or MR commanded endpoint.

Example: suppose you are using a laser to trim thick-film resistors on a hybrid integrated circuit. A sensing device measures the value of a resistor while it is being trimmed; when a particular value is reached external circuitry shuts off the laser and sets Digital Input 0 true. After you make the first "rough" cut you wish to make another "fine tune" cut at right angles, beginning where the first cut left off. You might use the following sequence of ACL commands:

```
MN 3,1,1; CD 3,3; MR 0,1000; CD 0,3; MT;

CD 0,2; CD 3,3; MR 2000,0; MN;
```

The first MN sets up Dynamic Teach and Dynamic Deceleration. The first CD 3,3; turns on the laser and enables the sensor. (You might use MD, MM, or PD here instead of CD.) The MR 0,1000; makes the first cut in the Y direction; when the sensor triggers it shuts off the laser; the position is taught; and the vector decelerates. The CD 0,3; disables the sensor and the laser during the MT vector. The MT moves back to the Taught Point (i.e. the end of the first cut). The CD 0,2; CD 3,3; re-enables the sensor and turns the laser back on. The MR 2000,0; makes the fine-tuning cut in the X direction. Finally, the MN; disables Dynamic Teach and Dynamic Deceleration.

Example: suppose you want a Download Sequence to execute immediately when a certain Digital Input goes True, even in mid- vector. Just use personality parameters 36 through 38 to set up the Download Sequence as an interrupt. Then, at the beginning of the move sequence, execute an MN to abort any move when that Digital Input goes true. Remember, however, that there is a one- command latency on interrupts; this means that one more ACL command will execute after the aborted move, before the Download Sequence begins to execute.

Previous firmware versions used any input bits specified for the MN command as AND'ed for the MN to occur. Mode bit 0 is set to enable a Dynamic Teach, and mode bit 1 is set to enable Dynamic Deceleration. Two new bits (2 and 3) may now be used to perform the selected function if any of the value/which bits occur. If mode bit 2 is specified as set, the Dynamic Teach mode will occur if any of the which bits match their corresponding value bits. If mode bit 3 is specified as set, the Dynamic Deceleration mode will occur if any of the which bits match their corresponding value bits. To maintain the existing servo rate an alternate servo routine may be selected automatically when an OR'ed MN is in effect.

Examples:

MN4,3,3; allows bit 0 OR bit 1 to cause a Dynamic Teach, if either becomes set during a move.

MN8,3,3; allows bit 0 OR bit 1 to cause Dynamic Deceleration if either becomes set during a move.

If mode bits for ANDing and ORing are both set at the same time, the AND bits will have precedence to maintain compatibility with older software.

For more information about the Digital Inputs please see Chapter 8 Using the Digital Outputs and Inputs.

MR <dx>, <dy> — Move Relative

The MR command produces a straight vector motion like that of the MA command, except that the MR parameters <dx> and <dy> are interpreted as an offset relative to the current Commanded position. (In contrast, the MA parameters <x> and <y> are absolute; i.e., they give an offset from the Origin position.) The range of <dx> and <dy> is -32768.0000 through 32767.9999.

The System adds <dx> and <dy> to the Commanded position and saves the results as the new Commanded position. Next, it rotates the position by the pattern rotation angle (see BP). It then multiplies by the calibration factors, rounds to the nearest integer, and adds the Origin location to produce a pair of microstep coordinates. At this point the XY Linearity Correction Table is applied. Then a vector to this point is performed as by the MA command.

The intermediate computation results must be within the range -32768 through 32767 and the final results (before Linearity Correction) must be within the travel limits. If either restriction is not met an error is logged, the out-of-range coordinate is set to the value of the nearest travel limit, and a vector to the resulting point is performed. (This is NOT a straight-line clipping algorithm.)

The MR command is ignored while the machine is in the Emergency Stopped state. If the STOP button is pressed (or an external Emergency Stop switch closes) during the vector, the motors stop immediately and the Home position reference becomes inaccurate. See the STOP button in Chapter 4 and the FH command in this chapter.

The effects of the MR command can be altered by the Vector Mode; see the VM command.

If short-vector system throughput is important, see Appendix E Speed Considerations.

MT — Move To Taught Point

This command moves the carriage in a straight line back to the most recent XY Taught Point. All of the effects of this vector are as though an MA command had been executed. The Taught Point Available status bit is cleared. The Commanded position is set to the Calibrated Unit equivalent of the Taught Point, even if it is outside the travel limits, although the motion itself is constrained to the travel limits.

The Taught Point is established by pressing the TEACH button (see Chapter 4) or by executing a Dynamic Teach (see the MN command).

Note that it is possible to automatically "teach" the current carriage location by executing MN 1,0,0; MR 0,0; MN; . You can later return to this point via MT. For more information and an example see the MN command.

Example: suppose in your application the operator designates one "special" location on the platen where several operations are to be performed with different tools. The operator moves the carriage to this special place and pushes TEACH. Then he runs the move sequence. The machine fetches several tools and, with each one, moves to the taught position (via MT) and performs some operation.

MZ <numsteps> — Move Z (relative)

The Z axis motor takes <numsteps> steps from its present position, at the acceleration, step rate and half/full-step mode established by the CZ command. The <numsteps> parameter specifies a relative move, not an absolute move. <numsteps> can be in the range -32768 through 32767 and is not restricted by the Z axis modulus. (See the CZ command.)

The motor is left at full power after the move.

The MZ command has a "wraparound" feature. If at any time the Z axis position counter is decremented below zero, it is set to the Z axis modulus, minus one. If it is incremented above the modulus minus one, it is set to zero. Thus a series of MZ commands can move the Z axis an arbitrary distance in either direction. The only restriction is that a single move cannot exceed -32768 or 32767 steps. Note that this wraparound feature does not apply to the AZ command.

If the STOP button is pressed (or an external Emergency Stop switch closes) while the Z motor is moving, the motor stops immediately. Its physical position reference remains accurate unless the sudden deceleration causes the motor to slip. However, the No Z Reference status bit is set, regardless. (See the OS command.)

If the machine is already in the Emergency Stopped state when the MZ command is received, the command is ignored. See the STOP button in Chapter 4 and the FZ command in this chapter.

To perform absolute Z axis moves with travel limits, use the AZ ("Absolute Z Axis Move") command.

The Z axis has no calibration factor, origin offsetting or antibacklash vectors. A simple origin offsetting can be done via the SZ command.

OA Output Actual Position

The Automove System outputs the Actual carriage position, in microsteps, to the host computer, and then clears the Position Change status bit. The X coordinate is output first, followed by a comma, then the Y coordinate. The coordinates are integers in the range 0 through 32767.

The output values are unaffected by the calibration factors, pattern rotation angle, and the location of the Origin; they merely reflect how many net microsteps the carriage has traveled away from the Home position.

The Actual position may differ from the Commanded position if, for example, an out-of-range value was received; or if the pattern rotation angle is nonzero; or if the calibration factors are not 1.0000; or if the Origin is not at the Home position; or if the commanded position is not an integer.

OB Output Buttons

The OB command allows the current state of the front panel buttons mask to be output. The front panel buttons mask is set with the FP command. This command is useful to interrupting interlock routines which can capture the current state of the front panel into a variable and then reinstate the previous front panel conditions before the interlock interrupt exits.

OC Output Commanded Position

The Automove System outputs the Commanded position, in Calibrated Units, to the host computer. The X coordinate is output first, followed by a comma, then the Y coordinate. The values will be in the range -32768.0000 through 32767.9999.

The Position Change status bit is cleared.

Normally the Commanded position is the direct result of the AA, AR, MA and MR command parameters. However, moving the carriage by pressing the Arrow buttons or the GO TO ORIGIN button causes the Commanded position to be updated to the new carriage location, taking into account the pattern rotation angle, calibration factors, Origin location, and linearity correction. This may introduce numeric errors into the Commanded position coordinates; see the CF command.

OD Output Digital Outputs State

The Automove System outputs to the host the most recent value applied to the Digital Outputs, in the form of a decimal integer in the range 0 through 255. The meaning of this 8-bit number is the same as the <new> parameter in the CD command; i.e., each bit in the number corresponds to the state of one Digital Output.

For example, if outputs 2 and 0 are True and all others are False, the OD command outputs a 5. (5 corresponds to binary 00000101, which has only bits 2 and 0 set).

Contrast this command with the ON command, which reports the values on the Digital Inputs.

OE Output Error Code

The Automove System outputs its current ACL error code and then clears the error code and the Error status bit. The meaning of the code is as follows:

Table 13 - Error Code Descriptions

0	No ACL error has occurred since the last OE.
1	An unrecognized mnemonic has been received.
2	The wrong number of parameters has been received.
3	An out-of-range parameter has been received.
4	The X, Y, or Z Home switch was not found within 32767 motor steps during an FH or FZ command.
5	The download memory has overflowed or the Write Protect switch is not enabled during a CR or PE command or BD...ED sequence.
6	A position overflow has occurred; an AA, AR, AZ, MA, MR, SP, or SZ has attempted to move the carriage or Z motor outside the travel limits.
7	Download Sequences have called each other to a depth greater than 12. (This maximum depth includes interrupts.)
8	Patterns have been nested to a depth greater than 12, or an EP has been executed with no pattern active.
9	A disallowed command has been executed during a Continuous Path sequence, or too many physical action commands have occurred between BC and EC.
10	A Continuous Path sequence has exceeded the maximum allowable total length, or the carriage has been moved via the front panel controls between BC and EC, or a BC with a nonzero <re-use> parameter was disallowed.

Once an error has been logged it remains logged until cleared by an OE or IN command.

The error code reflects only the first ACL error that occurs; subsequent errors are logged only if an intervening OE or IN has cleared the error code.

Whenever any ACL error is logged, a "?" character is sent immediately to the host. This may occur even before the Automove System has received all of the characters associated with the bad command; for example, if the first of two parameters is out of range. See Exceptional Conditions in Chapter 5.

OF Output Calibration Factors

The Automove System outputs the current values of the calibration factors. The X value is output first, then a comma, then the Y value. The values will be in the range 0.0000 through 32767.9999.

OG Output Angle

The Automove System outputs to the host the current pattern rotation angle, in degrees. The value will be in the range 0 through 359.9999.

The rotation angle is the sum of the <angle> parameters of all currently active patterns. See the BP, EP, and CP commands.

OI Output Identification

The Automove System outputs an identification string of the form:

AUTOMOVE REV 3.22/3.15

This string includes the revision numbers of the ROMs for the two internal Automove System microprocessors.

OL Output Travel Limits

The Automove System outputs the current values of the travel limits, in microsteps. (See the TL command.) Four integer values are output, separated by commas. The values are in the range 0 through 32767, and are in the same order as the TL command's parameters: <xmin>, <ymin>, <xmax>, <ymax>.

This command can be used, for example, to see what default values the System assigns to the travel limits at power up.

ON Output Digital Inputs State

The Automove System outputs the current values of the 8 Digital Inputs, in the form of a decimal integer in the range 0 to 255. Each bit in the number represents one Digital Input: if the Input is True the bit is 1; if False, 0.

(The ON command can also be used to test the value of an ACL variable. For more information, please see the VT, V<, V=, and V> commands, as well as Chapter 10. The discussion, below, assumes that the Digital Inputs are being sampled.)

For example, if Digital Inputs 2 and 1 are True and all others are False, the ON command outputs a 6. (Decimal 6 is equivalent to binary 00000110, which has only bits 2 and 1 set).

Note that the sense of the Digital Inputs can be set to be either High True or Low True; see personality parameter 38 in Chapter 7. As shipped from the factory the Digital Inputs are all Low True (i.e. Negative True, such that a low voltage means True and a high voltage means False). This is in contrast to the Digital Outputs, which are always High True.

Contrast the ON command with the OD command, which reports the current state of the Digital Outputs.

For more information about the Digital Inputs see Chapter 8.

OO Output Origin

The Automove System outputs to the host the location of the Origin position relative to the Home position, in microsteps. The X coordinate is output first, followed by a comma, then the Y coordinate; the values are integers in the range 0 through 32767.

The Origin Change status bit is cleared. See the SO command and the SET ORIGIN button.

OP <selector> Output Personality Parameter

The Automove System outputs to the host the current value of one personality parameter, as stored in the System's non-volatile memory.

<selector> is an integer which determines which personality parameter is to be output. The allowable <selector> range is given in Chapter 7. The range and form of the value sent to the host depend on which parameter is being output; the value may be a fractional or whole number, signed or unsigned.

If <selector> is out of range an error is logged and the System outputs the value zero.

For a list of the personality parameters and an explanation of how to use them, see Chapter 7 Changing the Personality.

OQ Output Qualities

The OQ command allows the current state of the VM (vector mode) condition to be output. The vector mode conditions are set with the VM command. This command is useful to interrupting interlock routines which can capture the current state of vector mode into a variable and then reinstate the previous vector mode conditions before the interlock interrupt exists.

OR <x index>, <y index> **Output Correction**

This command causes the Automove System to output a pair of correction offsets as previously established via the CR command. The X value is output, then a comma, then the Y value. The values are in microsteps, in the range -128.0000 through 127.0000, except as described below.

<x index> and <y index> determine which location within the XY Linearity Correction Table is to be output; they correspond to the <x index> and <y index> parameters of the CR command, and must be in the range 0 through 7, except for the special case described below.

If <x index> and <y index> are both -1 (minus 1) then the OR command outputs the X and Y grid spacing of the linearity correction grid, in microsteps. The numbers are in the range 0.0313 through 32767.0000.

For more information about the OR command please see Chapter 9 Linearity Correction.

OS Output Status

The Automove System outputs the value of its status code to the host, then clears the Initialized bit. The code is output as a decimal integer in the range 0 through 1023, whose bits have the following meanings:

Table 14 - Status Code Definitions

Bit	Dec. Val.	Name	Meaning, if "1"
0	1	Position Change	The position has been changed manually since the last OA, OC, or OZ.
1	2	Origin Change	The Origin was changed manually since the last OO or SO.
2	4	Taught Point Available	A taught point has become available since the last OT.(See OT, MN, and TEACH.)
3	8	Initialized	An IN has occurred or the machine has been powered up since the last OS; this bit is cleared after OS reads it.
4	16	Emergency Stopped	The machine is in the Emergency Stopped state. See theCS command in this chapter and the STOP Button in Chapter 4.
5	32	Error	An ACL error was detected since the last OE.
6	64	No Reference	The XY Home switches have not yet been found via an FH or a manual re-reference since the most recent power up, Emergency Stop, reset, or RE.
7	128	No Z Reference	The Z Home switch has not yet been found via an FZ, FH, or manual re-reference since the most recent power up, reset, or Emergency Stop.
8	256	Z Arrow Mode	The front panel Arrow buttons are in Z Arrow mode. See the AM command in this chapter and "Z Axis Arrows" in Chapter 4.
9	512	Motor Slipped	The optional Step Verify circuit has detected a stepping motor slip in X or Y. See CS.

OT Output Taught Point

If the Taught Point Available status bit is not set, the Automove System first waits until it is. (This bit is set when the front panel TEACH button is pressed or when a Dynamic Teach occurs as the result of an MN command.) Then the System outputs the location of last Taught Poin to the host and clears the Taught Point Available status bit. The X coordinate is output first, then a comma, then the Y coordinate; the values will be in the range -32768.0000 through 32767.9999.

The Taught Point is the carriage position which was current when the most recent TEACH or Dynamic Teach occurred. It is saved as a microstep location. The OT command converts that location to Calibrated Units according to the pattern rotation angle, calibration factors, Origin, and linearity correction in effect at the time of the OT. Therefore the values could be affected by an intervening BP, CR, EP, CF, or SO command or by a push of the SET ORIGIN button.

If the Taught Point Available status bit is already set, pressing the TEACH button does not store a new point; the host must first "use" the previous point via OT or MT. In contrast, a Dynamic Teach overwrites any old Taught Point regardless of the Taught Point Available status bit.

An unwanted previous taught point can be cleared by executing a dummy OT and ignoring the results.

The host can use the OS command to test whether the Taught Point Available status bit is set without risking hanging up if no Taught Point is available. Alternatively, the host can just send OT and wait for the results to come back. The OT command can be aborted (without pressing TEACH) by sending the. ESC K command; see Chapter 6.

You can move the carriage to the Taught Point via the MT command.

The Taught Point defaults to (0,0) at power up and IN.

If you need to teach the Z axis position use the programming techniques described under the AM ("Arrow Mode") command.

OU <quotechar> [<char> [, <char> ...]] <quotechar> [, <suppress term>]

Output Literal String

This command causes the Automove System to output a literal string of ASCII (American Standard Code For Information Interchange) characters to the host. The string may contain up to 64 characters.

The <quotechar> is any ASCII character other than NULL or Escape. (As with all ACL, each character used in an OU command must not have bit 7 set; i.e., the decimal equivalent must be less than 128.) You may choose any character to delimit the string, as long as that character does not appear within the string itself. The suggested character to use under normal conditions is the ASCII "double quote" itself ("). If a double quote must appear in the string, some other character (such as single quote or apostrophe) should be used to delimit the string. Blanks preceding the first <quotechar> are ignored.

When the OU command is executed, the System sends each of the <char> characters to the host. A maximum of 64 characters may appear between the two <quotechar> delimiters. If too many characters appear, an error is logged and the excess characters are treated as normal ACL commands.

Don't forget the closing <quotechar> -- if you don't send it, some of the subsequent ACL commands will be "eaten up" by the OU command.

If the <suppress term> parameter is present and is nonzero then the Output Terminator sequence is suppressed (for this particular OU command only). The Output Terminator defaults to an ASCII Carriage Return and Linefeed; these are normally sent after each ACL output. <suppress term> also suppresses any Echo Terminator wait. See the ESC.M command in Chapter 6.

Any ASCII control character can be included in the string via the following technique: the ASCII carat ("^"), followed by some other ASCII character, generates the "control equivalent" of the second character. For example, ^M generates a Carriage Return and ^L generates a Formfeed. The ASCII Escape character can be generated via ^[. To include a carat in the string, simply put in two carats, ^^.

Any actual control characters (Carriage Return, Linefeed, etc.) appearing literally between the delimiters are faithfully recorded and sent back to the host. Exceptions: the NULL character (decimal value 0) is completely ignored by the Automove System; and the Escape character (decimal value 27) is treated as part of an Automove escape sequence, legal or otherwise.

Warning: Easymove interprets the question mark character (?) in the OU command as an error report from the Automove System and attempts to display an error message.

OV <varid> [, <suppress term>] — Output Variable

The Automove System outputs to the host computer the contents of the variable whose i.d. is given by <varid>. The value is in the range -32768.0000 through 32767.9999.

If the <suppress term> parameter is present and is nonzero then the Output Terminator sequence is suppressed (for this particular OV command only). The Output Terminator defaults to an ASCII Carriage Return and Linefeed; these are normally sent after each ACL output. <suppress term> also suppresses any Echo Terminator wait. See the ESC.M command in Chapter 6.

For more information about ACL variables please see Chapter 10.

OX — Output AP value

This command will output the currently selected AP mode. This allows the variable “shadow register” in ACL programs to be eliminated.

OZ — Output Z Position

The Automove System outputs to the host the current value of the Z axis position counter; this will be an integer in the range 0 through M - 1, where M is the current Z axis modulus. With the power up default modulus, the range of values is 0 through 32766. See the CZ, AZ, and MZ commands.

The Position Change status bit is cleared.

PD [<delay>, <new> [, <which>]] Pre-Move Digital Outputs Changes

The PD command establishes Digital Outputs changes and delays which are to occur before each subsequent AA, AR, MA, MR, MT, or Continuous Path move until another PD command is executed. This is the equivalent of executing a CD command and a WA command before each move. However, with PD the delay between the Outputs change and the beginning of motion is precisely controllable and cannot be interrupted, for example, by pressing the PAUSE button at the wrong moment.

Each Digital Output can only have one value at a time, either True or False (one or zero). The CD and TD commands change the outputs immediately; the PD command causes the outputs to change later, when X/Y motion is performed. Once an output has been changed either by a CD or TD or by a move after a PD, MD, or MM, the old value of that output is forgotten.

PD can be used with the MD or MM commands to establish up to 33 Digital Outputs changes to be performed before and during each move.

After power up and IN, and if a PD with no parameters is executed, pre-move Digital Outputs changes are disabled. They can also be temporarily disabled via the VM command.

<delay> is optional; it is the number of seconds to wait after changing the Outputs, before beginning the motion. It can be in the range 0 through 6.5535, with a resolution of 0.0001 second. If <delay> is zero then the PD command is equivalent to an MD or MM command with a <count1> of zero. In other words, the Outputs change just before the first step is taken. If both a PD 0,... and an MD 0,... or MM 0,... are active then the PD's value is output approximately 40 microseconds before the MD's or MM's first value.

If <delay> is specified then <new> must also be specified.

<new> and <which> determine the new values to be written to the Digital Outputs. For information about the range and meaning of these parameters please see the <new> and <which> parameters of the CD command.

If <which> is omitted it defaults to 255 (i.e. binary 11111111), meaning that all 8 Digital Outputs are affected by the value of <new>. <which> cannot be specified unless <delay> and <new> are also specified.

A zero-length move (for example, due to MR 0, 0; two MA's to the same point, or an arc with zero radius or sweep angle) produces the specified Outputs change and delay, but no motion.

Only one PD command at a time can be active; a PD cancels the parameters of any preceding PD.

If the machine is in the Emergency Stopped state when an AA, AR, MA, or MR occurs the motion command is ignored. Therefore no PD-induced change occurs at the Digital Outputs. If an Emergency Stop occurs during a move the Digital Outputs all go immediately to the False state.

The effects of the PD command can be suspended for a single move or group of moves through use of the VM command.

The most recent value sent to the Digital Outputs can be determined via the OD command.

Example: the command sequence PD .05, 4, 4; MR 1000, 0; produces the following results. Digital Output 2 is set True; 50 milliseconds later (i.e. .05 seconds) the carriage begins to move. (Note that 4 is equivalent to binary 00000100, which has only bit 2 set.)

Example: PD 0.008, 16, 16; MD -20, 0, 16; MR 0, 1000; MR 100, 100; produces the following physical effects. Digital Output 4 becomes True, then after .008 seconds the carriage begins to move. After 980 steps (1000 minus 20) in the Y axis, Digital Output 4 becomes False again. After the end of the first move, Output 4 goes True again, and .008 seconds later the second move begins. 20 counts before the end of the second move, Digital Output 4 is set False again. (Note that 16 is equivalent to binary 00010000, which has only bit 4 set.)

For more information about the Digital Outputs see Chapter 8 Using the Digital Outputs and Inputs.

PE <selector>, <value> — Personality

This command changes a parameter of the Automove System's "personality". The new value is written to the System's non- volatile memory so that it becomes "permanent". This means that the parameter retains its value even if the power is turned off.

The personality parameters are, for example, the speed and direction of Arrow button motion, the speed and maximum travel of the FH command, the size of antibacklash vectors, and so forth. For a complete list of the parameters and further explanation, see Chapter 7 Changing the Personality.

Before sending the PE command be sure that the backpanel Memory Enable switch is in the Down position. If it isn't then the System logs an error and ignores the PE command.

<selector> is an integer which determines which personality parameter is to be changed. The allowable <selector> range is given in Chapter 7 Changing the Personality.

<value> is the new value to be assigned; its allowable range depends on which personality parameter is being changed. You must be careful to send the correct value because the PE command does not check it. An incorrect value may cause an ACL error at IN or at power up.

The personality parameters are not affected by the IN command, but the factory-set values can, with some exceptions, be reinstated via the ESC.!6: command; see Chapter 6.

The active value of any personality parameter can be read via the OP command.

Example: PE 2, 200; sets the Arrow Slow Step Delay to the value 200, which is twice as long as the factory-set value 100. The new value is written to non-volatile memory and thus becomes the "permanent" value.

PM [<level>] **Power Level of Motors**

The power level applied to the X and Y motors is altered immediately, according to the value of <level>:

Table 15 - Power Levels of X and Y Motors

-1	Switching drivers are shut off
0	Drivers on, no current
1	Drivers on, half current (1/4 static power)
2	Drivers on, full current

This new setting lasts until the next XY motor movement occurs, at which time the motors are set to full current for the motion and left at full current after the motion.

Caution: Be sure to shut off the switching drivers before electrically disconnecting a motor while the Automove power is on.

At power up the switching drivers are enabled but no current is applied to the X and Y motors until the first front panel or ACL motion command. The IN command does not affect the power level.

If omitted, <level> defaults to 0, meaning drivers on, no current.

If <level> is specified as 0 or -1 (i.e. there will be no current to hold the motors in place) then the optional Step Verify circuit is deactivated. This allows the motors to be moved by external means without triggering the Motor Slipped state. Step Verify is reactivated when motor current is turned back on by another PM command or by any motion command.

If necessary you can keep Step Verify active while motor current is off in order to keep track of whether the motors did move. See personality parameter 50 in Chapter 7. But use caution -- if a motor is moved a small amount (say, 3 full steps) and then the current is turned back on then the motor may go to a different stable position than the original one, thus causing a slip to be detected.

PS — Pause

The machine enters the Paused state as though the front-panel PAUSE button had been pressed. Subsequent ACL commands are placed in the input buffer but are not executed until the Paused state is cleared.

The Paused state is typically cleared by pressing the PAUSE button a second time. It can also be remotely cleared by the host computer. See the ESC.!4: command in Chapter 6. See also the PAUSE button in Chapter 4. (The PS command itself can't clear a pause because the PS won't execute while the machine is paused!)

Note: if you press the SET ORIGIN button while an ACL command is paused the SET ORIGIN doesn't actually take effect until after the machine executes the paused ACL command. For example, suppose your move sequence contains the following commands:

```
MA0,0; PS; MA1000,1000; ...
```

The idea is that the carriage moves to the nominal Origin, then the operator changes the position a bit and presses SET ORIGIN to adjust workpiece/tool registration; then he/she presses PAUSE to begin the move sequence. Because of the one-command latency described above, the SET ORIGIN doesn't take effect until after the MA1000,1000; so the MA1000,1000; goes to the wrong place. The solution is to put a "dummy" do-nothing command after the PS, as follows:

```
MA0,0; PS; WA0; MA1000,1000; ...
```

If the machine is in the Emergency Stopped state when the PS command occurs, the command has no effect.

PZ [<zlevel>] — Power Level of Z Motor

The power level applied to the Z motor is altered immediately, according to the value of <zlevel>. If <zlevel> is zero or is omitted, current is removed from the Z axis motor; otherwise, full current is applied.

This power level lasts until the next PZ or the next Z axis motion occurs, at which time the Z motor is set to full power and left at full power after the motion.

At power up no current is applied to the Z motor until the first PZ, AZ, MZ, FZ command, or Z axis Arrow motion. The IN command does not affect the power level.

RE [<x res>, <y res>] **Resolution**

This command sets the number of microsteps per full step (the "microstep factors", or resolution) for the Automove System stepping motor drivers. All physical motion in the X and Y axes is affected, including vectors, arcs, Arrow buttons, and FH ("Find Home") motion.

A microstep is the smallest XY physical movement possible in the Automove System. For more information, see the section entitled "Microstep Size, Resolution, and Maximum Travel" in Chapter 1.

Whenever an RE command actually changes the microstep factors (i.e. sets a value different from the existing values) the System loses its Home position reference and sets the No Reference status bit. (See the OS command.) Thus if the RE command is to be used it should be the first command executed after power up, and should be followed by an FH command or a manual re-reference. (The most convenient way to establish the resolution is via personality parameters 21 and 22; see Chapter 7.)

The parameters <x res> and <y res> set the number of microsteps per full step in each axis. They must be in the range 1.0000 through 32.0000. For example, the value 32 causes a microstep to be one thirty-second of a full step, and the value 1 causes a microstep to be the same size as a full step.

Table 16 - Usable Values For <xres> <yres>

1.0000	1.0323	1.0667	1.1034
1.1429	1.1852	1.2308	1.2800
1.3333	1.3913	1.4545	1.5238
1.6000	1.6842	1.7778	1.8824
2.0000	2.1333	2.2857	2.4615
2.6667	2.9091	3.2000	3.5556
4.0000	4.5714	5.3333	6.4000
8.0000	10.6667	16.0000	32.0000

Values between these are interpreted as the nearest value in the above table. Values outside the range 1 through 32 cause an error.

If either <x res> or <y res> is less than 4.0000 the maximum speed is proportionally limited such that motion does not exceed a certain number of full steps per second. The approximate limits are as follows:

Single vectors:	approx. 7400 full steps/sec
Single arcs:	approx. 2200 full steps/sec
Continuous Path vectors and arcs:	approx. 2100 full steps/sec

Other speed limits may apply; see the SR command.

The microstep factors default to 8.0000 in both axes if <x res> and <y res> are omitted, and at power up, but not at IN. In a given RE command <x res> and <y res> must be either both specified or both omitted. The default value 8.0000 itself can be changed via the PE command, so that the System "wakes up" with the desired microstep factors at power up.

The RE command does not change the calibration factors established by the CF command or the linearity correction established by the CR command. In other words, the number of microsteps per Calibrated Unit remains unchanged. But, all other things being the same, the RE command does affect how much physical travel occurs for a given motion command.

When you change the microstep factors via RE, you will probably also want to change the acceleration, the step rate, and some of the personality parameters affecting Arrow button and FH motion. See the AC, SR, and PE commands and Chapter 7. In general, if you double the microstep factors you will get similar behavior if you halve personality parameters 2, 3, 8, 9, 12, and 13, and double personality parameters 6, 7, 10, 11, 14, 15, 17, 18, 23, 24, 25, 26, 32, and 33. However, your application determines the best values for these parameters.

SC <x>, <y>, <varid> [, <inverse>] Scale

This command computes the microstep equivalent of the Calibrated Units position in <x> and <y>. The computation takes into effect the current pattern rotation, calibration factors, Origin, and linearity correction. (See, respectively, the BP, CF, SO, and CR commands.)

<x> and <y> must be in the range -32768.0000 through 32767.9999. During the conversion from Calibrated Units to microsteps the intermediate and final computation results must be within the range -32768 through 32767. If this condition is not met, the out-of- range number is set to -32768 or 32767 and an error is logged.

The X microstep result is placed into the variable given by <varid>, and the Y result into the following variable. The results are always integers, and may be outside the travel limits. (See the TL command. For information about variables, see Chapter 10.)

If the <inverse> parameter is present and is nonzero, the inverse correction is performed; in other words, a microstep location in <x> and <y> is converted to the equivalent Calibrated Units position. In this case <x> and <y> are rounded to the nearest integer before the conversion is performed, and the results will be in the range -32768.0000 through 32767.9999.

Example: suppose you desire to begin a pattern whose Origin is to be at the microstep equivalent of (1000, 3000) Calibrated Units. You could use the following ACL commands:

```
MA 1000, 3000;  BP;
```

but this produces an undesired extra move to (1000, 3000). An alternative method uses the SC and SO ("Set Origin") commands and produces no motion:

```
SC 1000, 3000, 10;  BP;  SO @10, @11;
```

The SC scales into variables 10 and 11. The BP sets the Origin to the current carriage location, but this Origin is promptly overridden by the SO command. For the remainder of this pattern (i.e. until the EP "End Pattern" command) the Origin remains at this new location.

SO [<xorg>, <yorg>] Set Origin

The Calibrated Unit Origin position (the MA origin) is re- established at the microstep position given by <xorg> and <yorg>, relative to the Home position. The Origin Change status bit is cleared. The range of <xorg> and <yorg> is 0 through 32767.

Recall that the Home position is at (0,0) microsteps, where the Home switches closed during an FH or manual re-reference. <xorg> and <yorg> are absolute microstep values; in other words, they are offsets from the Home position, unaffected by the calibration factors, the linearity correction, or the previous location of the Origin position.

The origin defaults to (0,0) microsteps (i.e. the Home position) if <xorg> and <yorg> are omitted, and at IN or power up. In a given SO command, <xorg> and <yorg> must be either both specified or both omitted.

At power up the Home position is wherever the carriage is when motor power is first applied, so until an FH or manual re-reference occurs the Home and Origin positions will both be at unknown locations with respect to the platen.

The Origin affects only the AA, MA, OC, OT, and SP commands. It does not affect the existing microstep locations of the carriage, the travel limits, or the Taught Point. The SO command does not affect the value reported by the OA command for a given physical position, but does affect the values reported by the OC and OT commands.

If <xorg> or <yorg> is outside the travel limits, the GO TO ORIGIN button is disabled and the AA, AR, MA and MR commands cannot move to the Origin position.

See also the BP ("Begin Pattern") and EP ("End Pattern") commands.

For example, assuming 1000 microsteps per inch, the sequence FH; CF 1,1; SO 3000, 4000; MA 0,0; moves the carriage to a point approximately 3 inches to the right of and 4 inches above the lower-left corner of the platen. The OA command would give 3000,4000, although OC would give 0,0. The sequence CF 0.5, 0.5; SO 5000, 7000; MA -4000, -6000; moves the carriage to the same physical location, with the same OA result, but OC now gives -4000,-6000.

A suggested reason for moving the Origin manually is as follows. Suppose that when a workpiece is placed on the Automove platen it is sometimes in a different location than the nominal location, either due to mechanical tolerances or because there are several work areas on the platen. With the Arrow buttons move the carriage to, say, the lower-left corner of the workpiece and press the Set Origin button. Then the move sequence will be correctly registered to the workpiece. (Note that the SO command is not used here.)

SP <x>, <y> — Set Position Counters

The SP command sets the System's X and Y position counters to arbitrary new values, without causing any motion. This can be used to increase the travel in either direction beyond the normal 0 to 32767 microstep limits. For example, SP could allow travel at coordinates below the point where the Home switches closed, or it could increase the overall travel in a machine tool without sacrificing resolution. (Another way of increasing travel is via the RE command.)

Most normal operations, for example moving the (0,0) location to account for a tooling or setup location, should be done via the SO ("Set Origin") command. SP should only be used for increasing the travel in special situations. The SP command causes the System to "forget" where it is and where the Home switches are; thus the travel limits will no longer protect against the carriage crashing into tooling or into the edges of the X-Y table's platen. Also, the XY Linearity Correction will become invalid. If you use SP to move the carriage to a coordinate below where a Home switch originally closed, a subsequent FH command will continue to move in the negative direction and thus will not find the switch.

The <x> and <y> parameters are interpreted as Calibrated Units and must be in the range -32768.0000 through 32767.9999. The specified values are rotated by the pattern rotation angle. This location is then multiplied by the calibration factors, rounded to the nearest integer, and added to the Origin position to get a pair of microstep coordinates. These coordinates are adjusted via the Linearity Correction Table. The physical position counters are then set to these new values.

During the conversion from Calibrated Units to microsteps, the intermediate computation results must be within the range -32768 through 32767 and the final results must be within the travel limits. If either restriction is not met an error is logged and the position counters are not changed.

SR [<rate>] **Step Rate**

The value <rate> specifies the step rate, or speed, to be used in subsequent AA, AR, MA, MR, and MT commands, as well as Continuous Path motion, the GO TO ORIGIN vector, and the FH backoff vectors. The value is in microsteps per second. Zero is treated as 1.

Specifiable values are in the range 1 through 65535, although under most conditions the System cannot achieve 65535 microsteps per second. If a microstep is 0.001 inch the above numbers correspond to 0.001 through 65.535 inches per second. If <rate> is higher than the highest achievable speed, the System simply uses the highest speed it is capable of; see below.

The step rate defaults to 10000 at power up and IN, and if <rate> is omitted from the SR command. The default value 10000 itself can be changed via the PE command, so that the System "wakes up" with the desired step rate at power up.

During a given arc, vector, or Continuous Path move the step rate increases smoothly from zero to the specified value, at an acceleration rate determined by the AC command. Then there is a constant-speed "slew" phase, and a deceleration phase. Short moves may never reach the specified slew speed, since the entire move may be taken up by the acceleration and deceleration phases.

Diagonal vectors are performed at the specified speed; i.e., the X and Y motors travel at the appropriate speeds to make the diagonal speed equal to the specified value.

For arcs, the tangential speed (i.e. the speed along the path of the arc) will be the specified value, but never higher than approximately 17000 microsteps per second. If the radius is less than a certain value, the System automatically decreases the slew speed in order to limit the radial acceleration; see the AC command.

If <rate> is higher than the highest achievable arc speed then single vectors are performed at a higher speed than single arcs.

For a Continuous Path move, vectors are always performed at the same speed as arcs. This speed can never exceed approximately 16000 microsteps per second, and may be further limited by the smallest arc radius in the BC...EC sequence. See the BC command.

The maximum achievable step rate depends on the microstep factors (i.e. "resolution"). (For more information, see the RE command.) At the default corresponding to RE 8.00, 8.00; the approximate maximum speeds are as follows:

Vectors:	Approx. 59000 microsteps/sec
Arcs:	Approx. 17000 microsteps/sec
Continuous Path vectors and arcs:	Approx. 16000 microsteps/sec

Note that, although the controller electronics can slew at the above rates, the stepping motors themselves may not be able to keep up.

Above certain values of <rate> the System automatically performs microstep skipping during the moves. For more information see the AA, BC, and MA commands.

A single move or group of moves can be speeded up or slowed down through use of the VM command.

ST <x>,<y> Set Toggle head offset

ACL programs which use an SP command to create an offset cause the machine to operate in incorrectly calibrated platen space and also require homing between workpieces. The STx,y; command may be used with the same offset as the SP command and the machine will continue to operate in calibrated platen space. This command is useful with Toggle head valves. The x,y offset values will be subtracted from any move prior to linearity and calibration table corrections. An ST0,0; will cancel the offset.

Example: ST1000,1000;MA3000,3000; would cause the carriage to move to calibrated units position 2000,2000.

SZ <z> Set Z Position Counter

The SZ command sets the Z axis position counter to an arbitrary new value, <z>, without causing any motion. This can be used to increase the AZ command's travel range in either direction beyond the normal 0 to 32766 limits, or to allow AZ to travel beyond the point where the Z Home switch closed, or to provide a simple origin offsetting capability for the Z axis.

<z> must be within the range 0 through <mod> - 1, where <mod> is the Z axis modulus as set by the CZ command. If <z> is outside this range then an error is logged and the Z position counter is not changed.

The SZ command causes the System to "forget" where the Z Home switch was found. Thus, if you are counting on the Z axis modulus/travel limit to prevent the Z axis from crashing into things, you should use SZ with caution.

See the AZ, CZ, FZ, MZ, and OZ commands.

TD [<which>] **Toggle Digital Outputs**

This command is similar to the CD ("Change Digital Outputs") command; TD can immediately change any or all of the 8 Digital Outputs. The difference is that, while the CD command gives a specific new value to each Output, the TD command merely changes it from whatever value it has now to the opposite value; True changes to False, False to True.

The TD command is particularly useful in Download Sequences which are invoked from the front panel. The user can press a button to change the state of whatever hardware (valve, solenoid, etc.) is connected to a Digital Output. If you used CD instead of TD, you would need two separate buttons: one to turn it on, another to turn it off.

<which> is interpreted as an 8-bit number, with a useful range of 0 through 255. Each bit corresponds to one Digital Output. If the bit is 1, the Digital Output is toggled to the opposite state; if the bit is 0, the Digital Output is not changed.

For a table of the decimal value corresponding to each bit, see the CD command.

If <which> is omitted it defaults to 255 (whose binary equivalent is 11111111), meaning that all 8 Digital Outputs are toggled.

After the Digital Outputs have been set to their new values, the Automove System waits a certain amount of time before processing the next ACL command. This delay is specified via the WD command; it should be great enough to allow any associated external hardware to complete its action before the next motion begins, if necessary.

While the machine is in the Emergency Stopped state the TD command is ignored; all Digital Outputs remain in the False state.

If <which> is above 255, the least significant 8 bits are used.

For more information about the Digital Outputs see Chapter 8 Using the Digital Outputs and Inputs.

Example: assume the Digital Outputs are all False. Then the sequence TD 1; TD 3; first toggles bit 0 to the True state, then simultaneously toggles bit 0 back to False and bit 1 to True.

TL [<xmin>, <ymin>, <xmax>, <ymax>] **Travel Limits**

The TL command establishes a set of travel limits for the X and Y motors. The carriage can never be commanded to travel outside the rectangle defined by <xmin>, <ymin>, <xmax>, and <ymax> microsteps. (Exception: the Find Home Switches operation.) Each parameter must be in the range 0 through 32767.

Recall that the Home position is at (0,0) microsteps. <xmin>, <ymin>, <xmax>, and <ymax> are absolute microstep values; in other words, they are offsets from the Home position, unaffected by the pattern rotation angle, the calibration factors, or the location of the Origin position.

The max limit for each axis must be greater than or equal to the min value; if not, an error is logged. If the parameters are omitted, the System uses default values at IN and power up. The defaults are read from the non-volatile memory at power up. They can be changed via the PE command. At the factory the defaults are set approximately as follows:

Table 17 - Factory Defaults For Travel Limits

System as shipped	Approx. Travel Limit Defaults, Microsteps
Bare Controller	0,0,32767,32767
Controller with 6 x 6 inch X-Y Table	0,0,6000,6000
Controller with 12 x 12 inch X-Y Table	0,0,12000,12000
Controller with 18 x 18 inch X-Y Table	0,0,18000,18000

The 12 x 12 defaults, for example, are equivalent to executing TL 0,0,12000,12000; .

Note: For systems shipped with X-Y tables, the actual travel limit defaults are factory-set according to the measured calibration factors; thus, the values may differ slightly from those given in the table.

If an MA, MR, or MT command attempts to move the carriage outside the travel limits, an error is logged, the offending coordinate is set to the nearest in-range value and the carriage moves to the resulting point. Arcs are a bit more complicated; see the AA command.

Front panel Arrow button motion from within the travel limits is stopped at the edge of the travel limits. (Exception: at power up, arrow motion is unconstrained until the first FH command or manual re-reference.)

If the present carriage position is outside the newly-specified limit rectangle, MA and MR can be used to move it inside. An MR 0,0; for example, logs an error (since the Commanded position is probably outside the limits) and then move to the nearest edge of the rectangle. The Commanded position remains where it was, outside the travel limits.

While the carriage is outside the limits, the Arrow buttons are only able to move it closer to the rectangle, not further away. If the Origin position is outside the limits, the GO TO ORIGIN button is ignored.

The current travel limits can be read via the OL command.

Example: suppose that your application uses a vise to hold the workpiece on the Automove platen; the vise is located near the upper-right corner of a 12 x 12-inch platen, about 8 inches in X and Y from the lower-left corner. The command TL 0, 0, 7800, 7800; would prevent the carriage from hitting your vise due to a programming error or out-of-range data point.

VA <x1>, <y1>, <x2>, <y2>, <varid> — Compute Vector Angle

The VA command computes the angle of a vector which passes through the specified points. <x1> and <y1> define the start of the vector, while <x2> and <y2> define its endpoint; these parameters can range from -32768 through 32767 and are rounded to the nearest integer.

The angle is computed in degrees, with zero degrees being to the right (i.e., along the positive X axis) and positive angles being counterclockwise from zero. The angle result is placed in the variable given by <varid>; the range of values is 0.0000 through 359.9999.

If (<x1>, <y1>) and (<x2>, <y2>) are the same point then the VA command returns 0.0000 as the angle.

For more information about ACL variables, please see Chapter 10.

VC <varid> — Variable Capture

The VC command modifies the behavior of the next ACL output command (OA, OC, etc.). Instead of sending the numeric values to the host computer, the output command places the values into the variable whose i.d. is given by <varid> and, if necessary, in subsequently-numbered variables. For example, the sequence VC 5; OL; causes the current travel limits to be placed in variables 5, 6, 7, and 8.

An OI ("Output Identification") command following VC causes the numeric values of the two ROM version numbers to be captured. For example, if OI outputs AUTOMOVE REV 3.11, 3.1 the captured values will be 3.11 and 3.1.

If an IN ("Initialize") command occurs after the VC but before any ACL output command, the effect of the VC is canceled.

For program clarity it is suggested that the VC command be immediately followed by the ACL output command. In any case, if the VC is in a Download Sequence, the ACL output command must be in the same Download Sequence. The output command must not be in a Download Sequence called by the one containing the VC, and must not be in a Download Sequence which called the one containing the VC.

Note: VC does **not** work with escape sequences in the ES command. For example, there is no way to capture the output of the ESC.E ("Output Communications Error") command.

For more information about ACL variables, please see Chapter 10.

VL <x1>, <y1>, <x2>, <y2>, <varid> — Vector Length

This command computes the length of a vector. The Pythagorean theorem is used, where the length of the vector is the square root of: the square of the X coordinate change plus the square of the Y coordinate change.

The <x1> and <y1> parameters define one end of the vector, and <x2> and <y2> the other end. These parameters may range from -32768 through 32767 and are rounded to the nearest integer. The intermediate computation results (i.e. the squares and the sum of the squares) can be any values resulting from legal input parameters, but the final result must be in the range 0 through 32767. If the result would be greater than 32767 it is set to 32767.

The computed length is placed in the variable given by <varid>. The value returned is the largest integer which is not greater than the true square root.

For more information about ACL variables, please see Chapter 10.

VM [<mode>] **Vector Mode**

The VM command modifies the effects of all subsequent AA, AR, MA, MR, MT, and Continuous Path moves until another VM command is executed. The parameter <mode> is treated as a whole number whose bits have the following meanings:

Table 18 - <mode> Parameter Controls For Command VM

Bit	Dec.Val	Name	Meaning
0	1	No SR	If this bit is set the System temporarily performs X/Y motion at an alternate acceleration and step rate, which are typically 193 and 20000, respectively. (Note that single arcs cannot actually go 20000, so single vectors will go faster than single arcs; see SR.) The normal AC and SR parameters are reinstated when a VM is executed with this bit cleared. The factory-set alternate values 193 and 20000 can be changed via the PE command.
1	2	No MD	If this bit is set the System temporarily suspends pre-move and the mid-move digital Outputs changes as established by the PD, MD, and MM commands. They will be reinstated when a VM is executed with this bit cleared. Arcs and vectors are affected.
2	4	No AB	If this bit is set the System temporarily suspends antibacklash vectors as established by the AB command. They will be reinstated when a VM is executed with this bit cleared. Arcs and Continuous Path moves are not affected.
3	8	No MN	If this bit is set the System temporarily suspends the Mid-Move Digital Inputs Response as established by the MN command. The response is reinstated when a VM is executed with this bit cleared.

The other bits of <mode> are ignored; for compatibility with future enhancements, they should be set to zero.

The value of <mode> defaults to zero at power up and at IN, or if <mode> is omitted from the VM command. This default means that the SR step rate is used, and the MD, AB, and MN commands are not suppressed.

The parameters established by SR, PD, MD, MM, AB, and MN are not actually changed by the VM command; they are merely temporarily ignored. If an SR, PD, MD, MM, AB, or MN occurs while the corresponding VM bit is set, the SR, PD, MD, MM, AB, or MN is "remembered" and will take effect when the bit is cleared.

Example: VM 7; MA 5000, 6000; VM 0; causes one vector to be performed at alternate acceleration and speed with no pre- or mid- move Digital Outputs changes or antibacklash vectors. Subsequent moves are at normal speed with whatever Digital Outputs changes and antibacklash vectors were previously in effect.

VR <x1>, <y1>, <x2>, <y2>, <angle>, <varid> — Vector Rotate

This command computes the endpoint of a vector after it has been trigonometrically rotated about a specified starting point. <x1> and <y1> specify the starting point; they can range from -32768.0000 through 32767.9999. <x2> and <y2> are the endpoint of the vector before rotation, and have the same allowable range. <angle> is the number of degrees through which the vector is to be rotated. Positive angles result in counterclockwise rotation. <angle> may range from -32768.0000 through 32767.9999; rotations less than zero or greater than 360 degrees are increased or reduced modulo 360 to the corresponding angle in the range 0 through 359.9999 degrees.

The X coordinate of the rotated vector is put into the variable given by <varid>, and the Y coordinate into the following variable. If the rotation would produce an endpoint coordinate outside the range -32768.0000 through 32767.9999, the VR command returns the highest or lowest value within that range.

For more information about ACL variables, please see Chapter 10.

VS <varid> [, <value>] — Variable Set

This command places a new value into the variable whose i.d. is given by <varid>. The <value> parameter is optional. If specified, it can be a numeric constant such as 123.45 or it can be the value of another variable, such as @22. The resulting numeric value is placed directly into the variable.

If <value> is omitted the VS command waits for a numeric value to be received on the RS-232C port; this value is placed into the specified variable. Leading non-numeric characters are ignored and the number is terminated by any non-numeric character. (This is different from other ACL commands in that a control character such as Carriage Return or Linefeed terminates the VS number, whereas the Automove System normally ignores control characters.) Otherwise, the number should have the same syntax as any ACL command parameter; see Chapter 1.

A VS without the <value> parameter is primarily useful in downloaded move sequences. The numeric value would be specified, by a computer or some other device, at the time the move sequence is executed.

For more information about ACL variables, please see Chapter 10.

VT <varid> Variable Test

The VT command modifies the behavior of the next ON, XI, XU, or XW command, so that instead of testing the Digital Inputs, it tests bits 0 through 7 of the integer portion of the variable specified by <varid>. For example, suppose variable 6 contains the value 3, which is binary 11. The following sequence:

```
VT 6; XI 13, 2, 2;
```

tests bit 1 of variable 6; since bit 1 is set, the XI executes Download Sequence 13.

For program clarity, the VT should be immediately followed by the ON, XI, XU, or XW. In any case, if the VT is in a Download Sequence, the ON, XI, XU, or XW must be in the same Download Sequence. The ON, XI, XU, or XW must not be in a Download Sequence called by the one containing the VT, and must not be in a Download Sequence which called the one containing the VT.

The ON command behaves slightly differently than the others -- it tests all 16 bits of the integer portion of the variable. For example, if variable 22 contains 123.456 then the sequence:

```
VT 22; ON;
```

outputs the value 123. This feature can be used to separate the integer and fractional portions of a variable's value. For example, assuming variable 22 still contains 123.456, then the sequence:

```
VT 22; VC 5; ON; VS 6, @22; V- 6, @5;
```

puts 123 into variable 5. Then, it puts variable 22's contents (i.e. 123.456) into variable 6. Then it subtracts 123 from variable 6, leaving 0.456.

Note that VT does not affect the WN ("Wait For Digital Inputs") command.

V< <varid>, <value> — Variable Less Than

This command tests whether the contents of the variable whose i.d. is given by <varid> is numerically less than the specified <value>. The test result is available to the next ON, XI, XU, or XW command; bit 0 will be "1" if and only if the test was True (i.e. the variable was less than <value>). (The other 7 bits will be seen as "0".) For example:

```
V< 5, 100;  XW 22, 1;
```

executes Download Sequence 22 while variable 5 contains any value less than 100.

If an IN ("Initialize") command occurs after the V< but before any ON, XI, XU, or XW, the effect of the V< is canceled.

For program clarity, the V< should be immediately followed by the ON, XI, XU, or XW. In any case, if the V< is in a Download Sequence, the ON, XI, XU, or XW must be in the same Download Sequence. The ON, XI, XU, or XW must not be in a Download Sequence called by the one containing the V<, and must not be in a Download Sequence which called the one containing the V<.

If <value> is specified as an at-sign "@" and another <varid>, say, variable 2, then the value of this second variable is captured only once, at the time the V< command is executed. This means that an XU or XW will always test against the original value of variable 2, even if that variable has changed since the V< was executed.

Note that V< does not affect the WN ("Wait For Digital Inputs") command.

V= <varid>, <value> — Variable Equal To

This command is similar to the V< command, but instead tests whether the variable's contents are equal to the specified <value>. Bit 0 in the following ON, XI, XU, or XW will be "1" if the equality is True.

You should use caution when comparing two computed values for equality; small roundoff errors could result in unexpected inequality of two numbers that are very close to one another.

V> <varid>, <value> — Variable Greater Than

This command is similar to the V< command, but instead tests whether the variable's contents are greater than the specified <value>. Bit 0 in the following ON, XI, XU, or XW will be "1" if the greater-than comparison is True.

V+ <varid>, <value> — Variable Add

This command adds a specified value to the variable whose i.d. is given by <varid>. Saturation arithmetic is used; in other words, the result is never greater than 32767.9999 or less than -32768.0000. If the correct result of the addition would have been outside of this range, an error is logged and the variable is set to the highest or lowest possible value.

V- <varid> [, <value>] — Variable Subtract/Negate

If <value> is specified, this command subtracts the specified value from the variable whose i.d. is given by <varid>. As with V+, saturation arithmetic is used.

If <value> is omitted, the contents of variable <varid> are negated, with the result being placed back into the same variable. The value -32768.0000, when negated, becomes the same value.

V* <varid>, <value> — Variable Multiply

This command multiplies the variable whose i.d. is given by <varid>, by a specified value. As with V+, saturation arithmetic is used.

V/ <varid>, <value> — Variable Divide

This command divides the variable whose i.d. is given by <varid>, by a specified value. As with V+, saturation arithmetic is used.

V& <varid>, <value> — Variable AND

This command performs a bitwise logical AND between the specified value and the variable whose i.d. is given by <varid>. The result is placed back into the same variable.

The result is always an integer. Any fractional part of <value> or the variable's value is ignored.

V| <varid>, <value> — Variable OR

This command performs a bitwise logical OR between the specified value and the variable whose i.d. is given by <varid>. The result is placed back into the same variable. (The second character of the mnemonic is the ASCII Vertical Bar character, "|".)

The result is always an integer. Any fractional part of <value> or the variable's value is ignored.

V! <varid> [, <value>] — Variable NOT/Exclusive OR

If <value> is omitted, this command performs a bitwise logical ones'-complement on the variable whose i.d. is given by <varid>. The result is placed back into the same variable.

If <value> is specified, the command performs a bitwise logical exclusive-OR between the value and the variable whose i.d. is given by <varid>. The result is placed back into the same variable.

The result is always an integer. Any fractional part of <value> or the variable's value is ignored.

WA <wait> Wait

The Automove System waits approximately <wait> seconds before processing the next ACL command. The range of <wait> is 0 through 65.535; the resolution is 0.001 second (one millisecond). The delay begins when the previous command has been completely processed, or when the WA is received, whichever comes later.

The WA command is ignored if the machine is in the Emergency Stopped state. Once a wait has begun it can be aborted via Emergency Stop or the ESC.K command.

If the time spacing between two actions is critical in your application, the WA command may be useful. Because of the Automove System's input buffer, even if the host computer delays several seconds between sending two commands, the two commands may be executed back-to-back, one right after the other.

On the other hand, if it is crucial that two commands be executed one right after the other (perhaps with some WA-induced amount of delay between them) you may wish to lock out the front panel while the time-critical section is executing, then re-enable it later. See the FP command, and also Tricks in Appendix E Speed Considerations.

If many CD or TD commands are to be followed by identical delays, the WD command may be more convenient than following each CD or TD with a WA. Also, the PD command can insert precise, uninterruptable delays between Digital Outputs changes and subsequent moves.

WD [<delay>] Wait after Digital Outputs Change

The WD command establishes an implicit delay which is to be executed after each CD or TD command, before processing the subsequent command. <delay> is in seconds; the allowable range is 0 through 6.5535. The resolution is 0.0001 second.

The implicit delay defaults to 0 at power up and at IN, or if <delay> is omitted from the WD command.

If you need a precise delay between a Digital Outputs change and a vector or arc motion, use the PD ("Pre-move Digital Outputs Changes") command.

For more information about the Digital Outputs, see Chapter 8 Using the Digital Outputs and Inputs.

WN <value> [, <which> [, <timeout>]] Wait for Digital Inputs

This command causes the Automove System to delay before processing the next ACL command. The delay is terminated when a certain condition occurs on any or all of the 8 Digital Inputs. If the condition already exists at the time the WN command is executed, there is no delay.

The <value> and <which> parameters are treated as 8-bit numbers and are similar to the <new> and <which> parameters of the CD command, respectively. That is, the System waits until each Digital Input matches the corresponding bit of <value>, where a True input matches a 1 bit and a False input matches a 0. However, the System only checks those Digital Inputs which correspond to a 1 bit in <which>. All other Digital Inputs are ignored and do not need to match.

The <which> parameter is optional; if omitted it defaults to 255 (i.e. binary 11111111), meaning that all 8 Digital Inputs must match <value>.

The <timeout> parameter is also optional, but cannot be specified unless <which> is also specified. <timeout> can be any value in the range 0 through 65.535; it specifies the maximum amount of time, in seconds, to wait before giving up and proceeding to the next ACL command. If <timeout> is zero or is omitted then there is no timeout; i.e., the System will potentially wait forever for the specified condition to occur.

The WN command is ignored if the System is in the Emergency Stopped state. Once a wait has begun it can be aborted via Emergency Stop or the ESC.K command.

The host computer can directly determine the current values of the Digital Inputs via the ON command. Like all other ACL commands, of course, ON won't work during a WN wait.

The Digital Inputs can be enabled to trigger certain functions automatically upon a False-to-True transition. See personality parameters 36 and 37 in Chapter 7. If any such transitions occur during the WN command, regardless of whether the WN command is looking at the particular input(s) which changed, the associated functions are performed after the wait, even if the triggering Digital Input has meanwhile gone False again. (This contrasts with the fact that transitions on the Digital Inputs are usually ignored during motion commands such as vectors and arcs. See the MN command and Chapter 8.)

During the WN command the Digital Inputs are checked approximately every 36 microseconds (0.000036 second). If a pulse of shorter duration occurs on an Input, the System may not see it.

If you need to check the Digital Inputs during a vector or arc, see the MN ("Mid-move Digital Inputs Response") command.

Example: the command WN 1, 1; waits until Digital Input 0 is True. It does not pay attention to any other Inputs. (The 1's correspond to binary 00000001, which has only bit 0 set.)

Example: the command WN 0, 4; waits until Digital Input 2 is False. (The 4 corresponds to binary 00000100, which has only bit 2 set.)

Example: the command WN 2, 3, 4.6; waits until Input 1 is True and Input 0 is False. If this condition has not occurred within 4.6 seconds the wait is terminated.

For more information about the Digital Inputs see Chapter 8 Using the Digital Outputs and Inputs.

XD [<id> [, <rept>]] Execute Download Sequence

This command invokes, or starts the execution of, a Download Sequence previously defined via the BD and ED commands. The saved ACL commands are executed as though they had been received from the host computer through the RS-232C port.

The XD command itself can be made part of a Download Sequence. In this way complex motion sequences containing repetitive patterns can be built up using only a small amount of download memory. The maximum nesting depth is twelve; in other words, if sequence 1 invokes sequence 2 and sequence 2 invokes sequence 3, etc., then if sequence 12 attempts to invoke sequence 13 an error is logged and Download execution halts. However, sequence 1 can invoke sequence 2 a hundred times in a row if it is desired.

If a Download Sequence directly or indirectly invokes itself the maximum nesting depth will probably soon be exceeded. Exception: if there is an XI ("Execute If") in the sequence it may terminate the nesting before it becomes too deep. See the BD command.

<id> is the identifying number of the Download Sequence to be invoked; it must be an integer in the range 0 through 255. <id> is optional; if omitted it defaults to zero (i.e. Download Sequence 0 is invoked). If the specified Download Sequence is empty (i.e. has not been defined via BD and ED) then the XD command has no effect.

<rept> is the repetition count; it is optional but cannot be specified unless <id> is also specified. If <rept> is specified it must be an integer in the range 0 through 65535; this value overrides the repetition count given in the BD command. If <rept> is omitted the BD command's repetition count, if any, is used.

If <rept> is 1 the invoked sequence is executed once; if greater than one the sequence executes the specified number of times. If <rept> is zero the sequence repeats forever, or until it is halted; see below.

The execution of a Download Sequence can be prematurely halted four ways: by resetting the machine or turning off the power; by executing an AD command contained in any Download Sequence; by sending the ESC.!5: command (see Chapter 6); or by exceeding the maximum nesting depth (see above). In addition, a Download Sequence can be interrupted by another Download Sequence invoked via the front panel or the Digital Inputs; see Chapter 8.

For more information about storing, invoking, and executing Download Sequences, see the BD, ED, XI, XU, and XW commands. Also see personality parameters 36 - 38 in Chapter 7.

Example: the sequence BD 2; AA 0, 0, -10; CD 1; WA 0.1; CD 0; ED; defines Download Sequence 2, which performs a 10-degree clockwise arc centered about the Origin location (0,0), then changes the Digital Outputs to 1, then waits 0.1 second, then changes the Digital Outputs back to zero. After setting up this sequence, send FH; SO 2000,2000; MA0,1000; XD2,36; . This sets the Origin at (2000,2000), moves the carriage to a point 1000 Calibrated Units above the Origin, and steps to 36 points around a circle, pausing at each one to change the Digital Outputs.

Example: after doing the previous example, send BD 0, 3; MR 0, 1000; XD 2, 36; ED; . Then, using the Arrow buttons, move the carriage to some location in the middle of the platen, and press the SET ORIGIN button. Then press FAST and, while holding it, press PAUSE. This invokes Download Sequence 0, which executes 3 times and then quits. Each time it executes it performs a circular pattern of 36 moves, each pattern with a radius 1000 Calibrated Units larger than the previous one.

XI <id>, <value> [, <which> [, <else id>]] — Execute If

This command invokes a Download Sequence in the same way as the XD ("Execute Download Sequence") command, except that XI only invokes the Sequence if a certain condition currently exists (or does not exist) on the Digital Inputs or in an ACL variable.

(The discussion below assumes that the Digital Inputs are being tested. For information about testing ACL variables, please see the VT, V<, V=, and V> commands, as well as Chapter 10.)

<id> is the identifying number (0 through 255) of the Sequence to be invoked if the condition is met. This is like the <id> parameter of the XD command.

<value> and <which> establish the condition to be tested for. They are identical in meaning to the <value> and <which> parameters of the WN ("Wait For Digital Inputs") command. That is, <value> specifies the values to compare against the Digital Inputs, and <which> specifies which Digital Inputs to check and which to ignore.

<which> is optional. If omitted it defaults to 255, meaning that all 8 Digital Inputs are checked.

<else id> is also optional, but cannot be specified unless <which> is also specified. <else id> is the identifying number (0 through 255) of a Download Sequence which is to be executed if the condition is not met. In other words, either <id> or <else id> is used, but not both. If <else id> is omitted and the condition is not met, no Download Sequence is invoked; i.e., the XI command has no effect.

Unlike the XD command, the XI command does not allow a repetition count to be specified. The Download Sequence can only be executed once, unless its BD had a repeat count.

For information about debouncing, signal sense (High True vs. Low True), and the Digital Inputs in general, see Chapter 8 Using the Digital Outputs and Inputs.

Example: the command XI 22, 4, 4; invokes Download Sequence 22 if and only if Digital Input 2 is presently True. If it isn't, the command has no effect. (The value 4 is equivalent to binary 00000100, which has only bit 2 set.) All other Inputs besides Input 2 are ignored.

Example: the command XI 13, 0, 255, 15; invokes Download Sequence 13 if all 8 Digital Inputs are currently False. But if any Digital Input is True, Download Sequence 15 is invoked instead. (The 255 corresponds to binary 11111111, which has all eight bits set.)

XU <id>, <value> [, <which>] **Execute Until**

This command invokes a Download Sequence in the same way as the XD ("Execute Download Sequence") command, except that XU repeats the Download Sequence until a specified condition exists on the Digital Inputs or in an ACL variable. Any repeat count specified in the BD command is ignored.

(The discussion below assumes that the Digital Inputs are being tested. For information about testing ACL variables, please see the VT, V<, V=, and V> commands, as well as Chapter 10.)

<id> is the identifying number (0 through 255) of the Sequence to be repeated. This is like the <id> parameter of the XD command.

<value> and <which> establish the condition to be tested after each execution. They are identical in meaning to the <value> and <which> parameters of the WN command. That is, <value> specifies the values to compare against the Digital Inputs, and <which> specifies which Digital Inputs to check and which to ignore.

<which> is optional. If omitted it defaults to 255, meaning that all eight Digital Inputs are checked.

The Digital Inputs are tested after each execution of the Download Sequence. If the condition is met the Download Sequence quits repeating and execution resumes with the ACL command following the XU command. If the condition is not met the Download Sequence is repeated one more time, and the test is made again. Thus the XU command always executes the Download Sequence at least once.

For information about debouncing and signal sense (High True vs. Low True) see Chapter 8 Using the Digital Outputs and Inputs.

Example: suppose your application is to drill several holes in a block of wood. Download Sequence 27 contains the commands to pull the drill bit out of the hole, turn on compressed air to blow away the chips, and plunge the drill back into the hole. A microswitch detects if the drill has gone all the way through the wood yet. Assuming that the microswitch is connected to Digital Input 0, you might use XU 27, 1, 1; to repeat the cycle until a hole is finished. Then you could move the carriage to another location and do another XU 27, 1, 1; to drill another hole.

XW <id>, <value> [, <which>] **Execute While**

This command repeats a Download Sequence in the same way as the XU ("Execute Until") command, except that the termination condition is slightly different. As with XU, any repeat count specified in the BD command is overridden.

(The discussion below assumes that the Digital Inputs are being tested. For information about testing ACL variables, please see the VT, V<, V=, and V> commands, as well as Chapter 10.)

<id> is the identifying number, 0 through 255, of the Sequence to be repeated. <value> and <which> specify the condition to test before each execution; see the WN command for the meaning of these parameters. <which> is optional; it defaults to 255 if omitted, meaning that all eight Digital Inputs are checked.

The Digital Inputs are tested before each execution of the Download Sequence. If the condition is met the Download Sequence is executed. But if the condition is not met the Download Sequence is not executed any more and execution resumes at the ACL command following the XW command. Thus, the XW command might not execute the Download Sequence at all if the condition is not met the very first time the Inputs are tested.

For information about debouncing and signal sense (High True vs. Low True) see Chapter 8 Using the Digital Outputs and Inputs.

Example: suppose you are using the Automove System to dip integrated circuit leads into a vat of molten solder. The IC's are fetched, one at a time, from a feeder station. A microswitch at the feeder station closes when the station is empty; i.e., when there are no more IC's left. The microswitch is connected to Digital Input 2.

Suppose you have set up Download Sequence 16 to fetch a part, dip it in the solder, and drop it into a bin. You do not wish to execute Sequence 16 if the feeder has emptied. You might use XW 16, 0, 4; to process IC's as long as there are any left; i.e., as long as the microswitch is open. If you had used the XU ("Execute Until") command (XU 16, 1, 4;) your system would have performed Sequence 16 once before discovering that the feeder is empty.

ZM [<count>,<new>[,<which>]] Z-axis Multiple Mid-move Digital Outputs

This command will perform the same action as the MD command only it is applied to the Z axis. This feature allows the dot dispense cycle to occur during the Z axis decent to a dispense point which speeds up the dispense operation.

The parameters syntax is the same as the MM command.

* Dispense Dot

This is a single character command which will perform the equivalent of the following series of ACL commands to speed up the dispensing for dots:

VM0:MR0,0;VM15;

At 19,200 baud this command could save 7 MS per dot. At 9,600 baud it could save 14 MS per dot. The asterisk was chosen because it looks something like a dot to help you remember it.

The PD and MD commands must have been previously executed and the * command must be the first character of a new command.

4 Front Panel Functions

This chapter describes the functions of the buttons on the Automove System's front panel.

All buttons produce immediate effects if they are pressed while the Automove System is sitting idle. If pressed during carriage motion the RESET, STOP, and PAUSE, buttons take effect immediately; other buttons must be held down until the current vector, arc, or Continuous Path move is completed or the System does not react to them.

All buttons except RESET can be locked out under command of the host computer. See the FP command in Chapter 3.

The FAST button (the unlabeled button at the center of the four Arrow buttons) is like the Shift key on a typewriter; it must be held down while another button is pressed. Be careful about pressing anything other than the Arrow buttons with the FAST button; see "Execute Download Sequence", "Manual Re-reference", and "Suppress Autostart", below.

RESET

This button puts the Automove System into the power up state. For a list of all the effects, see Appendix A Power Up Actions. If there is an Autostart Sequence defined it starts executing. (See personality parameter 39 in Chapter 7.)

You can prevent the Autostart sequence from executing if desired; see "Suppress Autostart", below.

STOP

STOP is activated when the STOP button is pressed or an external Emergency Stop switch actuates. The machine immediately stops all three motors, sets all Digital Outputs to the False state, begins blinking the STOP light on the front panel, and sends a "?" character to the host computer. (See "Exceptional Conditions" in Chapter 5.) The PAUSE light stays on solidly as long as the Stop switch remains actuated.

During the Emergency Stopped state all physical action commands (AA, AR, AZ, CD, FH, FZ, MA, MR, MT, MZ, TD, WA, WN, Continuous Path moves, Arrow buttons, GO TO ORIGIN button, manual re-reference) are ignored.

The Emergency Stop, No Reference, and No Z Reference status bits are set. The Paused state is cleared. The motor power level is not changed. If a Download Sequence is executing it continues to execute, although the motion commands will be ignored.

If the STOP light glows steadily (i.e. doesn't blink) then the STOP button has been locked out via the FP ("Front Panel Lockout") command. In this case the machine will not actually stop until another FP is executed to re-enable STOP. You do not need to press the STOP button a second time to make it stop.

Eventually a CS, IN, or ESC.!2: must be executed in order to re-enable motion. There is no front-panel way to clear the Emergency Stopped state, other than to reset the machine or cycle the power switch. The physical position reference is usually lost if the motors are moving when the STOP occurs, but the No Reference status bit is always set regardless. Therefore, after the Stop is cleared, an FH command or manual re-reference should be performed before any more XY motion, and an FZ before any Z motion.

The Emergency Stopped state can be detected by the host computer through the use of the ESC.O command or the OS command; the ESC.O reports back immediately, whereas the OS is not processed until any preceding ACL commands in the input buffer have been processed.

The host can remotely force the Automove System into the Emergency Stopped state; see the ESC.! command. The effect is the same as if the STOP button had been pressed at the instant the ":" of the ESC.!1: command was received.

PAUSE

When the PAUSE button is pressed, the machine finishes any ACL command it was executing and enters the Paused state. The PAUSE light on the front panel begins to blink, and the machine suspends the processing of ACL commands. When the button is pressed a second time, the light turns off and the machine resumes processing commands.

No ACL commands are lost during the Paused state if the host computer is properly executing an input buffer handshake. See Chapter 5. If the Automove System is executing a Download Sequence it pauses and resumes the Download Sequence; nothing is lost.

The PAUSE button is ignored while the machine is in the Emergency Stopped state. The Paused state is canceled when the STOP button is pressed or an external Emergency Stop switch closes.

If the PAUSE light glows steadily (i.e. doesn't blink) then one of two things has happened: a) The PAUSE button has been locked out via the FP ("Front Panel Lockout") command, and the button has subsequently been pushed. In this case the machine will not actually pause until another FP is executed to re-enable the PAUSE button. You do not need to press the PAUSE button a second time to make it pause. b) The STOP button is being pushed, or an external Stop switch is currently actuated. (In this case the STOP light will be blinking.) The PAUSE light will continue to glow until the STOP button or external switch is released.

The host computer can detect the Paused state through the use of the ESC.O command. The host can also force the Automove system into or out of the Paused state. See the PS command in Chapter 3 and the ESC.! and ESC.K commands in Chapter 6.

SET ORIGIN

This button causes the current carriage position (X,Y) to become the Calibrated Unit Origin position, and the Origin Change status bit to be set. See Numeric Coordinate Systems in Chapter 1.

If a pattern is active then the new Origin position is effective only until the end of the pattern. When the EP ("End Pattern") command is executed, the old location of the Origin is reinstated. See the BP and EP commands in Chapter 3.

During Z Arrows mode (see "Z Axis Arrows", below) the SET ORIGIN button has no effect, since the Z axis origin always remains at the Z Home position.

GO TO ORIGIN

The machine performs a vector to the Origin position, as though an MA 0,0; command had been received. The motors are left at full power. The Position Change status bit is set.

If a pattern is active the carriage moves to the pattern Origin rather than the "main" Origin. (See the BP command in Chapter 3.)

If personality parameter 48 is set properly the vector to the Origin is preceded by a Find Z Home operation, and is followed by a Z move to the original Z location. This can help prevent mechanical interference during the vector.

You can use this button to find out where the Origin position is, or simply to move the carriage to the Origin for any reason.

The Digital Outputs are unaffected by this vector, regardless of the MD and VM commands. An antibacklash vector is performed if it is enabled by the AB command and not disabled by the VM command. The acceleration and step rate are as established by the AC and SR commands. No mid-move Digital Inputs Response is active; see MN.

If the Arrow Buttons are in Z mode (see "Z AXIS", below) then GO TO ORIGIN causes the Z motor to move directly to the Z Home position. X and Y are not affected. Note that the Z origin always remains at the Z Home position, so the SET ORIGIN button does not affect the Z axis. Also note that it is possible for the Z axis to be "below" its Home position; in this case GO TO ORIGIN will move in the negative direction to the next position where Z is zero. Caution: this could be a long way if the Z axis modulus is large; see the CZ command.

The GO TO ORIGIN button is ignored while the machine is in the Emergency Stopped state, but works normally during the Paused state. It is also ignored (in XY mode) if the XY Origin position is outside the travel limits. (See the TL command.)

TEACH

The machine saves the current XY position as a Taught Point and sets the Taught Point Available status bit. If the host has previously sent an OT ("Output Taught Point") command, the taught point coordinates are immediately sent to the host and the Taught Point Available status bit is cleared.

If the Taught Point Available status bit was already True, the TEACH button has no effect; thus, each taught point must be "used" (via OT or MT) before the TEACH button can be pressed again.

The AM ("Arrow Mode") command in Chapter 3 describes a way of using the TEACH button to teach Z axis points. The MN ("Mid-move Digital Inputs Response") command can teach the current position automatically during a move. The MT ("Move To Taught Point") command moves the carriage back to the most recent Taught Point.

Arrow Buttons

The Arrow buttons are four buttons (Up, Down, Left, Right) for producing manual carriage motion in X and Y or Z.

When an Arrow button is pressed, the carriage steps one physical motor count (i.e. one microstep). If the button is held down for a short while, the carriage begins to move slowly. If the FAST button is pressed along with an Arrow button, the carriage immediately begins to move at a higher speed. (The FAST button is the unlabeled button in the center.) When all Arrow buttons are released, the motors are stopped immediately and are left at full power. The Position Change status bit is set. (See the OS command in Chapter 3.)

Immediately after power up, Arrow motion is unconstrained. As soon as the first FH command or manual re-reference has executed, XY Arrow motion is constrained to the travel limits. If the carriage is presently outside the travel limits, the Arrow buttons are only able to move it closer to the travel limit rectangle, not further away. (See the TL command.)

The Arrow buttons are ignored while the machine is in the Emergency Stopped state, but they work normally during the Paused state.

All of the speeds and delays associated with the Arrow buttons can be changed via the PE command. Also, the functions of the Left and Right Arrow buttons can be swapped, or the Up and Down buttons.

Z AXIS

This button changes the effects of the Arrow buttons and the SET ORIGIN and GO TO ORIGIN buttons. When you push Z AXIS the machine goes into Z Arrow mode and turns on the lights in the Up and Down Arrow buttons. The second time you push the button the lights go off and the Arrows revert to their normal XY mode.

In Z Arrow mode the Up and Down Arrows control the Z motor instead of the Y motor. The GO TO ORIGIN button causes the Z motor to move to its Home position. The manual re-reference (FAST/GO TO ORIGIN -- see below) causes the Z motor to seek its Home switch and sets the Position Change status bit. SET ORIGIN has no effect, since the Z axis origin always remains at the Z Home position. Left and Right Arrows also have no effect.

After power up until the first FZ command or manual Z re-reference, Z axis Arrow motion is unconstrained. After the Z Home switch has been found, Z Arrow motion is limited by the Z axis modulus. (See the CZ command.)

Z Arrow mode can also be set or canceled via the AM ("Arrow Mode") command. See Chapter 3.

Joystick

The optional external joystick produces the same effects as the Arrow buttons. If you move the stick away from its center position the motor takes one step, and after a while begins moving slowly. The button at the end of the stick has the same effect as FAST. The two buttons on the joystick housing are both TEACH buttons.

If the Arrow Buttons are in Z mode then the joystick affects the Z motor instead of X and Y.

Manual Re-reference

If the FAST button is held down while the GO TO ORIGIN button is pressed, the System performs a manual re-reference operation. First the machine finds the X and Y Home switches, as though an FH command had been received. Then the machine performs a vector back to the position it "thought" it was at before the button was pressed. The No Reference status bit is cleared. (See the OS command in Chapter 3.)

This sequence can be used to reestablish the position reference if it has been lost, for example due to the carriage having "crashed" into a fixed object. The host computer program or Download Sequence does not necessarily have to be restarted.

A manual re-reference can also be used after reset or power up, to provide a known position reference and travel limits for manual (Arrow button) XY carriage motion and for setting the Origin. However, it is recommended that the host program always execute the FH command as the first command after reset or power up, to be sure that the position reference has been established.

The direction, speed, half-step mode, and maximum distance can be changed via personality parameters 31, 44, 46, and 16, respectively; see Chapter 7.

If the Arrow Buttons are in Z mode then the manual re-reference causes the Z motor to seek the Z Home switch. (It does not return to where it "thought" it was.) X and Y are not affected. The No Z Reference status bit is cleared. (See the OS command.)

The XY manual re-reference can be configured to automatically find Z Home first, for example, to prevent crashing into fixturing during the XY motion. See personality parameter 48.

Invoke Download Sequence

If the FAST button is held down while the PAUSE or TEACH button is pressed, the System invokes Download Sequence 0 or 1, respectively. The Download Sequence executes as many times as specified by the <rept> parameter of the BD command which created it, and then it quits. See the BD and ED commands in Chapter 3.

While the Download Sequence is executing, all front panel buttons have their normal functions but the System does not process ACL commands from the host computer.

If you press FAST/PAUSE or FAST/TEACH while any Download Sequence is executing, the System interrupts the old sequence and immediately starts executing the newly requested one. The old sequence resumes executing when the new one is finished.

(One restriction applies to interrupts: you cannot interrupt an interrupt. In other words, if a first sequence has been interrupted by a second and you try to invoke a third, the third sequence is held off and does not begin executing until the second finishes executing. For more information see Chapter 8.)

Program Select and GO

The Program Select thumbwheel switch and GO button can invoke any Download Sequence from 0 through 9. Just press the upper or lower thumbwheel button until the desired program number appears. Then press GO. The Download Sequence runs until it is finished or you press the PAUSE or RESET button.

Front panel buttons and interrupts behave as described above under Invoke Download Sequence.

Self Test

If the Left Arrow and Right Arrow buttons are both depressed when the power switch is turned on, the machine begins to execute a Self Test sequence which exercises X, Y, and Z motors, Digital Outputs, and lights. This sequence repeats forever.

If the Up Arrow is also pressed at power up, the self test sequence skips the Z axis portion of the test routine.

You must hold the buttons for several seconds after turning on the power to be sure the self test is activated.

Suppress Autostart

When you turn on the power or press RESET the Automove System may automatically execute an Autostart Download Sequence; see personality parameter 39 in Chapter 7.

You can prevent the Autostart sequence, if any, from executing; simply keep the FAST button depressed while you reset or turn on the power. You must hold the button for several seconds to be sure the Autostart sequence does not execute.

5 RS-232C Communications

The RS-232C Interface

The Automove System is connected to the host computer via an RS-232C/CCITT V.24 serial interface, herein referred to simply as RS-232C. The cable should be fitted with the standard male 25-pin type "D" subminiature CINCH DBC-25P connector or equivalent.

The Automove System functions as a DTE (Data Terminal Equipment) on RS-232C. It transmits data on pin 2 (Transmitted Data) and receives it on pin 3 (Received Data). The Automove System always drives pin 4 (Request To Send) to the High state. With special factory modification, pin 5 (Clear To Send) can be used to temporarily inhibit the Automove System from transmitting characters to the host. Pin 7 is Signal Ground. Pin 20 (Data Terminal Ready, or DTR) is used in conjunction with the Hardwired DTR handshake (if enabled -- see below).

If the Hardwired DTR handshake is explicitly disabled, the Automove System continuously drives pin 20 to the High state.

The Automove System can be told to temporarily ignore incoming data on pin 3; see the ESC.) and ESC.(commands in Chapter 6.

Baud Rate

In general, the baud rate should be set to 9600 unless a lower rate is dictated by host hardware or software limitations, long cable length, or electrical noise considerations. See Appendix E Speed Considerations. The speed at which the host can receive and process incoming data from the Automove System should not be a limiting consideration here, as the Automove System can be instructed to delay and slow down its transmissions to the host. See ESC.M and ESC.N in Chapter 6.

Character Format

The host computer sends ACL commands and escape sequences to the Automove System as ASCII characters. Each character should be sent with a start bit, 7 data bits, an optional parity bit as established by the backpanel switches (see the Automove Operation Manual), and one or two stop bits. Slightly higher throughput will result if only one stop bit is sent. It is recommended that a parity bit always be used, because there is no other mechanism for detecting transmission errors.

When a parity, framing, or overrun error is detected in received data, a communications error is logged (see ESC.E) and a "?" is sent to the host, but the incoming character is used anyway, as received. Thus if the host is sending the wrong parity the Automove System logs one error but works correctly. (Subsequent communications errors will not be logged. Again, see the ESC.E command.)

Outputs to the Host

When the Automove System sends data back to the host it also uses 7-bit ASCII. Transmitted parity is the same as the received parity established by the backpanel switches. The Automove System always transmits two stop bits, although it is happy to receive either one or two.

Many host computers have only one or two characters of buffering on their RS-232C input channel. This means that, if the host software does not read and process the characters as fast as the Automove System sends them, data will be lost. If your host software is losing characters you can ask the Automove System to delay and slow down its transmissions. See the turnaround delay in the. ESC M command and the intercharacter delay in the ESC.N command, Chapter 6.

When an ACL command or escape sequence requests status information the Automove System sends one or more numeric values to the host. Multiple values are separated by commas. A negative number is preceded by a minus sign. Following the last value, the System sends the Output Terminator sequence. At power up this sequence defaults to a Carriage Return (ASCII code 13) followed by a Linefeed (ASCII code 10); it can be changed via the ESC.M command. See Chapter 6.

In general, when the host requests output data (via either an ACL command or an escape sequence), it should await the response before sending any more characters (including handshake characters) to the Automove System. This is not an absolute rule but, if followed, may prevent host/Automove handshake deadlocks and unpredictable sequences of status information change.

It is a good idea for the host program always to read the data into a string variable instead of directly into a numeric variable. That way if an Exceptional Condition (see below) occurs or has occurred, the "?" will be the first character of the string, and the host program can simply check the string before trying to decode the numbers themselves. (Some BASIC interpreters, for example, don't like getting a question mark when they are expecting a number.)

Exceptional Conditions

When the Automove System detects certain exceptional conditions, it immediately sends a "?" (question mark) character to the host. These conditions are:

* Detecting invalid non-volatile memory contents at power up or RESET
* Entering the Emergency Stopped state
* Detecting motor slip via the optional Step Verify circuit
* Detecting any communications error (see ESC.E)
* Detecting any ACL error (see OE)

When the host receives a "?" it can interrogate via the OS, OE, ESC.E and ESC.O commands to determine the cause. Remember that OS and OE do not execute immediately if any ACL commands precede them in the input buffer.

In the case of erroneous ACL commands the "?" may be transmitted before the Automove System has received all of the characters associated with the bad command; for example, if the first of two parameters is out of range. Also, due to the Automove System's input buffer, the host may have already sent several more commands by the time an erroneous command causes a "?" to be sent.

The host software must be able to handle and respond to the unsolicited "?" character even while the host is expecting data from some output request previously sent to the Automove System. However, the "?" never occurs in the middle of the output sequence generated by a given output request; it always comes between outputs. The "?" never causes an "output conflict"; see the. ESC E command in Chapter 6.

For example, assume that a series of OA commands are causing the Automove System to repeatedly output 1234,5678 plus a Carriage Return and Linefeed; if the STOP button is pushed, the "?" is always sent to the host between a Linefeed and the following "1"; never within a given OA's output string. Thus if the host is putting the data into a string variable, the string will contain ?1234,5678 instead of the expected 1234,5678.

The Input Buffer and Handshakes

As ACL commands are received they are placed in a 256-byte first-in/first-out input buffer. Whenever the buffer contains any characters, the System extracts and processes them as fast as possible, in the order they were received.

In order to prevent the buffer from overflowing, the host must implement one of four types of handshake. These are: Hardwired DTR, Xon/Xoff, Enq/Ack, and Software Checking. At power up the Automove System defaults to the Hardwired DTR handshake. To use a different handshake, the host must send one or more escape sequences to the System. See Appendix F Choosing a Handshake.

If the System becomes Paused, it simply stops extracting characters from the input buffer. Eventually the buffer fills up and, if all goes well, the handshake prevents the host from sending more characters. When the Paused state is cleared, the System resumes processing the characters. As soon as the buffer becomes empty enough, the handshake allows the host to resume transmission.

Escape Sequences

Escape sequences are used to establish communications parameters. They are processed immediately; they are not placed in the input buffer and are not inhibited by the Paused state. (For an exception to this rule, see the ES command in Chapter 3.) See Chapter 6 Escape Sequences.

The names of the escape sequences are here represented as ESC.X which stands for three ASCII characters: Escape (character code 27), followed by Period (character code 46), followed by a specified third character.

The host application software and driver software must coordinate their use of escape sequences to provide for smooth and reliable system startup, to properly respond to error conditions, and to avoid output conflicts. An output conflict occurs when an escape sequence requests output before the last character of a previous output request has been transmitted. (The previous request could be either an ACL command or an escape sequence.) See the ESC.E command in Chapter 6.

Hardwired DTR Handshake

The Hardwired DTR (Data Terminal Ready) handshake works as follows. The host establishes a Block Size; characters are always sent to the Automove System in groups smaller or equal to the Block Size. If the Automove System has room for a block of characters in the buffer, it sets the Data Terminal Ready, CD, line to the High state ("asserted"). When there is insufficient room for a block, the System sets Data Terminal Ready to the Low state ("deasserted"). The DTR signal is electrically connected to the host; the host should not transmit any characters (or should not begin a new block of characters) while the DTR line is Low.

The Hardwired DTR handshake is enabled at power up or by using the ESC.@ command. The Block Size is established via the ESC.H or ESC.I command; it defaults to 80, which is long enough for most typical PRINT statements. Note that the Hardwired DTR handshake can be active simultaneously with the Xon/Xoff or Enq/Ack handshake.

Xon/Xoff Handshake

The Xon/Xoff handshake works as follows. As the host sends ACL commands to the System, they are stored in the input buffer and are processed by the System as fast as possible. If the host sends them faster than they can be processed, the buffer begins to fill up. If the number of empty bytes in the buffer becomes less than a certain value, called the Xoff threshold, the System sends an Xoff Trigger String to the host, in effect saying "stop talking". The System continues to process characters; when the number of characters in the buffer falls to a certain value, called the Xon threshold, the System sends an Xon Trigger String to the host, in effect saying "you may begin talking again". This process continues until all data have been transmitted.

The Xoff Threshold is established via the ESC.I command. The Xon Threshold is automatically set to 128 characters (half the buffer size); if the Xoff Threshold is set greater than or equal to 128, the Xon Threshold is automatically decreased so that the sum of the two is equal to the buffer size minus one. The Xon Trigger String is established via the ESC.H or ESC.I command. The Xoff Trigger String is established via the ESC.N command. The Xon/Xoff handshake itself is enabled by executing an ESC.H or ESC.I which specifies an Xon Trigger String but no Enquiry Character, plus an ESC.N which specifies an Xoff Trigger String.

Example: the sequence ESC.I;;17:ESC.N;19:ESC.M10: establishes a "normal" XON/XOFF handshake (using the ASCII DC1 and DC3 characters) with a 10 millisecond turnaround delay.

Enq/Ack Handshake

The Enq/Ack ("Enquire/Acknowledge") handshake works as follows. As with the Hardwired DTR handshake, the host establishes a Block Size. The host sends an Enquiry Character to the Automove System, effectively asking "do you have room for a block of characters?" If the buffer has room, the System immediately sends back an Acknowledge String, effectively saying "yes". Otherwise the System waits until the buffer has room for a block, then sends the Acknowledge String. After it has received the Acknowledge String the host transmits a group of characters whose size is less than or equal to the established Block Size. This sequence repeats until all data have been transmitted.

If necessary, the host can set up an Immediate Response String which the System is to send to the host immediately when an Enquiry Character is received. The Acknowledge String is sent following the Immediate Response String, as soon as there is room for a block.

The Block Size is established via the ESC.H or ESC.I command. The Enquiry and Acknowledge Characters are established via the ESC.H or ESC.I command. The Immediate Response String is established via ESC.N. The Enq/Ack handshake itself is enabled by setting up an Enquiry Character with ESC.H or ESC.I. Other communications parameters which may affect the Enq/Ack handshake are Turnaround Delay (ESC.M), Output Trigger Character (ESC.M), Echo Terminate Character (ESC.M), Output Initiator Character (ESC.M), Output Terminator (ESC.M), and Intercharacter Delay (ESC.N).

The Enq/Ack handshake is mutually exclusive with the Xon/Xoff handshake; only one at a time can be active.

Example: the sequence ESC.I;5;6:ESC.M10: sets up a "normal" ENQ/ACK handshake using the ASCII characters ENQ and ACK, with a 10 millisecond turnaround delay.

Software Checking Handshake

The Software Checking handshake works as follows. The host repeatedly sends the ESC.B command to the Automove System, effectively asking "how much room do you have in your buffer?" The System sends back a number giving the available buffer space. When the number is at least equal to the size of the block to be transmitted, the host sends the block of data. This sequence repeats until all data have been transmitted.

Because of the great number of characters transmitted in each direction, the Software Checking handshake significantly slows down system throughput. It should only be used in cases where none of the other three handshakes can be made to work. See Appendix F Choosing A Handshake.

The host need take no specific action to begin using the Software Checking handshake. Other communications parameters which may need to be changed for the handshake to work successfully are: Turnaround Delay (ESC.M), Output Trigger Character (ESC.M), Echo Terminate Character (ESC.M), Output Initiator Character (ESC.M), Output Terminator (ESC.M), and Intercharacter Delay (ESC.N).

Dummy ACK

In order to facilitate startup of the handshake in hosts which must use the Enq/Ack handshake, the Automove System supports a Dummy ACK response. If no Enq/Ack handshake is active, or if the <Enquiry Character> is not an ASCII ACK (character code 6), the Dummy ACK is enabled. In this case, whenever the Automove System receives the ASCII ENQ character (character code 5), it immediately transmits the ASCII ACK character (character code 6). This is not a handshake and will not prevent buffer overflow.

For more information about the different handshake types, please see Appendix F Choosing a Handshake.

6 Escape Sequences

The Automove System recognizes a number of escape sequences to establish communications parameters and to request that various information be sent back to the host. The escape sequences do not go into the input buffer; instead, they are acted on immediately as they are received.

Each sequence consists of at least three ASCII characters: Escape (ESC, character code 27 decimal), period (character code 46 decimal), and one of the characters @, !,), (, B, E, H, I, J, K, L, M, N, O, R, S, Y, or Z. These letters must be uppercase. In addition, some sequences allow one or more numeric parameters, separated by semicolons ";" and terminated by a colon ":". If no digit characters precede a semicolon or colon, the corresponding parameter receives a default value.

No semicolon precedes the first parameter. When a colon is received, any parameters which have not been specified are defaulted. If a particular escape sequence does not allow parameters, no colon is allowed after the sequence. Conversely, if a sequence allows any parameters, even if they are all to be defaulted, a colon must be sent.

If a parameter is to be defaulted but a following parameter is to be specified, a semicolon must be sent as a "placeholder" for the defaulted parameter.

There are three types of parameters:

A DEC parameter can be in the range 0 through 65535; larger numbers are interpreted as 65535. Some DEC parameters have other restrictions, which are explicitly noted below.

An ASC parameter is a numeric representation of a single ASCII character; it can be in the range 0 through 127. (For example, the value 13 represents the Carriage Return character and 88 represents the letter "X".) Numbers greater than 127 cause an error to be logged.

A STR parameter represents a string of ASCII characters; it may consist of up to 10 ASC values separated by semicolons. A colon, a defaulted value, or a value of zero anywhere in a STR parameter terminates the string; the zero does not become part of the string, and subsequent values are ignored. The STR parameter is set to the null string (i.e. no string) if no ASC parameters precede the colon, or if the first ASC value is zero or is defaulted.

No ASCII spaces, control characters, or other extraneous characters are allowed within the escape sequence; the only allowable characters are the Escape itself, the period, the digits 0 through 9, semicolon, and colon.

When an escape sequence (or an OS, OE, etc.) requests output from the Automove System, the System sends the requested information in ASCII, then sends an Output Terminator sequence. The Output Terminator defaults at power up to be a Carriage Return followed by a Linefeed. It can be changed via the ESC.M command to be any sequence of zero, one, or two characters. (Zero would not be very useful, since the host would not know when to terminate the last numeric value.)

Normally, escape sequences cannot be stored as part of a Download Sequence. However, for certain special system setup needs, it may occasionally be necessary to execute an escape sequence from a Download Sequence; for example, as part of an Autostart sequence. The ES ("Execute Escape Sequence") ACL command can be used in this case. See Chapter 3.

In the following descriptions, square brackets, "[" and "]", enclose groups of optional parameters (i.e. those which may be omitted so as to receive default values). Parentheses, "(" and ")", enclose individual optional parameters. Angle brackets, "<" and ">", enclose the name of each parameter. ESC.X stands for three ASCII characters: Escape (character code 27), followed by Period (character code 46), followed by a specified third character.

As explained above, DEC indicates a decimal numeric parameter; ASC indicates a numeric parameter whose value represents a single ASCII character; and STR indicates a string of up to 10 ASC parameters separated by semicolons.

ESC.(Programmed On

The Programmed On command puts the Automove System into the Programmed On state; in this state, the machine pays attention to the incoming data from the RS-232C channel. This state is the power up default.

This command has the same effect as ESC.Y.

ESC.) Programmed Off

The Programmed Off command puts the System into the Programmed Off state; in this state, the machine ignores all incoming data (including other escape sequences) until an ESC.(or ESC.Y occurs.

This command has the same effect as ESC.Z.

ESC.! [<Reset Code>] : Change System State

The Automove System state is altered according to the value of the DEC <Reset Code> parameter:

Table 19 - ESC.! [<Reset Code>] Parameters

0	The System immediately enters the power up state. This is identical in all software respects to cycling the power switch to the OFF position and back to ON, or pressing the RESET button. All motor power is off. Digital Outputs are all False. The contents of the input buffer are lost. Communications parameters are set to their default values. The X-Y and Z position references are lost. Arrow motion is unconstrained. If there is an Autostart sequence, it starts executing. (See personality parameter 39 in Chapter 7.) The host must wait approximately two seconds before sending the next ACL command or escape sequence, to give the Automove System time to reinitialize itself.
1	The System immediately enters the Emergency Stopped state, halts all motor motion, and turns on the STOP light. This is as though the STOP button had been pressed, except that no "?" is transmitted to the host. See the STOP button in Chapter 4. If the "ESC.!1:" command is executed while the System is already in the Emergency Stopped state the command has no effect. The FP command (see Chapter 3) does not disable this command even when the STOP button is disabled.

2	The System clears the Emergency Stopped state and turns off the STOP light, thus enabling subsequent motor motion. See the STOP button in Chapter 4. The optional Step Verify circuit is reset. The "ESC.!2:" command is ignored if it is executed while the System is in neither the Emergency Stopped state nor the Motor Slipped state. It is also ignored if it is executed while the Stop switch is still actuated. The FP command may be useful here; see the example under CS. For more information about Step Verify see the CS command in Chapter 3 or personality parameters 49, 50, and 51 in Chapter 7. The ACL "CS" command has the same effect as "ESC.!2:" except that CS is not executed until all preceding ACL commands in the input buffer have been processed. Note that all motion commands are ignored while the System is Stopped. If the input buffer contains any such commands, they will be quickly processed. However, to remove any uncertainty, the ESC.K command can be used before "ESC.!2:" to clear out any ACL commands remaining in the buffer. If the host does not either send ESC.K or wait (via ESC.B, ESC.L, or ESC.O) for the buffer to empty, the remaining commands will continue to execute after the "ESC.!2:" is received. This is undesirable because the Home reference is lost when an Emergency Stop occurs.
3	The System immediately enters the Paused state as though the PAUSE button had been pressed. The currently- executing vector or arc, if any, is allowed to finish. See the PAUSE button in Chapter 4. If the "ESC.!3:" command is executed while the System is already in the Paused state, the command has no effect. The ACL "PS" command has the same effect as "ESC.!3:" except that PS is not executed until all preceding ACL commands in the input buffer have been processed. Note that "ESC.!3:" may not execute immediately. The host should verify with ESC.O before assuming that the "ESC.!3:" has taken effect. The FP command (see Chapter 3) does not disable "ESC.!3:" even when the PAUSE button is disabled.
4	The System clears the Paused state and turns off the Paused light, thus enabling subsequent ACL processing. This is as though the PAUSE button had been pressed a second time. If "ESC.!4:" is received while the System is not in the Paused state, the command has no effect. There is no ACL command to perform the function of "ESC.!4:", because such a command would itself be inhibited by the Paused state. Note that "ESC.!4:" may not execute immediately. The host should verify with ESC.O before assuming that the "ESC.!4:" has taken effect.
5	The System aborts the downloading or execution of any Download Sequence at the end of the current command and resumes processing ACL commands from the input buffer. If no sequence is being downloaded or executed when this command is received then the only effect is to reestablish non-volatile memory internal consistency; see ESC.O and Appendix A. See also the AD command in Chapter 3.

6	The System initializes the personality parameters in the non-volatile memory to their factory-default values. The values are those specified as the factory defaults in Chapter 7, except for the following: personality parameter 5 is always set to 0; parameters 19 and 20 are always set to 1.0000; and parameters 25 and 26 are always set to 32767. After the System rewrites the personality parameters it performs the equivalent of an IN command to invoke the new values of the parameters. Note:At the factory, numerous personality parameters are set to custom values according to the configuration of your Automove System. You should have received a floppy disk with an Asymtek software program which re-establishes your custom values. If you are unable to run this program, be sure to record all personality parameter values (via the OP command) before you perform the "ESC.!6:" command, so that you can reinstate the custom values.
7	The System clears the entire download portion of the non-volatile memory. This command may be necessary if a portion of the memory has become corrupt or inaccessible due to a reset or power failure during downloading. See the BD command in Chapter 3, the ESC.S command in this chapter, and Appendix A Power Up Actions. Do not turn off the power or press RESET while the memory is being erased.
8	The System initializes the XY Linearity Correction Table to its factory default value. All X and Y corrections are set to zero and the X and Y grid spacings are set to zero, which disables the linearity correction mechanism. See Chapter 9 Linearity Correction. Note:At the factory, the Linearity Correction Table is set according to the calibration procedure for your X,Y table. You should have received a floppy disk with an Asymtek software program which reestablishes your custom linearity corrections. If you are unable to run this program, be sure to record all table values (via the OR command) before you perform the "ESC.!8:" command, so that you can reinstate the custom values.
9	The System sets the values of all ACL variables to 0.0. Note that the variables are stored in non-volatile memory and are not automatically zeroed at power up or at any other time.
35	This escape sequence combines the action of ESC.!5,6,7,8 & 9 into an easier-to-type command.

If the <Reset Code> parameter is omitted it defaults to 0, meaning that the System enters the power up state. If a value other than 0 through 9 is specified, the ESC.! command has no effect.

ESC.@ [(<Logical Buffer Size>) ; (<DTR Mode>)] :

Set Communications Configuration

The DEC <Logical Buffer Size> parameter establishes an effective buffer size for purposes of handshaking. Values above 256 are interpreted as 256. The physical buffer remains 256 bytes, and no characters will be lost if the logical buffer size is exceeded but the physical buffer size is not. Handshakes will occur as though the buffer size were <Logical Buffer Size>. The parameter defaults to 256.

The DEC <DTR Mode> parameter enables or disables the Hardwired DTR handshake; if bit 0 of the parameter is True, the Hardwired DTR handshake is enabled; otherwise it is disabled. The parameter defaults to 1. While this handshake is enabled, the Data Terminal Ready line (pin 20 of the RS-232 connector) goes True whenever the buffer has room for <Block Size> characters, and false otherwise. If the Hardwired DTR handshake is disabled, the Data Terminal Ready line remains True at all times. <Block Size> is specified via the ESC.H or ESC.I command.

ESC.B Output Buffer Space

This command causes the Automove System to output the amount of space currently available in the logical buffer. The response is an integer in the range 0 through 256, corresponding to the number of bytes of buffer space available for ACL commands.

ESC.E Output Extended Error

This command causes the System to output the current value of the Communications Error Code and then set the Communications Error Code to zero. The response is an integer with the following meaning:

Table 20 - ESC.E Response Definitions

0	No communications error has occurred.
10	An output command was received while another output command was executing. The original command will finish normally; the second one will be ignored. See the table on the next page.
11	An invalid character was received after the first two bytes, ESC and period, of an escape sequence.
12	An invalid character occurred in the parameter list of an escape sequence. The parameter containing the invalid character and all subsequent parameters will be defaulted.
13	An ASC parameter was outside the range 0 through 255. The parameter is defaulted and all subsequent parameters are ignored.
14	Too many parameters were received. Excess parameters are ignored until a colon or illegal character is received, or until another escape sequence is begun.
15	A framing error, parity error, or overrun error has been detected in incoming data.
16	The input buffer has overflowed. As a result, one or more bytes of ACL data have been lost, and an ACL error will probably occur.

Note that the error code reflects only the first communications error that occurs; subsequent communications errors will be logged only if an intervening ESC.E clears the error code.

Whenever any communications error is logged, a "?" character is sent immediately to the host. See "Exceptional Conditions" in Chapter 5.

Table 21 - In Case of Communications Error Code Conflicts

	If OA, OS, etc. is pending	If handshake string is pending	If output from ESC is pending
OA, OS, etc.	can't happen *	error	error
ESC.B	error **	error	error
ESC.E	error	error	error
ESC.L	error	error	error
ESC.O	error	error	error
Ack	error	error	error
Immed. Response	error	error	error
Xon	can't happen	can't happen	error
Xoff	enqueue ***	can't happen	enqueue
Dummy ACK	enqueue	enqueue	enqueue
"?"	enqueue ****	enqueue	enqueue

* "Can't happen" means that the normal order in which the Automove System attends to its duties prevents this situation from occurring.

** "Error" means that Communications Error 10 is logged.

*** "Enqueue" means that the new request will be "remembered" and processed as soon as the currently-active one is finished.

**** The "?" can be transmitted to the host at virtually any time, for example when the STOP button is pressed. The "?" never results in an output conflict. For more information, see "Exceptional Conditions" in Chapter 5.

ESC.H [(<Block Size or Xoff Threshold>) ; (<Enquiry Character>) ;
(<Acknowledge String or Xon Trigger String>)] :

Set Handshake Mode 1

This command sets the same parameters as the ESC.I command, but enables Handshake Mode 1. In Handshake Mode 1, certain parameters of the ESC.M and ESC.N commands will affect the response to the Enquiry Character or the Xon Trigger String, according to the Applicable Output Modifiers tables on page 6-10.

ESC.I [(<Block Size or Xoff Threshold>) ; (<Enquiry Character>) ;
(<Acknowledge String or Xon Trigger String>)]:

Set Handshake Mode 2

This command sets the same parameters as the ESC.H command, but enables Handshake Mode 2. In Handshake Mode 2, the response to the Enquiry Character or the Xon Trigger String is affected by the Turnaround Delay, but not by the other parameters of the ESC.M and ESC.N commands. See the Applicable Output Modifiers table on page 6-11.

If the second parameter is nonzero, the Enq/Ack handshake is enabled and the parameters are interpreted as follows: the first parameter is <Block Size>, the second is <Enquiry Character>, and the third is <Acknowledge String>. If the second parameter is zero, the Enq/Ack handshake is disabled and the parameters are interpreted as follows: the first parameter is <Xoff Threshold>, the second parameter is zero, and the third parameter is <Xon Trigger String>.

Enq/Ack Handshake:

The DEC <Block Size> parameter becomes the block size used in Enq/Ack and Hardwired DTR handshakes. Values greater than 256 are interpreted as 256. Default is 80.

The ASC <Enquiry Character> parameter, if nonzero, enables the Enq/Ack handshake and specifies the character which will be recognized by the Automove System as the Enquiry Character. The default is zero. The most common value to use is 5, corresponding to the ASCII ENQ character.

The STR <Acknowledge String> parameter becomes the Acknowledge String which is transmitted after the <Enquiry Character> is received, when the buffer has room for a block of characters. The default is the null string. If <Acknowledge String> is null, no characters will be transmitted when the buffer has room. The most common value to use is a 6, which establishes the ASCII ACK character as the Acknowledge String.

Example: the sequence ESC.I;5;6:ESC.M10: sets up a "normal" ENQ/ACK handshake using the ASCII characters ENQ and ACK, with a 10 millisecond turnaround delay.

Xon/Xoff Handshake:

The DEC <Xoff Threshold> parameter specifies the number of empty bytes remaining in the buffer when the <Xoff String> is to be transmitted. The default is 80. The number should be large enough to allow for any latency within the host, during which the host continues to send characters before recognizing that the Xoff String has been transmitted. If the Xoff Threshold is set greater than or equal to 128, the Xon Threshold is automatically decreased so that the sum of the two is equal to the buffer size minus one.

If the second parameter is not defaulted or zero, the Xon/Xoff handshake is not enabled. (Xon/Xoff is also disabled if the <Xon Trigger String> parameter is null, or if no Xoff Trigger String has been established via an ESC.N command.)

The STR <Xon Trigger String> parameter sets up the string which is to be transmitted when the number of characters in the buffer falls to the Xon Threshold. Default is the null string. If this parameter is the null string, the Xon/Xoff handshake is disabled.

Example: the string ESC.I100;;17: establishes an Xoff Threshold of 100 characters, such that the ASCII DC1 character (character code 17, also known as XON) is sent when 100 empty bytes remain in the buffer. The Xoff Trigger String must be established via an ESC.N command.

Example: the sequence ESC.I;;17:ESC.N;19:ESC.M10: establishes a "normal" XON/XOFF handshake (using the ASCII DC1 and DC3 characters) with a 10 millisecond turnaround delay.

APPLICABLE OUTPUT MODIFIERS

Table 22 - Applicable Output Modifiers in Handshake Mode 1

Type of Output	Output Trigger	Turn-around Delay	Echo Terminator	Output Initiator	Output Terminator
OA, OS, etc.	yes	yes	yes	yes	yes
ESC.B	yes	yes	yes	yes	yes
ESC.E	yes	yes	yes	yes	yes
ESC.L	yes	yes	yes	yes	yes
ESC.O	yes	yes	yes	yes	yes
ESC.S	yes	yes	yes	yes	yes
Acknowledge	yes	yes	yes	no	yes
Immed. Response	yes	yes	yes	no	yes
Xon	no	no	yes	no	yes
Xoff	no	no	no	no	no
Dummy ACK	no	yes	no	no	no
"?" message	no	no	no	no	no

APPLICABLE OUTPUT MODIFIERS

Table 23 - Applicable Output Modifiers in Handshake Mode 2

Type of Output	Output Trigger	Turn-around Delay	Echo Terminator	Output Initiator	Output Terminator
OA, OS, etc.	yes	yes	yes	yes	yes
ESC.B	yes	yes	yes	yes	yes
ESC.E	yes	yes	yes	yes	yes
ESC.L	yes	yes	yes	yes	yes
ESC.O	yes	yes	yes	yes	yes
ESC.S	yes	yes	yes	yes	yes
Acknowledge	no	yes	no	no	no
Immed. Response	no	yes	no	no	no
Xon	no	no	no	no	no
Xoff	no	no	no	no	no
Dummy ACK	no	yes	no	no	no
"?" message	no	no	no	no	no

Note: The Immediate Response String and Acknowledge String share between them one output trigger, one turnaround delay, one echo terminator, and one output terminator.

ESC.J Abort Output Request

This command aborts all pending or partially transmitted output, from either ACL commands or escape sequences. This includes the output data itself, handshake strings, turnaround delays, echo suppression, output triggers, and the "?" message.

ESC.K Abort ACL Command

This command aborts any partially decoded ACL command and discards commands in the buffer. Except for OT, WA, and WN, any partially executed ACL command is allowed to finish. If an OT command is waiting for the TEACH button to be pressed, the command is aborted and sends no output to the host. Any WA or WN wait is terminated immediately.

If the System is paused and waiting to finish a previously-received ACL command, ESC.K does not abort the paused command. In order to resume ACL processing the user must press the PAUSE button again or send the ESC.!4: command. There is no way to prevent the paused command from being executed, other than to "power up" the machine, either by resetting it, cycling the power switch, or sending the ESC.!0: command. You can, however, prevent any motion from occurring: send ESC.!1: to put the machine in the Stopped state. This automatically clears the Paused state, so the paused command executes but does not produce any motion.

If a BC command has been executed but the Continuous Path motion has not yet begun, the entire Continuous Path sequence is ignored. If the motion has already started it is allowed to finish.

ESC.L Output Buffer Size

The Automove System waits until the buffer is empty, then transmits the current Logical Buffer Size as established at power up or via the ESC.@ command. The value sent will be an integer in the range 1 through 256.

ESC.M [(<Turnaround Delay>) ; (<Output Trigger Character>) ; (<Echo Terminate Character>) ; (<Output Terminator 1>) ; (<Output Terminator 2>) ; (<Output Initiator Character>)] : Set Output Mode

The DEC <Turnaround Delay> parameter establishes a delay, in milliseconds, to be executed following each output request, before beginning to transmit the first character back to the host. The default is zero.

The ASC <Output Trigger Character> parameter specifies an Output Trigger Character. When the host requests output (via either an ACL command or an escape sequence), the Automove System does not begin transmitting until the host sends the Output Trigger Character; the System then executes the Turnaround Delay, if any, and begins transmission. All other incoming characters between the output request and the Output Trigger Character are ignored. The parameter defaults to zero, or no Output Trigger Character active. See the table under ESC.I.

The ASC <Echo Terminate Character> parameter establishes echo suppression. While the Automove System is transmitting characters to the host, it ignores any incoming characters until the Echo Terminate Character is seen. The parameter defaults to zero, or no Echo Terminate Character active. See the table under ESC.I.

The ASC <Output Terminator 1> parameter establishes the first character of the Output Terminator sequence. This sequence of up to 2 characters is transmitted following any output requested by the host; see the table under ESC.I. This parameter defaults to 13, or Carriage Return. If it is specified to be zero, no Output Terminator sequence is sent.

The ASC <Output Terminator 2> parameter establishes the second character of the Output Terminator sequence. If <Output Terminator 1> was zero, <Output Terminator 2> is ignored. If <Output Terminator 2> is omitted or is zero, only <Output Terminator 1>, if any, is transmitted. If neither <Output Terminator 1> nor <Output Terminator 2> is specified, <Output Terminator 2> defaults to zero. In other words, if neither is specified, the sequence defaults to CR-LF. If only <Output Terminator 1> is specified, the sequence is only the one character.

The ASC <Output Initiator Character> parameter establishes a character to be sent after the turnaround delay, if any, but before the first character of any output is transmitted. The default is zero, or no Output Initiator Character active. See the table under ESC.I.

Example: ESC.M100;63;;10: establishes a 100 millisecond turnaround delay, an Output Trigger Character of "?" (character code 63), and a one-character Output Terminator sequence consisting of a Linefeed (character code 10). This command would be useful if the host computer were running a BASIC interpreter which sends a "?" as the result of an INPUT statement, but is not ready to read the data for a few milliseconds after sending the "?". The Linefeed Output Terminator sequence ensures that the BASIC interpreter will properly terminate its INPUT statement. (Perhaps this particular BASIC interpreter does not like to see a Carriage Return preceding the Linefeed.) An intercharacter delay (see ESC.N) might also prove necessary.

ESC.N [(<Intercharacter Delay>) ;
(<Immediate Response String or Xoff Trigger String>)] :
Set Extended Output and Handshake Mode

The DEC <Intercharacter Delay> parameter specifies a delay, in milliseconds, which is to be executed before the transmission of each character to the host. The delay begins after the transmission of the previous character is complete. The Intercharacter Delay is executed for the first character even if a Turnaround Delay is also active; so the net delay is the sum of the two. The Intercharacter Delay is also executed for the Output Initiator Character and the Output Terminator sequence; it is not executed for the first character of an unenqueued Xoff Trigger String.

If an Enq/Ack handshake has been established via an ESC.H or ESC.I command, the second parameter is interpreted as <Immediate Response String>. Otherwise, it is interpreted as <Xoff Trigger String>.

The ASC <Immediate Response String> parameter becomes the string which is transmitted immediately when the Enquiry Character is received. (This is not the same as the Dummy ACK described at the beginning of this chapter.) The parameter defaults to zero, or no Immediate Response String active.

The ASC <Xoff Trigger String> parameter becomes the string which is transmitted when the buffer space decreases below the Xoff Threshold while an Xon/Xoff handshake is enabled. The <Xoff Trigger String> parameter must be specified in order to enable the Xon/Xoff handshake; also, an Xoff Threshold and Xon Trigger String must be specified via an ESC.H or ESC.I.

Example: the command ESC.N;19: establishes an Xoff Trigger String consisting of the ASCII DC3 character (character code 19, also known as XOFF). This character is sent when the buffer is nearly full, at a threshold established by the ESC.H or ESC.I command. There is no intercharacter delay, since the first parameter was defaulted to zero by the semicolon.

Example: the command ESC.N10;19: establishes an Intercharacter Delay of approximately 10 milliseconds, or 0.01 second. This might prove useful if the host computer is unable to accept Automove transmissions at whatever speed happens to result from the current baud rate setting. Many small-computer BASIC interpreters fall into this category. The second parameter, 19, sets up the same DC3 Xoff Trigger String as in the above example.

ESC.O Output Extended Status

This command causes the Automove System to output an Extended Status code. This will be a decimal value whose bits have the following meanings:

Table 24 - Extended Status Code Definitions

Bit	Dec. Value	Meaning if 1
0	1	The machine is executing a Download Sequence.
1	2	Not used, value undefined.
2	4	The STOP button or external Emergency Stop switch is currently actuated. The CS and "ESC.!2:" commands are ignored while this bit is set, unless the FP command disables STOP. This bit is cleared when the Stop switch is released. (See also bit 6, below).
3	8	The input buffer is empty.
4	16	The machine is in the Paused state.
5	32	The machine is in the Motor Slipped state; i.e., the optional Step Verify circuit has detected a motor slip. This bit is cleared by the CS command or by "ESC.!2:".
6	64	The machine is in the Emergency Stopped state. This bit is cleared by the CS command or by "ESC.!2:".
7	128	The non-volatile memory has failed its internal consistency check. This bit may be set after reset or power up. It is cleared by "ESC.!5:", "ESC.!6:", "ESC.!7:", or "ESC.!8:". See Appendix A.

Executing this command is the same as executing the following commands without parameters: ESC.@, ESC.H, ESC.I, ESC.M, and ESC.N. It has no effect on preceding ACL commands; the input buffer is NOT cleared. (See ESC.K.)

The following parameters are established by ESC.R :

Hardwired DTR handshake enabled.
Block Size = 80.
System is in the Programmed On state.
Logical buffer size = 256.
Enquiry Character = 0 (no Enq/Ack handshake enabled).
Acknowledge String = 0 (no Acknowledge String).
Xon Trigger String = 0 (no Xon Trigger String).
Xoff Trigger String = 0 (no Xoff Trigger String).
Turnaround delay = 0.
Output Trigger Character = 0 (no Output Trigger Character).
Echo Terminate Character = 0 (no Echo Terminate Character).
Output Terminator = 13;10 (Carriage Return, Linefeed).
Output Initiator = 0 (no output initiator).
Intercharacter Delay = 0.
Immediate Response String = 0 (no Immediate Response String).
Xon Threshold = 128.

In addition, the communications error code is cleared.

See Appendix A Power Up Actions.

ESC.S [(<selector>) [; (<id>)]] : Output Resource Allocation

This command causes the Automove System to output the current allocation or consumption of some resource, according to the value of the DEC parameter <selector>:

Table 25 - <selector> Parameter Values for Command ESC.S

<selector>	Action
1	Output the overall size of the non-volatile download memory, in bytes. This will normally be 13894. For release 3.5 or later, the allowable download memory is 32986 bytes.
2	Output the number of bytes currently unused in the download memory. The value is in the range 0 through 13894. For release 3.5 or later, the range may be 0 through 32986.
3	Output the identifying number, 0 to 255, of the currently executing Download Sequence, or output -1, if no sequence is executing.
4	Output the current level of pattern nesting. If no pattern is active, zero is output. Otherwise an integer in the range 1 through 12 is output.
5	Output the actual number of memory bytes occupied by the Download Sequence whose identifying number is given by <id>, or zero if that Download Sequence is not defined.
6	Output the number of physical action commands which have been processed during the current Continuous Path sequence, or -1 if no BC command has been executed. The value output is 0 if only a BC has been executed; it is in the range 1 through 200 as the commands are processed; and it returns to -1 as soon as the EC command executes and the motion begins.
7	Output the maximum number of physical action commands allowable in a BC...EC Continuous Path sequence. This will normally be 200.

If <selector> is omitted or is not one of the above values, a zero is output.

The DEC <id> parameter specifies which Download Sequence is meant if <selector> is 5. <id> is ignored if <selector> is not 5. If <selector> is 5 but <id> is not in the range 0 through 255 then the System outputs the value zero.

ESC.Y Programmed On

The Programmed On command puts the Automove System into the Programmed On state; in this state, the machine pays attention to the incoming data from the RS-232C channel. This state is the power up default.

ESC.Vn: Retrieve Variable Value

This escape sequence allows the value of a variable to be output the same way as an OV command but can be sent while a machine is performing a downloaded program that is repeating forever, such as a conveyor controller routine. This allows a "state" variable to be retrieved to identify what the conveyor program is doing.

ESC.Wn;v: Write Variable Value

This escape sequence allows a value to be written to a variable while the machine is performing a downloaded program that is repeating forever, such as a conveyor controller routine. This allows the downloaded program to include a test for a variable value in its program loop. If the value is equal, it can execute another downloaded procedure such as turning on the red error lamp on an A-600. The downloaded program continues to monitor the same variable until it changes back to 0, at which point the error lamp can be turned off. If both tests are included in the same downloaded procedure this can be done without significantly affecting how the downloaded program is running.

ESC.Z Programmed Off

The Programmed Off command puts the System into the Programmed Off state; in this state, the machine ignores all incoming data (including other escape sequences) until an ESC.(or ESC.Y occurs.

7 Changing the Personality

You can change various aspects of the Automove System's "personality" via the PE command. The syntax of the command is:

PE <selector>, <value>

The new value is written to the System's non-volatile memory so that it becomes permanent; i.e., it is retained even when the power is turned off.

Before sending the PE command be sure the backpanel Memory Enable switch is in the Down position.

<selector> An integer which determines which personality parameter is to be changed. It must be in the range 1 through the maximum personality parameter number; see below.

<value> The new value to be assigned; its allowable range depends on which personality parameter is being changed.

The personality parameters are not affected by the IN command. The factory-set values can, with a few exceptions, be reinstated by the ESC.!6: command; see Chapter 6.

If you change several personality parameters from their factory-set values you should write a short computer program to set those parameters. Then if, for some reason, you need to use ESC.!6: to reinstate the factory defaults you can run your program to re- customize the machine.

The current value of any personality parameter can be read via the OP command.

Example: PE 2, 200; sets the Arrow Slow Step Delay to the value 200, which is twice as long as the factory-set value 100. The new value is written to non-volatile memory and thus becomes a "permanent" value.

Some personality parameters take effect immediately. For example, if you send PE 4, 1; to swap the Left and Right Arrow buttons, then immediately press the Left Arrow, the carriage moves to the right.

However, some of the personality parameters are only default values, and thus don't take effect until the next time the default is applied. These are indicated by the word "default" in the parameter name. For example, consider parameter 19, Default X Calibration Factor. If you execute the command PE 19, 2.0000; the X calibration factor does not change immediately. But if you later turn the power off and back on, or execute CF; or IN; the X calibration factor then defaults to 2.0000.

No checking of values is done by the PE command, other than to give an error if the value is outside the range of the number type. (See "Command Syntax" in Chapter 1.) If you give a value that doesn't make sense for the particular personality parameter, the System may behave in strange ways or completely lock up; or you may get an ACL error at power up or at IN, or when you execute an ACL command with no parameters.

Parameters 27 and 28 are not used by the System. They can be used to record your own configuration or status information in non- volatile memory.

Following is a list of the personality parameters. Each appears with the <selector> value to access that parameter, the name of the parameter, the allowable range, the factory-set value, and a description of the parameter's function.

There is a summary of the personality parameters in Appendix D.

The Personality Parameters

Arrow Buttons

1: Arrow Single Step Delay, range 0 to 65535, factory 3000

The Arrow buttons (see Chapter 4) have three modes of operation. If you press and immediately release a button, the motor takes a single microstep. If you hold the button for a while, the motor begins to turn slowly. If you press FAST with the button, the motor moves rapidly. This personality parameter sets the amount of delay after the single step, before the motor begins to move slowly. The parameter is multiplied by approximately 100 microseconds; thus the factory-set value 3000 corresponds to approximately 0.3 seconds of delay.

2: Arrow Slow Step Delay, range 0 to 65535, factory 100

This parameter sets the delay between slow Arrow button steps (i.e. the speed of travel); see Arrow Single Step Delay, above. The parameter is multiplied by approximately 100 microseconds.

3: Arrow Fast Step Delay, range 0 to 65535, factory 3

This parameter sets the delay between fast Arrow button steps (i.e. the speed of travel); see Arrow Single Step Delay, above. The parameter is multiplied by approximately 100 microseconds.

4: X Arrow Direction Swap, range 0 to 255, factory 0

If this parameter is zero the Left Arrow button moves the carriage toward lower-numbered X coordinates and the Right Arrow, toward higher. If the parameter is nonzero the functions of the Left and Right Arrows are swapped. See personality parameter 29.

5: Y Arrow Direction Swap, range 0 to 255, factory: see text

If this parameter is zero the Down Arrow button moves the carriage toward lower-numbered Y coordinates and the Up Arrow, toward higher. If the parameter is nonzero the functions of the Up and Down Arrows are swapped. The factory setting is zero for normal Automove X-Y tables and controllers, and is 1 for Automove X-Y tables configured for upside-down mounting. See parameter 30.

Find Home and Manual Re-Reference

6: First Find Home X Back Off Distance, range -32768 to 32767, factory 250

The FH ("Find Home Switches") command and the manual re-reference operation perform the same sequence of actions twice: once quickly, the second time more slowly, for a more accurate measurement of the switch position. Each sequence consists of a diagonal vector away from both X and Y switches (to be sure the switches are open), then a move toward the Y switch until it closes, then a move toward the X switch until it closes. This personality parameter sets the X distance, in microsteps, of the initial diagonal move for the first sequence.

7: First Find Home Y Back Off Distance, range -32768 to 32767, factory 250

This parameter sets the Y back off distance for the first Find Home move, in microsteps; see parameter 6, above.

8: First Find Home X Seek Step Delay, range 0 to 65535, factory 2

This sets the delay between steps when seeking the X limit switch the first time. The value is multiplied by approximately 100 microseconds. See parameter 6.

9: First Find Home Y Seek Step Delay, range 0 to 65535, factory 2

This sets the delay between steps when seeking the Y limit switch the first time. The value is multiplied by approximately 100 microseconds. See parameter 6.

10: Second Find Home X Back Off Distance, range -32768 to 32767, factory 100

This is like parameter 6, but is used in the second Find Home sequence rather than the first.

11: Second Find Home Y Back Off Distance, range -32768 to 32767, factory 100

This is like parameter 7, but is used in the second Find Home sequence. See parameter 6.

12: Second Find Home X Seek Step Delay, range 0 to 65535, factory 100

This is like parameter 8, but is used in the second Find Home sequence rather than the first. See parameter 6.

13: Second Find Home Y Seek Step Delay, range 0 to 65535, factory 100

This is like parameter 9, but is used in the second Find Home sequence rather than the first. See parameter 6.

14: Find Home Max X Distance, range 0 to 32767, factory 32767

This sets the maximum number of steps the System will take when seeking the X home switch, in both the first and second sequences. If the switch has not closed before this number of steps has been taken, the motor stops moving, an error is logged, and the present position becomes zero. See parameter 6.

If parameter 14 is zero then there is no maximum number of steps; i.e., if the switch never closes the X motor will travel forever.

If you set parameter 14 to 1 and connect a wire in place of the X home switch you can prevent the X axis from moving during the FH.

15: Find Home Max Y Distance, range 0 to 32767, factory 32767

This sets the maximum number of steps the System will take when seeking the Y home switch. If the parameter is zero there is no maximum number of steps. See parameter 14.

16: Find Z Home Max Distance, range 1 to 32767, factory 32767

This sets the maximum number of steps the System will take when seeking the Z home switch, in both the first and second sequences. If the switch has not closed before this number of steps has been taken, the motor stops moving, an error is logged, and the present position becomes zero. If the parameter is zero then there is no maximum number of steps; i.e., if the switch never closes the Z motor will travel forever.

Antibacklash Vectors

17: Antibacklash Vector X Offset, range -32768 to 32767, factory -15

This sets the X offset for antibacklash vectors. See the AB command in Chapter 3. Use caution when increasing this number, since the antibacklash offset position is not constrained to the travel limits.

18: Antibacklash Vector Y Offset, range -32768 to 32767, factory -15

This sets the Y offset for antibacklash vectors. See parameter 17.

Calibration Factors and Resolution

19: Default X Calibration Factor, range 0.0000 to 32767.9999, factory: see text

This parameter sets the default X calibration factor which is applied at power up, at IN, and at a CF command with no parameters.

For X-Y tables shipped with built-in or external controllers, the factory-set value depends on the measured calibration; it will be a number near 1.0000, for example 0.9996 or 1.0003. (This number will also appear on a sticker on the back of the X-Y table.) For bare controllers shipped with no X-Y table the factory-set value is 1.0000. See the CF and OF commands in Chapter 3.

20: Default Y Calibration Factor, range 0.0000 to 32767.9999, factory: see text

This sets the default Y calibration factor. See parameter 19.

21: Default X Microstep Factor, range 1.0000 to 32.0000, factory 8.0000

This parameter sets the default X microstep factor (i.e. resolution) which is applied at power up and an RE command with no parameters (but not at IN!).

22: Default Y Microstep Factor, range 1.0000 to 32.0000, factory 8.0000

This sets the default Y microstep factor; see parameter 21.

Travel Limits

23: Default Travel Limit <xmin>, range 0 to 32767, factory 0

This sets the default <xmin> travel limit value to be applied at power up, at IN, and at a TL command with no parameters. See the TL command in Chapter 3.

24: Default Travel Limit <ymin>, range 0 to 32767, factory 0

This sets the default <ymin> travel limit; see parameter 23.

25: Default Travel Limit <xmax>, range 0 to 32767, factory: see text

This sets the default <xmax> travel limit; see parameter 23. The factory setting is described under the TL command in Chapter 3.

26: Default Travel Limit <ymax>, range 0 to 32767, factory: see text

This sets the default <ymax> travel limit; see parameter 23. The factory setting is described under the TL command in Chapter 3.

User Definable

27: User Definable 1, range -32768 to 32767, factory 0

This parameter is not used by the System. It can be set to any value and can be read back later via the OP command. You could use this, for example, to store your own system status or configuration codes in non-volatile memory.

28: User Definable 2, range -32768 to 32767, factory 0

This is another user-definable parameter. See parameter 27.

Motor Direction

29: X Motor Direction, range 0 to 255, factory 0

If this parameter is nonzero the rotation of the X motor is reversed. It should be set so that X-axis travel toward lower- numbered coordinates produces motion towards the X Home switch.

When separate Automove motors are shipped from the factory they are wired such that negative motion produces a clockwise rotation as viewed from the protruding end of the shaft. If this is inappropriate for your X home switch location then set personality parameter 29 to a nonzero value.

Note that this personality parameter should be set before you pick an appropriate value for personality parameter 4.

30: Y Motor Direction, range 0 to 255, factory 0

This changes the Y motor direction. See parameter 29. Parameter 30 should be set before you pick an appropriate value for parameter 5.

31: Z Motor Direction, range 0 to 255, factory 0

This changes the Z motor direction. It should be set nonzero if the FZ command produces motion away from the Z Home switch. See parameter 29.

XY Acceleration and Step Rate

32: Default Acceleration, range 1 to 65535, factory 193

This sets the default XY acceleration which is applied at power up, at IN, and at an AC with no parameter. See the AC command in Chapter 3.

33: Default Step Rate, range 1 to 65535, factory 10000

This sets the default XY step rate which is applied at power up, IN, and at an SR with no parameter. See the SR command in Chapter 3.

34: Alternate Acceleration, range 1 to 65535, factory 193

This sets the XY acceleration which is used when a VM command sets the No SR bit. A new value in this parameter does not take effect until after the next AC or SR command. In other words, after changing parameter 34 you must send an AC and a VM, or an SR and a VM.

35: Alternate Step Rate, range 1 to 65535, factory 20000

This sets the XY step rate which is used when a VM command sets the No SR bit. A new value in this parameter does not take effect until after the next AC or SR command. In other words, after changing parameter 35 you must send an AC and a VM, or an SR and a VM.

Digital Inputs

36: Digital Inputs Trigger Enable, range 0 to 255, factory: 0

This determines which Digital Inputs are allowed to trigger some action. (Parameter 37 determines which action is triggered.)

If a particular bit of parameter 36 is a 1, any change from False-to-true on the corresponding Digital Input causes the action to be triggered.

Conversely, if a particular bit of parameter 36 is a 0, False-to-true transitions on the corresponding Digital Input are ignored for purposes of triggering. (However, the correct states of the Digital Inputs can still be sensed via the ON and WN commands; see Chapter 3.)

Digital Inputs changes are normally only sensed while the motors are idle. If you need quick response even in the middle of a move, consider using the MN ("Mid-move Digital Inputs Response") command in addition to these personality parameters.

If a Digital Input is sitting True at power up, no function is triggered until the Input goes False, then True again. Similarly, if parameter 36 is changed to enable an Input which is currently sitting True, no function is triggered until the Input goes False, then True again.

For more information, see Chapter 8.

37: Digital Inputs Function Select, range 0 to 2047, factory: 0

False-to-true transitions on the Digital Inputs can automatically trigger some action if they are enabled by parameter 36. Parameter 37 determines which action is triggered; each bit has a specific meaning. Bits 0 through 7 each determine whether the corresponding Digital Input is a Toggle Input or a Download Invoke Input. Bits 8 through 10 change the meaning of the Download Invoke Inputs; see below.

If one of bits 0 through 7 is a 1 (and the corresponding bit in parameter 36 is also a 1) then a false-to-true Input transition toggles the corresponding Digital Output as though the TD command had been executed. (If bit 7 is a 1 then a transition on Digital Input 7 toggles Digital Output 7, etc.) The toggle is followed by a wait as established by the WD command.

Alternatively, if one of bits 0 through 7 is a 0 (and the corresponding bit in parameter 36 is a 1) then the corresponding Digital Input becomes a Download Invoke Input. It can participate in invoking a Download Sequence as though an XD command had been executed or the GO button had been pushed. The Download Invoke Inputs are grouped together and interpreted according to the values of bits 8 and 9, as follows:

Table 26 - Download Invoke Inputs Interpretations

Bit 9	Bit 8	Interpretation of Download Invoke Inputs
0	0	One-for-one invoke. A transition on Input 0 invokes Download Sequence 0, etc.
0	1	Same as above.
1	0	Binary-encoded invoke. See below.
1	1	BCD-encoded invoke. See below.

With the one-for-one invoke you can potentially invoke up to 8 different Download Sequences (numbers 0 through 7) via the Digital Inputs. With the binary-encoded invoke you can invoke up to 128 different sequences (numbers 0 through 127) and with the BCD encoding you can invoke up to 80 sequences (0 through 79).

The binary-encoded invoke works as follows: the most-significant (i.e. highest-numbered) Download Invoke input is a trigger; when it goes from False to True the System invokes a Download Sequence. To decide which Download Sequence, the System looks at the remaining Download Invoke inputs and interprets them as a binary number.

For example, suppose you have activated Inputs 5, 4, 1, and 0; i.e., parameter 36 is 51 (binary 00110011) and parameter 37 is 512 (binary 1000000000 -- only bit 9 is set). Input 5 becomes the trigger. If Inputs 4, 1, and 0 are all False then Download Sequence 0 is invoked. If Input 1 is True then Download Sequence 2 is invoked -- the active Inputs are

False True False, corresponding to the binary number 010, which is equal to decimal 2. If Inputs 4, 1, and 0 are all True (binary 111 = decimal 7) then Download Sequence 7 is invoked.

The BCD ("Binary Coded Decimal") -encoded invoke works the same as the binary-encoded invoke, except that the Download Invoke Inputs are interpreted as a BCD number rather than a binary number. In a BCD number bits 0 through 3 are the ones digit and bits 4 through 6 are the tens digit. So a BCD 1010011 is equal to decimal 53; the tens digit is a 5 (binary 101) and the ones digit is a 3 (binary 0011). Invalid BCD numbers (e.g. 1011100) produce undefined results.

For example, suppose you have activated all 8 Inputs; i.e., parameter 36 is 255 (binary 11111111) and parameter 37 is 768 (binary 1100000000 - only bits 8 and 9 set). Input 7 becomes the trigger. Inputs 0 through 6 are interpreted as a BCD number; to invoke Download Sequence 49 you would set the inputs to 1001001, for example via a pair of BCD-encoded thumbwheel switches set to the decimal number 49.

If you activate only one Digital Input it becomes a trigger for Download Sequence 0.

Bit 10 affects the Download Invoke feature as follows: when bit 10 is set (i.e. 1) the Automove System adds 100 to the i.d. of the invoked Sequence. For example, instead of invoking Download Sequence 49 in the above example, the System would invoke Download Sequence 149. Bit 10 affects all three invoke modes discussed above (one-for-one, binary-encoded, and BCD-encoded) but does not affect front-panel invokes. Thus, by setting bit 10, you can invoke 10 different Download Sequences from the front panel, plus as many as 128 other Sequences (numbered 100 through 227) from the Digital Inputs.

For more information, see Chapter 8.

Example: suppose parameter 36 is set to 255 (thus enabling triggering on all eight Inputs), and parameter 37 is set to 3 (i.e. bits 0 and 1 are ones, bits 2 through 7, 8, and 9 are zeros). Furthermore, assume parameter 38 is set to 255, meaning that all eight Inputs are Low True. A High-to-Low transition on Input 0 or Input 1 causes Digital Output 0 or 1 to toggle, respectively. A High-to- Low transition on Inputs 2 through 7 invokes Download Sequence 2 through 7, respectively.

38: Digital Input Sense, range 0 to 255, factory 255

This determines the logic sense of the Digital Inputs. If a given bit of parameter 38 is a 1 then the corresponding Digital Input is Low True (i.e. Low = True, High = False). If the bit is a 0 then the Input is High True.

This logic sense applies to all modes of using the Digital Inputs: the MN, ON, WN, XI, XU, and XW commands as well as function triggering via parameters 36 and 37. Thus a False-to-True transition could be caused either by a High-to-Low or Low-to-High voltage change.

No functions are triggered as the immediate result of changing parameter 38. In other words, if an Input is sitting False and you change the definition so that it is now True, no function is triggered. The input must subsequently go False, then True again in order to trigger a function.

Autostart

39: Autostart Sequence I.D., range -1 to 255, factory -1

If this parameter is in the range 0 to 255, it specifies the I.D. of a Download Sequence which is to be executed automatically after subsequent power ups. If it is -1, no Download Sequence is automatically executed.

The Autostart sequence can be temporarily suppressed by holding down the FAST button (the unlabeled button in the center of the four Arrow buttons) while turning on the power. You must hold the button for several seconds to be sure the Autostart sequence is suppressed.

Additional Z Axis Parameters

40: Z Axis Arrow Slow Step Delay, range 0 to 65535, factory 200

This sets the time delay between Z axis slow Arrow button steps (either half steps or full steps). It is analogous to parameter 2. The delay is multiplied by approximately 100 microseconds.

In order to use the Arrow buttons to control the Z axis you must set the Arrow mode appropriately; see the AM command in Chapter 3. Z axis Arrow single step delay is controlled by parameter 1.

41: Z Axis Arrow Fast Step Delay, range 0 to 65535, factory 25

This sets the time delay between Z axis fast Arrow button steps. See parameter 40. The parameter is multiplied by approximately 100 microseconds.

42: Z Axis Arrow Direction Swap, range 0 to 255, factory 0

If this parameter is zero the Down Arrow button moves the Z axis motor toward lower-numbered Z coordinates and the Up Arrow, toward higher. If the parameter is nonzero the functions of the Up and Down Arrows are swapped. You should pick an appropriate value for parameter 31 before choosing a value for parameter 42.

In order to use the Arrow buttons to control the Z axis you must set the Arrow mode appropriately; see Arrows in Chapter 4 and the AM command in Chapter 3. Z axis Arrow single step delay is controlled by parameter 1.

43: Default Arrow Mode, range 0 to 255, factory 0

This parameter sets the default Arrow mode which is applied at power up, at IN, and at an AM with no parameters. See the AM command in Chapter 3 and the Z AXIS button in Chapter 4.

44: Default Z Axis Step Rate, range 1 to 10000, factory 100

This sets the default Z axis step rate which is applied at power up, at IN, and at a CZ with the <rate> parameter omitted. See the CZ command in Chapter 3.

45: Default Z Axis Modulus, range 1 to 32767, factory 32767

This sets the default Z axis modulus which is applied at power up, at IN, and at a CZ command with the <mod> parameter omitted.

46: Default Z Axis Half Step Mode, range 0 to 255, factory 1

This sets the default Z axis half-step flag which is applied at power up, at IN, and at a CZ command with the <halfflag> parameter omitted.

47: Default Z Axis Acceleration, range 1 to 10000, factory 40

This sets the default Z axis acceleration which is applied at power up, at IN, and at a CZ command with the <accel> parameter omitted.

48: Default Find-Z-Before-XY-Home, range 0 to 255, factory 0

This parameter controls whether the Z Home switch is always found automatically before the X and Y switches in an FH command or XY manual re-reference operation. It also controls whether the GO TO ORIGIN button's XY motion is preceded by an automatic Z Find Home. (This can be used to prevent crashing into fixturing, etc., during the XY motion.) See the FH and FZ commands in Chapter 3; see also the GO TO ORIGIN button and the manual re-reference operation in Chapter 4.

Table 27 - Find-Z-Before-XY-Home Default Value Interpretations

Value	FZ before FH	FZ before GO TO ORIGIN
0	No	No
1	Yes	Yes
2	Yes	No
3	No	Yes
other	Yes	Yes

Step Verify Options

49: Step Verify Control, range 0 to 3, factory 0

This parameter controls the behavior of the Step Verify option. Each bit has a specific meaning, as follows:

Table 28 - Step Verify Control Value Definitions

Bit	Dec. Value	Meaning, if 1
0	1	If this bit is set then the Automove System suppresses the automatic Emergency Stop at slip detection. The machine will continue to execute motion commands after a slip is detected.
1	2	If this bit is set then the Step Verify circuit remains active while the motor current is off. If any XY motion occurs due to external forces the machine will enter the Motor Slipped state. See the PM command.

50: Slip Response Download Sequence I.D., range -1 to 255, factory -1

If this parameter is in the range 0 through 255 it specifies the I.D. of a Download Sequence which is to be automatically executed when the optional Step Verify circuit detects a slip of the X or Y stepping motor. The Download Sequence is started after the automatic Emergency Stop, if any. (See parameter 49, above.)

If parameter 50 is -1 then no Download Sequence is executed when a slip is detected.

The Slip Response Download Sequence executes in a privileged mode. Normally the Automove System will not interrupt an interrupt. (See "Interrupting An Interrupt" in Chapter 8.) The Slip Response Sequence is allowed to interrupt an interrupt (unless it exceeds the maximum nesting depth) and cannot itself be interrupted by anything other than another Slip Response sequence.

Note that when a slip is detected the machine may become Emergency Stopped. If the automatic Download Sequence is to perform any physical action then it must first execute a CS ("Clear Stop") command. The CS will automatically be delayed a few moments to allow things to come to rest after the slip is detected, so that another slip will not be immediately detected. (See parameter 51, below.)

Caution: Emergency Stop causes the XY position reference to be lost in unexpected ways. You must execute an FH or SP command before attempting any vectors or arcs after an Emergency Stop.

Use caution when designing automatic responses to motor slip. Be sure you have considered all of the circumstances under which a slip might occur, and that your automatic response isn't more destructive than the thing that caused the slip. Consider whether another slip might occur during the response to a slip.

In general, the automatic response should be simple and brief. It should start with a CS to clear the Slipped and Stopped states so that the Digital Outputs can be changed, and should end with an AD ("Abort Download Sequence Execution") command. It is usually best to call for operator intervention, perhaps by turning on a light or buzzer. Humans are much better at "fixing things up" than machines!

Note that Step Verify can be temporarily locked out via the FP command. If a slip occurs while locked out then the SLIP light glows steadily (i.e. does not blink) and the slip is not acted upon until a subsequent FP re-enables Step Verify. Also, Step Verify is completely deactivated while motor current is off; see the PM command.

51: Step Verify Delay, range 0 to 65535, factory 500

This parameter establishes a delay, in milliseconds, which is to be taken automatically before executing any CS command or "ESC.!2:" while the System is in the Motor Slipped state. The purpose is to give the mechanical system time to coast to a stop after an Emergency Stop caused by a slip. If a CS command were executed while the motors were still moving then another slip would be detected. (Note that the WA command is ignored while the machine is Emergency Stopped.)

Additional Z Axis Parameters

52: First Find Z Home Back Off Distance, range 0 to 32766, factory 0

This parameter sets the number of steps that the Z motor moves away from the Z home switch before the first of the two seeks performed by an FZ command or a manual Z-axis Find Home. Note that the factory default is 0, so there is no motion. (This prevents bending the needle in a fluid dispensing system if the needle is already resting against the surface.)

53: Second Find Z Home Back Off Distance, range 0 to 32766, factory 150

This is like parameter 52, but is used in the second Find Z Home sequence rather than the first.

54: First Find Z Home Seek Speed, range 1 to 10000, factory 750

This sets the speed, in steps per second, at which the Z motor seeks the Z home switch during the first of the two seeks performed by an FZ command or a Z axis manual find home.

Note that parameters 54 through 56 do not take effect until the next CZ, IN, or reset.

55: Second Find Z Home Seek Speed, range 1 to 10000, factory 100

This is like parameter 54, but is used in the second Find Z Home sequence rather than the first.

56: Find Z Home Acceleration, range 1 to 10000, factory 10

This sets the acceleration, in thousands of steps per second per second, at which the Z motor accelerates toward the Z home switch in both the first and second Find Z Home sequences. The motor accelerates smoothly from zero speed to the value specified by parameter 54 or 55, but stops instantly as soon as the Z home switch closes.

57: Z head identification code. Must be 0 or 128, factory 128.

Must contain the I.D. code of the Z head being used since wiring to the Z head for automatic identification has been changed to implement the retractable height sensor. 128 is used for 1 mil Z heads and 0 is used for 2 mil Z heads.

58: Calibration map code for square or rectangle. Must be either 0 for square or 1 for rectangular, factory 0.

Allows a special rectangular calibration map to be used with DispenseJet products only.

59: Paused Digital Outputs, range 0-255, factory 0

This parameter will cause the dispensing valve to be shut off if the machine is PAUSED from the front panel or if a PS command is received. The value must be set to a non-zero value whose bits correspond to the digital output bits to be turned off. When the machine pauses, this command effectively turns off a valve that was turned on by the Fluidmove **zfast** dispensing function. The valve will not be turned back on when the machine is un-paused. If PE 59 is 0, no output changes will occur when the machine is PAUSED from the front panel or a PS command is received. PE59,2; would be the typical value used for this feature but other bits could be used if a non-standard valve or two valves are used. Non-specified bits will not be affected.

60: Interlock Control, range 0-255, factory 0

Automove systems use a two processor architecture (8085 IOP and 6803 VGP) that utilize "pipelining" to increase speed. This means that the IOP can serve a new ACL command and pre-process it while the VGP is executing the previous command. Each time a new command is sent from the IOP to the VGP, the machine's digital inputs and front panel buttons status is transferred from the VGP to the IOP. This causes the IOP to become "aware" of any interlock download sequence requests AFTER a command is sent to the VGP. Thus, while making the first move of two back-to-back moves, if the VGP detects a digital input request for an interlock download sequence, the IOP will not become aware of this state until AFTER the VGP receives the second move command. All commands are processed to completion except the STOP request. Normally, two back-to-back XY moves serve no purpose and should be avoided. However, Fluidmove can currently send two back-to-back XY moves during spiral fills, and a pending interlock download sequence will not occur until the second move finishes.

If personality parameter 60 is set to 1, pipelining will be disabled and the IOP will automatically get new VGP status after an MA or MR command is issued BEFORE issuing a new VGP command. This allows an interlock to occur between moves without both moves completing first as previous firmware versions required. HOWEVER, this change effectively reduces throughput since it disables pipelining. If personality parameter 60 is set to 0, interlock actions will remain the same as version 3.59 EPROMS, but there will be no reduced performance.

8 Using the Digital Outputs and Inputs

The Digital Outputs

The Digital Outputs are optically-isolated logic outputs tied to a back-panel connector. They are always High True (i.e. High = True, Low = False), assuming you connect them in the grounded-emitter configuration. At power up and Emergency Stop they are all set False (i.e. Low).

For the connector pinout and electrical specifications, see the Automove Operation manual.

The Digital Outputs can be used to control any sort of external hardware such as relays, solenoids, valves, lamps, measurement devices, cameras, etc.

There are several advantages to using the Digital Outputs instead of, say, some other output channel connected directly to your host computer. First, the Digital Outputs can be tightly synchronized to the move sequence. Second, they can be actuated by a Download Sequence, so that your system can operate without a host computer. Third, the value output can be changed via the Digital Inputs, so that manual or externally-controlled interaction is possible.

The Digital Outputs are affected by the following ACL commands: CD ("Change Digital Outputs"), MD ("Mid-move Digital Outputs Changes"), MM ("Multiple Mid-move Digital Outputs Changes"), OD ("Output Digital Outputs State"), PD ("Pre-move Digital Outputs Changes"), TD ("Toggle Digital Outputs"), and WD ("Wait After Digital Outputs Changes"). They can also be affected by the Digital Inputs if personality parameters 36-38 are set appropriately; see Digital Inputs, below.

Simple Use

The easiest way to use the Digital Outputs is to send the CD command as part of your move sequence. As soon as the CD command is executed, the Digital Outputs change to the specified new values.

The <which> parameter of the CD command can simplify your host software. If your host program has several separate subroutines to control several different devices, each subroutine can change only the Digital Output(s) it is concerned with, leaving the others alone.

If your external hardware needs time to respond before the next operation can begin, you can use either the WA ("Wait") command or the WD command. If you find yourself sending the same WA command after each CD command, you can switch to using a single WD command before all the CD's.

For example, suppose you have connected a relay to Digital Output 0, and suppose the relay needs 0.01 seconds to close and 0.02 seconds to open. When you send CD 1; the relay closes; CD 0; opens it. If you first send WD .02; the System automatically waits .02 seconds after any CD command. If you would rather, you can save .01 seconds now and then by using pairs of commands: CD 1; WA .01; and CD 0; WA .02; .

Getting Fancier

Now suppose you need to change the Digital Outputs while the motors are moving. The classic example of this is a fluid-dispensing system: a small pipette is attached to the carriage. A solenoid- operated valve starts and stops the fluid flow through the pipette, with some consistent but unavoidable delay in response.

In order to avoid a gap in dispensed fluid at the beginning of each move, the valve must turn on a short while before the carriage begins to move. And in order to avoid getting a blob of fluid at the end of each move, the valve must turn off before the carriage has come to rest.

The solution here is the PD command and the MD or MM command. Use PD to set up a valve turn-on and delay. Then decide how far before the end of the move, to turn off the valve, and send an MD or MM command with a negative <count> to turn it off just before the end. The same PD and MD/MM will work equally well for vectors, arcs, and Continuous Path moves.

If you have an occasional move in which you don't want to dispense any fluid, just use the VM ("Vector Mode") command to temporarily disable the PD and MD commands, and perhaps even to speed up the motors. VM needs fewer characters to transmit (and less download memory) than PD and MD/MM, and executes faster within the Automove System.

If you need more than two changes in a single move, use the MM command instead of the MD command.

Reading Them Back

The OD command allows your host software to read back the current state of the Digital Outputs. This might be useful if, for example, you have several host programs which need to work together in controlling the same Outputs; or if the host software needs to see which state the Outputs were left in by a TD command.

Don't confuse the OD command with the ON command, which reads the Digital Inputs.

The Digital Inputs

The Digital Inputs are eight logic inputs tied to a back-panel connector. For the connector pinout and electrical specifications see the Automove Operation manual.

There are eight ways to use the Digital Inputs:

1. Your host computer can test the state of the inputs by using the ON ("Output Digital Inputs State") command.

2. The move sequence can be delayed until a particular condition occurs, by using the WN ("Wait For Digital Inputs") command.
3. You can execute one of two Download Sequences depending on whether a certain condition does or does not exist on the Digital Inputs; this is the XI ("Execute If") command.
4. You can repeat a Download Sequence until the Digital Inputs are in a certain state; this is XU ("Execute Until").
5. You can repeat a Download Sequence while a certain state exists on the Digital Inputs; this is XW ("Execute While").
6. A certain state of the Digital Inputs can cause automatic actions to occur during vectors, arcs, Continuous Path moves, or Z moves; this is set up by MN ("Mid-move Digital Inputs Response").
7. A transition on one of the Digital Inputs can toggle the state of a Digital Output.
8. A transition on one of the Digital Inputs can invoke a Download Sequence as an interrupt.

Logic Sense and Debounce

The logic sense of the Digital Inputs can be changed to be either High True or Low True; see personality parameter 38 in Chapter 7. This logic sense applies equally to all eight ways of using the Digital Inputs. For example, the command WN 1,1; waits for a True on Input 0, regardless of whether the True is to be represented by a high voltage or a low voltage.

Note: the following discussion about debouncing does not apply to the automatic mid-move responses set up by the MN command. See MN.

The Automove System contains two microprocessors: an Input/Output processor (IOP) for receiving and processing characters from the host computer, and a vector generator processor (VGP) for controlling the motors and the front panel. For purposes of toggling Digital Outputs or invoking Download Sequences the Digital Inputs are polled (i.e. periodically checked) by the VGP when it isn't busy generating motion or responding to a front panel button.

Consequently, if a Digital Input changes during a vector, arc, Continuous Path move, or WA wait, the VGP will not see the change until after the motion or the wait is complete. Also, if an Input changes from one state to the other and back during a move, the entire pulse will be missed. Thus you must be sure that your electrical signals remain valid long enough to be seen by the VGP. If things happen too quickly in your application, you may have to add external logic circuitry.

The VGP processes the inputs so as to remove the effects of switch contact bounce. The debouncing is performed as follows: First, the logic sense is evaluated as prescribed by personality parameter 38. Then the signals are analyzed in the following way. If an Input was False and is now seen to be True, it is immediately declared True. It is not declared False until it has been seen to be uninterruptedly False for a period of approximately 20 milliseconds (0.020 seconds).

It is this declared value which drives all seven non-MN ways of using the Digital Inputs. Thus a False-to-True transition can produce an immediate result, whereas a True-to-False transition will not be responded to for at least 20 milliseconds. If possible, you should arrange the logic sense (via personality parameter 38) so that any transition requiring immediate response is False-to-True.

If your switch contacts bounce for more than 20 milliseconds you can do one of two things. Either add external hardware to debounce the switches, or use careful programming in your ACL move sequence, to ensure that the switch bounce does not have bad effects. For example, make sure that the response to a switch closure takes longer than the bounce time, and that the switch is ignored during the response (i.e., don't retest the switch immediately and don't use interrupts).

During the 20 milliseconds following any True-to-False transition the VGP is busy watching the Digital Inputs. Therefore there will be a slight delay in your move sequence. In most cases this delay will not be objectionable, but in some applications you may need to arrange your external hardware so that the Digital Inputs do not dither unnecessarily.

If the Automove System is used in an electrically noisy environment you may need to shield the cable connected to the Digital Inputs. Otherwise, electrical noise may cause inaccurate responses as well as unnecessary debounce delays.

Sending Them Back To The Host

By using the ON ("Output Digital Inputs State") command the host computer can determine the current (debounced) state of the Digital Inputs. This can be used for: displaying system status on a screen; executing conditional branches within the host software (perhaps to simulate the behavior of XI in a Download Sequence); waiting for other hardware to finish some action; determining the state of a configuration or mode switch.

If you use ON to perform conditional branches in the host you will probably get the same results as if you had put an XI command into a Download Sequence. However, the ON/host scheme does not run as fast as the XI command would; so under some conditions the timing differences might cause different behavior. You might avoid this problem by using partly downloaded ACL and partly "live" ACL; the host sends an XI command which executes a previously-downloaded sequence.

In general, however, it is wise to eliminate those situations in which timing is critical, because a front-panel PAUSE or an interrupt might cause them to break down anyway.

Don't confuse the ON command with the OD command, which sends back the last value applied to the Digital Outputs.

Waiting

The WN ("Wait For Digital Inputs") command is useful when you need to delay or hold off the move sequence until some action is complete or until some triggering event occurs. For example, suppose a Digital Output is connected to a solenoid valve which controls a pneumatic cylinder. After actuating the cylinder you must wait until it completes its travel before beginning to move the motors.

The WN command's <which> parameter lets you specify that you only want to include certain Digital Input(s) in the test. Be sure to be as restrictive as possible about which Inputs you test. If certain Inputs don't matter, don't test them, because they might not be in the states you think they are. Usually you will only be testing one Input, so <which> will have only one bit set (i.e. it will be 1, 2, 4, 8, 16, 32, 64, or 128).

For example, you might think that the command WN 2; just waits until Digital Input 1 is True. However, it actually checks all 8 Inputs; if Input 5 were True the wait would not terminate even when Input 2 went True. A better command would be WN 2, 2; . This one looks at Input 2 and disregards all the rest.

The WN command's <timeout> parameter lets you place an upper limit on how long to wait. This is useful, for example, if the external hardware might occasionally fail to respond. To test for whether the timeout occurred, you can follow the WN command with an XI which tests the same condition; if the XI's <else id> executes, the condition still doesn't exist and the timeout probably occurred. (There is a small uncertainty since the Input(s) may have changed between the WN command and the XI command.)

If you want your system to perform some other action while waiting for an input, try the XU or XW command; see below.

Conditional Execution

The XI ("Execute If") command is like the "if" statement of a computer language. It says "do this if and only if a certain condition is true". With XI, the condition is some specified set of values on the Digital Inputs. Typically you will only be testing one Input, but you can test more than one if necessary.

XI also has a <which> parameter. As with the WN command, you should be as restrictive as possible when deciding how many Inputs to test.

Repeating

The XU ("Execute Until") and XW ("Execute While") commands are like the "repeat...until" and "while...do" loops of a computer language. A Download Sequence is repeated until a condition exists, or while a condition exists, on the Digital Inputs. Use XU if you want the Sequence always to be executed at least once; use XW if the Sequence should not be executed at all if the condition does not already exist.

XU and XW have a <which> parameter. As with WN and XI, you should be as restrictive as possible about which Inputs you test.

Notice that XU quits when the condition **is** met, but XW quits when the condition is **not** met. In switching from one to the other you must invert the sense of the crucial bit in the <value> parameter. If you are testing more than one bit you may need to use XU instead of XW, or vice-versa. You may also have to add one or more XI ("Execute If") commands to achieve the desired behavior.

Responding In Mid-move

The MN ("Mid-move Digital Inputs Response") command can set up automatic Digital Inputs responses that occur while the carriage or the Z motor is in motion. The System can "remember" where it was when the Digital Inputs reached a particular state and can immediately decelerate and come to a stop.

This capability is especially useful where there is a piece of external equipment monitoring some quantity which is affected by the Automove System's motion. (For a laser trimming example see the MN command in Chapter 3.)

To prevent unexpected results you should be sure that the MN features are only enabled during the particular move or group of moves where you desire a response.

Timing Considerations

You must design your ACL move sequence and your external hardware carefully to prevent timing problems when using the Digital Inputs. In general, you must ensure that inputs occur early enough, occur in the right order, and last long enough to be seen.

Suppose that in your application you wish to test the state of an Input and perform one action if the Input is True or a different action if it is False. You decide to put two XI commands, one following the other, the first testing for True, the second testing for False. When the first XI executes the Input is True, so its action is performed. But meanwhile the Input goes False. Then the second XI is encountered, and its action is performed as well! In cases like this it is safest to use the <else id> parameter of the XI command, in order to guarantee that only one of the two actions is performed.

Similarly, suppose you wish to perform some action only if several Inputs are all in the correct states. Where possible, it is safest to test them all in the same XI command, rather than testing some of them from within the Download Sequence invoked by the first XI. Otherwise, the first one you test may change before you test the others.

If a transition requires an extremely prompt response you should arrange for your move sequence to be waiting for the transition via a WN command. The delay between a False-to-True transition which terminates the WN wait, and the beginning of the next ACL command, can be well under a millisecond if the following command is a CD or TD, and about 3 milliseconds for an MA or MR vector.

Interrupts and Other Methods

If you set up personality parameters 36 - 38 appropriately, a False-to-True transition on a Digital Input will trigger a Download Sequence. If another Download Sequence was already executing it will be interrupted; after the interrupting sequence is finished the original sequence resumes where it left off. Interrupts can also be caused from the front panel, by pushing the FAST/PAUSE, FAST/TEACH, or GO buttons.

Interrupts are useful for starting or changing the course of a move sequence as the result of external inputs which may occur at unpredictable times. These inputs might result from a person pushing a button or from the activation of some switch or sensor in your system.

You may wish to disable interrupts during certain critical sections of your move sequence; for example, while fluids are flowing or heaters are turned on. You should not use personality parameter 36 for this purpose; instead, use the FP ("Front Panel Lockout") command.

If an interrupt occurs while locked out by FP the interrupt is not forgotten; it is merely postponed until a subsequent FP re-enables interrupts. Only one interrupt from each Digital Input or front panel button can be remembered in this way. If a second one occurs before the first one is serviced the second one is lost.

Remember that the maximum nesting depth of Download Sequences is 12, including interrupts. See **Interrupting an Interrupt**, below.

Interrupts have a one-command latency. If the System is sitting idle when a False-to-True transition occurs, the Download Sequence begins executing after only a few milliseconds of delay. But suppose the Automove System is performing a series of vectors. It performs vector A, then vector B. Sometime during vector A an Input changes from False to True. The interrupt will not occur until after vector B is complete!

Also, remember that the System only looks at the Digital Inputs and front panel between vectors. Therefore a short pulse might be missed.

Because of these limitations, there are several cases when interrupts should NOT be used. If the exact order of command execution is critical, you should use the XI command at appropriate places in your move sequence, instead of an interrupt. If an Input transition requires an extremely prompt response (say, less than a few milliseconds) or if the signal will be of brief duration, you should arrange for your move sequence to be waiting in a WN command for the transition to occur.

Don't overlook the technique of putting an "idle loop" into your move sequence. This can be an infinitely-repeating Download Sequence which periodically checks and responds to the Digital Inputs via an XI command. Meanwhile the idle loop can be performing some other action which is not allowed to be interrupted, except at one well-defined point in the loop; i.e., the XI command.

A non-interrupt technique for getting one and only one response per switch closure is as follows. Start the response as the result of a WN command or an XI command. Then, at the tail end of the response, use a WN to wait for the switch to open again. If using XI, be sure to put the final WN within the response Sequence itself. If you put the WN after the invoking XI and if the switch closes just after the XI but before the WN, an entire pulse will be missed.

If you decide to use WN or XI instead of interrupts, be sure to disable the interrupts via personality parameter 36, or you will get unexpected responses.

Interrupting An Interrupt

The maximum nesting depth of Download Sequences is 12, including interrupts. In other words, suppose Sequence 1 has called Sequence 2, 2 has called 3, and so forth up until 11 has called 12. Then if an interrupt occurs the maximum nesting depth will be exceeded. Or, if only 11 Sequences are nested when the interrupt occurs, but then the interrupting Sequence attempts to execute an XD or XI command, the maximum depth is exceeded.

When you attempt to exceed the maximum depth, instead of invoking a Download Sequence the System just logs an ACL error and halts Download Sequence execution.

To minimize the possibility that runaway or uncontrolled interrupts will cause this error, the Automove System does not allow you to interrupt an interrupt. In other words, suppose the System is sitting idle (or is just executing ACL commands from the host) when Download Sequence A is invoked, either via an XD or XI command or via the Digital Inputs or front panel. Sequence A is not considered to be an interrupt. Now a Digital Input or the front panel invokes sequence B; this is an interrupt. Now if a Digital Input or the front panel attempts to invoke sequence C (or sequence B again) this new interrupt is held off; the System postpones any response until after sequence B finishes executing.

When designing your Download Sequences you must make sure that the worst case non-interrupt nesting depth plus the depth of the worst- case interrupt routine itself does not exceed 12.

Toggling The Outputs

Via personality parameters 36 - 38 you can set up the Digital Inputs to toggle (i.e. change) the Digital Outputs directly, without involving any ACL commands or Download Sequences.

It would be easy to set up Download Sequences to perform the same function via the TD ("Toggle Digital Outputs") command. However, the built-in toggling feature is immune to accidental erasure of Download Sequences, and also it leaves all the Download Sequences available for other uses.

If Digital Inputs 0 and 1 have been set up for toggling, Download Sequences 0 and 1 can still be invoked via the FAST/PAUSE and FAST/TEACH buttons, respectively.

Don't forget that the WD command's wait is executed after each toggle. This ensures that the necessary hardware response times are allowed for, even if a toggle occurs during a running move sequence. If this delay is not desired, don't use the WD command; instead, put a WA command after each CD or TD command in the move sequence.

9 Linearity Correction

Whereas the calibration factors provide an overall stretching or shrinking of the size of a Calibrated Unit, the Automove System also contains a linearity correction mechanism to produce small regional adjustments over the entire travel of the motors.

The motor travel is divided up into a rectangular grid. There can be up to twenty grid points along each axis, for a total of 20 x 20 = 400 grid intersections. The spacing between grid intersections can be specified. For Rev 3.46 and Revs 3.51 and up, a 49 x 49 grid matrix is used.

When the Automove System receives an XY motion command, for example MA, it first multiplies the specified X and Y coordinates by the calibration factors, then adds the (reverse-corrected) Origin offset to produce the uncorrected microstep location. Then the System consults the XY Linearity Correction Table.

If the coordinate happens to fall exactly on a grid intersection then the X and Y corrections specified for that intersection are used. Otherwise, the actual corrections are determined by interpolating between the corrections specified for the four nearest grid intersections. ("Interpolating" means smoothly blending the effects of the four nearest intersections; the closer the commanded position is to a particular intersection, the stronger the effect of that intersection's correction values.)

After determining the appropriate X and Y correction values the System adds them to the uncorrected microstep location. Then, for example, in the case of the MA command, the carriage moves directly to the corrected location.

When the host computer asks for the Calibrated Unit equivalent of a physical motor location (via the OC command or the OT command) the Automove System uses the Linearity Correction Table "in reverse" to produce the corresponding reverse-corrected values.

Setting Up The Correction

The values in the table are set up via the CR ("Correction") command. The syntax is as follows:

CR <x index>, <y index>, <x correction>, <y correction>

Each CR command sets the X and Y corrections for one grid intersection. The <x index> and <y index> parameters determine which grid intersection is being referred to; they can be in the range 0 through 19. For example, (0, 0) is at the Home switches and (1, 2) is one grid intersection away in X and two in Y.

The <x correction> and <y correction> parameters determine how much correction is applied at that grid intersection. They are in microsteps.

The values of <x correction> and <y correction> must be limited such that the maximum correction is in the range -4.000 to +3.9688 full steps in either axis. Recall that the size of a microstep is specified by the RE ("Resolution") command. You can specify 1.0000 through 32.0000 microsteps per full step. For example, the Automove factory default is 8.0000 microsteps per full step; at this value the <x correction> and <y correction> parameters can range from -32.0000 to 31.7500 microsteps.

The Automove System internally stores the <x correction> and <y correction> parameters and the correction grid spacings in such a way that they remain independent of the resolution. Therefore, you can change the microstep size and the correction information will still relate in the same way to the physical travel of your mechanical system.

For example, if you have an Automove 403 XY table and the X correction at (12, 0) inches from the Home switches has been set to +0.003 inches, the same correction will apply whether you use RE2,2; or RE8,8; . The OR (Output Correction) command (see below) would return 0.75 while RE2,2; is in effect, and 3 while RE8,8 is in effect, even though no new CR command had been executed.

If you do not execute a CR command for a particular grid intersection then the correction values for that grid intersection remain 0, and no correction is performed there. If an adjacent grid intersection has nonzero values then the System blends smoothly from the nonzero values to zero over the intervening distance.

Beyond the ends of the grid the Automove System performs no correction. For example, suppose you have set the X and Y grid spacing to 1000 microsteps via a CR -1, -1, 1000, 1000; as described below. Also assume you have set all of the grid intersections to a correction of +10 microsteps in X and Y (via CR 0, 0, 10, 10; CR 1, 0, 10, 10; CR 2, 0, 10, 10; ... CR 19, 19, 10, 10;). A correction of +10 is applied to all coordinates in the range 0 through 19000 microsteps. If an MA occurs where either X or Y is outside the range 0 through 19000, no correction occurs.

The linearity correction mechanism affects only the AA, AR, MA, MR, OC, OT, and SP commands. It does not affect the travel limits, the Origin position, the MT command, the acceleration, the step rate, or mid-move Digital Outputs changes; these are all set directly in microsteps. See the TL, SO, MT, AC, SR, MD, and MM commands.

Setting The Grid Spacing

A special case of the CR command sets the correction grid spacing. If you specify -1 for <x index> and <y index> then the <x correction> and <y correction> parameters become the X and Y grid spacing, respectively, in microsteps. The grid spacing must be in the range 0.0313 through 1023.9688 full steps. For example, at the default resolution of 8.0000 microsteps per full step the grid spacing can range from 0.2500 through 8191.7500 microsteps.

Example

The following sequence of commands sets a correction grid spacing of 4000 microsteps in each axis and then establishes X and Y corrections for each grid intersection up to 8000 microsteps in each axis (i.e. at 0, at 4000, and at 8000 microsteps in each axis):

```
CR -1, -1, 4000,4000;
CR 0, 0, 2, 3.5;   CR 1, 0, -1, 4;  CR 2, 0, -2, 6;
CR 0, 1, 4, 7.75;  CR 1, 1, -2, 6;  CR 2, 1, 7, 6.5;
CR 0, 2, 5, 6;     CR 1, 2, -4, 6;  CR 2, 2, 6, 5;
```

Note: Some of the corrections are not whole numbers of microsteps (e.g. 7.75). At these grid locations the System adds this non-integer number of microsteps, but then it rounds to the nearest microstep.

Disabling the Correction Mechanism

As a further special case the linearity correction mechanism can be completely disabled by specifying both the X and Y grid spacings equal to 0. In other words, to disable linearity correction, send the following: "CR -1, -1, 0, 0;". The values in the table are unaffected. To re-enable correction, send another CR with the desired values of X and Y grid size.

Non-Volatile Storage

The linearity correction information is stored in the Automove System's non-volatile memory. This means that the System "remembers" the information even when the power is turned off.

Be sure to flip the backpanel Write Protect switch down (or up in a gantry-mount machine) before executing any CR command. If you don't, the CR command will give an error.

Outputting To The Host

The host computer can determine the current correction grid spacing and the values stored in the correction table by sending the OR ("Output Correction") command. The syntax is:

OR <x index>, <y index>

where <x index> and <y index> are interpreted as in the CR command. In other words, if <x index> and <y index> are in the range 0 through 19 then the Automove System sends the current values of X and Y correction, separated by a comma. The values are in micro-steps, in the range -128.0000 through 127.0000.

If <x index> and <y index> are both -1 then the Automove System outputs the X and Y correction grid spacing, separated by a comma. The values are in microsteps, and are either 0 (if the mechanism is disabled) or are in the range 0.0313 through 32767.0000.

If <x index> or <y index> is not in the range -1 through 19 then the System logs an error and outputs 0,0.

Reinitializing

The entire XY Linearity Correction Table can be set back to its "factory default" state by sending the ESC.!8: command; see Chapter 6. In the factory default state all of the X and Y correction values are zero and the grid spacing is zero in each axis, which disables the linearity correction mechanism. Use caution -- this command erases the linearity correction which was measured at the factory, and thus may cause the Automove System to fail to meet its accuracy specification.

Limitations

Normally, the command MA0,0; always moves exactly to the Origin location. The System reverse-corrects the Origin location as established by the SO command or the SET ORIGIN button. This reverse-corrected Origin is added before the linearity correction table is consulted; thus, the effects of linearity correction and Origin are separated.

Under some conditions the reverse-correction mechanism may be slightly inaccurate. This may occur when the correction grid spacing is very small (say, 100 microsteps) and large correction differences exist between adjacent grid locations (say, 20 microsteps). Under these conditions the MA0,0; may move to a point which is two or three microsteps away from the true Origin. No errors will occur if "normal" grid spacings of several thousand microsteps are used.

10 Using ACL Variables

What Are Variables?

The Automove System maintains 128 user-accessible storage locations for numeric values; these are called variables. Each location is referred to by an identifying number; the i.d. number can be in the range 0 through 127. Each variable can contain a value in the range -32768.0000 through 32767.9999.

Note: If you have Release 3.48 or above, you may have 384 variables.

The value of a variable can be used as a parameter to any ACL command which expects a numeric parameter, by using the "@" ("at") sign, followed by the variable i.d. For example:

```
MA 100, @12;
```

moves the carriage to the X coordinate 100 and the Y coordinate given by the contents of variable 12.

If the number following the "@" is outside the range 0 through 127 an error is logged.

How To Set Variables

The value of a variable can be set in several different ways. The VS ("Variable Set") command can directly assign a value; for example:

```
VS 13, 321;  VS 14, @21;
```

assigns the value 321 to variable 13 and assigns the contents of variable 21 to variable 14.

A VS without a value in it (e.g. VS 13;) waits for the value to be supplied via the RS-232C port. This might be useful, for example, in a system where the Automove's RS-232C port is used to control some sort of measuring instrument. The Automove could send commands to the instrument via the OU ("Output Literal String") command and store the resulting measurement data in variables for further processing.

Another use for VS with no value is to prompt the operator of a system. In this system the Automove's RS-232C port would be connected to an ASCII terminal or a computer running a terminal emulation program. The Automove could prompt the operator to insert a workpiece, adjust the Origin, and then enter the number of operations to be performed.

Another way of assigning a value is via the VC ("Variable Capture") command. The syntax is:

```
VC <varid>
```

where <varid> is a variable i.d. After this command has been executed, the next ACL output command (OA, OC, etc.) sends its values into variable(s) instead of sending them to the host computer. <varid> specifies the first variable which is to receive a value. Subsequent values go into the next higher-numbered variable(s). For example:

```
VC 11; OC;
```

puts the current commanded X position into variable 11 and the current commanded Y position into variable 12.

Other commands which can change the value of a variable are: V+, V-, V*, V/, V&, V|, V!. Respectively, these commands add a value to a variable, subtract a value, multiply by a value, divide by a value, logically AND with a value, logically OR with a value, and bitwise ones'-complement a variable. The logical operations always return zero for the fractional part of the value.

Finally, the SC ("Scale") command converts between Calibrated Units and raw microstep positions; the VL (Vector Length) command computes the length of a vector; the VR (Vector Rotate) command performs a trigonometric rotation of a vector endpoint about a specified point; and VA (Vector Angle) computes the rotation angle of a specified vector.

The ESC.!9: escape sequence sets the values of all variables to 0.0. Since the variables are stored in non-volatile battery-backed RAM their values are not affected by turning the power off, then back on. It is the user's responsibility to see that the variables get initialized to the proper values at the proper times.

How To Test Variables

The OV ("Output Variable") command sends the current contents of a variable to the host computer. For example, OV 8; sends the contents of variable 8.

To modify the execution of a Download Sequence based on the value of a variable, use one of the following commands: VT ("Variable Test"), V< (Variable Less Than), V= (Variable Equal To), or V> (Variable Greater Than).

After a VT command has been executed, the next XI, XU, XW, or ON command tests the value of the variable (bits 0 through 7), rather than testing the Digital Inputs.

The V< command tests if the contents of a variable is less than a specified value. The next XI, XU, XW, or ON can check the results of the comparison; bit 0 will be "1" if the comparison was True (i.e., if the variable was less than the specified value). Similarly, V= tests for a variable being equal to a value, and V> tests for greater than. For example:

```
V= 3, 19.33; XI 40, 1, 1;
```

executes Download Sequence 40 if and only if the value of variable 3 is exactly 19.33.

Another example:

```
V> 5, 20; XU 50, 1, 1;
```

executes Download Sequence 50 until the value of variable 5 is greater than 20. For a less-than-or-equal-to test:

```
V> 5, 20;  XU 50, 0, 1;
```

executes Download Sequence 50 until the value of variable 5 is less than or equal to 20. (Note the 0 in the XU command; it means "execute until the V> is false".)

Indirection And Indexing

A form of "indirection" or "array indexing" is also available. For example, suppose variable 3 contains the value 17. Then the command:

```
OV @3;
```

outputs the contents of variable 17.

Suppose variable 6 contains the value 12, and variable 12 contains 100. Then MA @@6, 20; moves to the coordinate 100, 20.

You may put up to 4 "@" characters before a number.

Numeric Representation Errors

Within the Automove System, the value of a variable is represented in 32 bits of memory. The first 16 bits are the integer portion, generally interpreted as a binary 2's-complement value. The last 16 bits are the binary fractional portion, with a fixed binary point between the integer and fractional portions.

Because of this representation, the fractional part is only accurate to one part in 2^16, or one part in 65536. This translates to about 7% error in the number 0.0001, which is the smallest nonzero number which can be parsed and input into the Automove System.

The representation error becomes significant for numbers whose values are much smaller than 1.0. For example:

```
VS 0, 0.0001; V* 0, 10000;  OV 0;
```

gives the result 1.0681 because of the error in representing 0.0001. Therefore, to preserve accuracy in numerical computations using variables, be sure that your intermediate and final results are not substantially smaller than 1.0.

Arithmetic Errors

Addition and subtraction of variable values are performed without any numerical errors; i.e., all 32 bits of the result are correct.

Multiplication and division are performed on normalized numbers. Multiplication gives 32 bits of precision in the internal result, and division gives 31 bits, but in both cases, the de-normalization process may discard some of the precision.

Multiplying two integers always gives the correct result, assuming that the result does not overflow the range -32768 through 32767.

Testing for Overflow

All of the ACL commands which operate on variables give an ACL Error 3 if a numeric overflow occurs during the processing of the command. To test whether overflow has occurred during a series of computations:

```
VC 0;  OE; (...perform the computations...)   VC 0;  OE;
```

The initial OE "clears" the error mechanism in case a previous ACL error was logged and not yet reported. The final OE sets variable 0 to zero if there have been no ACL errors, or 3 if there has been overflow. ***Note:*** Other values are also possible.

Some Possible Uses For Variables

Variables allow the Automove system to count items and to keep track of past events. Possible applications include:

- Stopping the move sequence after a certain number of failures or missing parts.
- Keeping track of whether a conveyor slot contains a workpiece.
- Remembering a Z-axis location for future use.
- Keeping track of occupied slots on a parts tray.
- Implementing a state machine (a series of operations based on the value of a "state variable", with different transitions, depending on previous state and current inputs.)
- Remembering several operator-taught locations for later use during a move sequence.
- Building a table of focal-find surface heights for interpolation.
- Storing manually-entered or automatically-measured information for workpiece skew correction via translation and rotation.
- Keeping track of the status of an automatic parts loader or other external device.

Examples

The following examples are not intended to be working move sequences, but are merely to give some ideas on how variables can be used.

Counters

The following three download sequences show how to use a variable to count how many times a particular process has been performed.

Table 29 - Single-Variable Download Sequences

ACL Commands:	Comments:
BD 0; VS 20, 0; ... ED;	Execute this in the morning to clear the counter in variable 20.
BD 1; ... V+ 20, 1; ... ED;	This sequence performs the process and adds 1 to the counter.
BD 3; OU"Total = ", 1; OV 20; ED;	Execute this at the end of the day to "print out" the total count on an attached terminal or printer.

The ", 1;" on the OU command causes its carriage return/linefeed to be suppressed, so that the output looks something like:

Total = 325

If the total count might exceed 32767, you may wish to divide the counter into two variables, as follows:

Table 30 - Two-Variable Download Sequences

ACL Commands:	Comments:
BD 0; VS 20, 0; VS 21, 0; ... ED; BD 1; V+ 20, 1 V= 20, 1000;	Clear two variables on Monday.
XI 2, 1; ... ED;	Add 1 to the first variable, then check if it is equal to 1000; if so, call Procedure 2 to "carry" into the other variable. Then, do the process.
BD 2; V+ 21, 1; VS 0, 0; ED;	Add 1 to the 1000's counter and set the 1's counter to zero.
BD 3; OU"1000's Total = ", 1; OV 21, 1; OU", 1's Total = ", 1; OV 20; ED;	Print both counters on Friday.

Again, the ", 1" on both OU and OV suppresses the carriage return, linefeed sequence, so the output looks something like:

1000's Total = 271, 1's Total = 758

which means that the procedure was performed a total of 271758 times.

Counting the Number of Retries

In this example, the workpieces are being fetched from a parts feeder. A microswitch attached to Digital Input 0 indicates "0" if a part was correctly fetched; otherwise the microswitch is "1". If the fetch fails three times in a row, the Automove will halt.

Table 31 - Counting the Number of Retries

ACL Commands:	Comments:
BD 0; XD 5, 0; ED;	Repeat Sequence 5 indefinitely until there is a failure.
BD 5; VS 44, 3; XU 6, 0, 1; ... ED;	Set variable 44 to the allowable number of failures. Call 6 repeatedly until a fetch succeeds. Then process the (correctly-fetched) part.
BD 6; ... XI 7, 1, 1; ED;	Attempt to fetch a part. If the fetch fails, call Sequence 7.
BD 7; V- 44, 1; V< 44, 1; XI 8, 1; ED;	Subtract 1 from the fail counter. If it reaches less than 1, call Sequence 8 to halt.
BD 8; AD; ED;	Halt (i.e., Abort Download Execution).

Timeouts and Flags

In this application, the Automove system is waiting for an external event to occur, for example, a workpiece arriving on a conveyor belt. If the workpiece does not arrive within a certain amount of time, the Automove is to turn off a heater. Assume that the event is indicated by a "1" on Digital Input 7, and the heater is activated by a "1" on Digital Output 2.

If the heater is turned off, the Automove clears a flag (i.e., changes the variable's value from 1 to 0). The next time the heater is needed, the Automove sees that the flag is clear, turns on the heater, and waits a few moments for it to warm up.

Table 32 - Timeouts and Flags

ACL Commands:	Comments:
BD 1; VS 18, 0; XD 2, 0; ED;	Variable 18 is the flag; here we set it to 0 to indicate that the heater must be turned on the first time. Then 2 is repeated forever to process parts on the conveyor.
BD 2; VS 16, 100; XU 3, 128, 128; V= 18, 0; XI 20, 1; ... ED;	Initialize the timeout variable (16) to 100. Repeat Sequence 3 until a part arrives. Test the flag. If the heater is off, call 20 to turn it on. Process the part.
BD 3; V= 16, 0; XI 21, 1, 1, 4; ED;	Test the timeout variable. If it is zero, call 21 to turn off the heater, Otherwise, call 4 to subtract 1 from the timeout and wait. Note: We actually call 21 many times after the timeout is 0, but in this case, it does not affect processing.
BD 4; V- 16, 1; WA 0.1; ED;	Subtract 1 from the timeout. Wait 0.1 seconds (so that the total timeout is 100 x 0.1 = 10 seconds).
BD 20; CD 4, 4; WA 1.5; VS 18, 1; ED;	Turn on the heater and wait 1.5 seconds for it to warm up. Set the "heater on" flag.
BD 21; CD 0, 4; VS 18, 0; ED;	Turn off the heater and clear the "heater on" flag.

Indexing and Parameters

Suppose that you have a Download Sequence which performs some action. In the middle of that action, it needs to move to a specified X,Y location, and then complete the action. You wish to have the operator teach 4 different locations, with the Automove saving the locations for future use. Then, the Download Sequence performs the action in each of these 4 locations.

In this example, we use an "index variable" -- a variable whose value indicates where to store the next location. It starts at 100, so that the X,Y locations will be stored in variables 100 and 101, 102 and 103, 104 and 105, 106 and 107.

Table 33 - Indexing and Parameters

ACL Commands:	Comments:
BD 0; VS 5, 100; XD 10, 4; VS 5, 100; XD 11, 4; ED;	Initialize the index (variable 5) to 100; call 10 four times to teach the locations, set the index back to 100; call 11 four times.
BD 10; OU"Please teach a point."; VC @5; OT; V+ 5, 2; ED;	Prompt the operator to teach a point. The "VC captures the taught location in this variable and the next one." @5 indicates that variable 5 shows which variable that gets the X value. The OT waits until the operator presses the Teach button, then "outputs" the X and Y values, which, in this case, are "captured" into two variables. Then, add 2 to the index for next time.
BD 11; VS 1, @5; V+ 5, 1; VS 2, @5; V+ 5, 1; XD 20; ED	Put an X value into variable 1. Put the following Y value into variable 2. Leave variable 5 pointing at the next X value. Call 20 to do the processing.
BD 20; ... MA @1, @2; ... ED;	Perform processing. Move to the location given by variables 1 and 2 to perform additional processing. (Variables 1 and 2 are "parameters" to Sequence 20.)

11 Appendix A Power Up Actions

At power up, the machine establishes default values for certain commands, but (unless there is an Autostart sequence) does not apply power to the motors or move them. The defaults are established as though the IN and ESC.R commands had been executed, plus the following:

- The Downloading mode is disabled, as though an ED had been executed.
- The microstep resolution is set to the default value. (See the RE command.)
- The present X-Y position is taken to be (0,0) microsteps (i.e., the Home position).
- The present Z position is taken to be 0.
- Arrow motion is unconstrained. (See the FH and FZ commands.)
- The XY motor switching drivers are on, as though a "PM 0;" had been executed. The optional Step Verify circuit is disabled. (See personality parameter 50 in Chapter 7.) Note that motor current turns on automatically, and Step Verify is enabled, as soon as any XY motion is attempted.
- The power level of the Z motor, if any, is zero.
- The non-volatile memory is checked for internal consistency. (See below.)
- The Autostart sequence, if any, begins executing. (See personality parameter 39 in Chapter 7.)

The ESC.!0: command (see Chapter 6) can simulate a true reset or power up in all software respects.

Non-Volatile Memory Consistency Check

At power up the Automove System checks its non-volatile memory for internal consistency. If any problems are found the System sends a "?" to the host, puts itself into the Emergency Stopped state, and sets a status bit that can be tested via the ESC.O command.

The non-volatile memory can become corrupt under a number of conditions; for example, turning the power off while downloading or while changing personality parameters; or unplugging a motor while the power is on and the Write Protect switch is down. Low-voltage "brownout" power line conditions can also cause this problem even if the Write Protect switch is up (i.e. "protected").

While the non-volatile memory is corrupt the machine may act erratically and may be unable to execute any ACL commands. To correct the problem you should enable the Write Protect switch; then send the ESC.!6: command to initialize the personality parameters and the ESC.!7: command to clear the download memory. You may also need to send ESC.!8: to erase the XY Linearity Correction Table. Then disable the Write Protect switch and turn the power switch off and back on. If the problem persists your Automove System may need service.

12 Appendix B Summary of Command Syntaxes

ACL Commands

Table 34 - Summary of Command Syntaxes

Command Code	Command
AA <x center>, <y center>, <angle>	Arc Absolute
AB [<flag>]	Anntibacklash Vectors
AC [<accel>]	Acceleration
AD	Abort Download Sequence Execution
AM [<mode>]	Arrow Mode
AP [<flag>]	Auxiliary Digital Port Select
AR <dx center>, <dy center>, <angle>	Arc Relative
AS [<flag>]	Auxiliary Speed
AT [<flag>]	Arm Timer
AZ <z>	Absolute Z Axis Move
BC [<re-use>]	Begin Continuous Path
BD [<id> [, <rept>]]	Begin Download
BP [<angle>]	Begin Pattern
CD [<new> [, <which>]]	Change Digital Outputs
CF [<xcal>, <ycal>]	Calibration Factors
CP	Clear Patterns
CR <x index>, <y index>, <x correction>, <y correction>	Correction
CS	Clear Stop
CZ [<rate> [, <mod> [, <halfflag> [, <accel>]]]]	Configure Z Axis
EC	End Continuous Path
ED	End Download
EP	End Pattern
ES <quotechar> [<char> [, <char> ...]] <quotechar>	Execute Escape Sequence
FH [<only if needed> [, <z first>]]l	Find Home Switches
FP [<mask>]	Front Panel Lockout
FZ [<only if needed>]	Find Z Home Switch

Command Code	Command
GD [<flag>]	Lower Retractable Height Sensor
GU [<flag>]	Raise Retractable Height Sensor
IN	Initialize
MA <x>, <y>	Move Absolute
MD [<count1>, <new1> [, <which1> [,<count2>,<new2> [, which2>]]]]	Mid-move Digital Outputs - Changes
MM [<count>, <new> [, <which>]]	Multiple Mid-move - Digital Outputs Changes
MN [<mode>,<value>[, <which>]]	Mid-move Digital Inputs - Response
MR <dx>, <dy>	Move Relative
MT	Move To Taught Point
MZ <numsteps>	Move Z (relative)
OA	Output Actual Position
OB	Output Buttons
OC	Output Commanded Position
OD	Output Digital Outputs - State
OE	Output Error Code
OF	Output Calibration - Factors
OG	Output Angle
OI	Output Identification
OL	Output Travel Limits
ON	Output Digital Inputs - State
OO	Output Origin
OP <selector>	Output Personality Parameter
OQ	Output Qualities
OR <x index>, <y index>	Output Correction
OS	Output Status
OT	Output Taught Point
OU <quotechar> [<char> [, <char> ...]] <quotechar> [, <suppress term>]	Output Literal String
OV <varid> [, <suppress term>]	Output Variable
OX	Output AP Value
OZ	Output Z Position
PD [<delay>, <new> [, <which>]]	Pre-move Digital Outputs - Changes
Command Code	Command

PE <selector>, <value>	Personality
PM [<level>]	Power Level of Motors
PS	Pause
PZ [<zlevel>]	Power Level of Z Motor
RE [<x res>, <y res>]	Resolution
SC <x>, <y>, <varid> [, <inverse>]	Scale
SO [<xorg>, <yorg>]	Set Origin
SP <x>, <y>	Set Position Counters
SR [<rate>]	Step Rate
ST <x>, <y>	Set Toggle Head Offset
SZ <z>	Set Z Position Counter
TD [<which>]	Toggle Digital Outputs
TL [<xmin>, <ymin>, <xmax>, <ymax>]	Travel Limits
VA <x1>, <y1>, <x2>, <y2>, <varid>	Compute Vector Angle
VC <varid>	Variable Capture
VL <x1>, <y1>, <x2>, <y2>, <varid>	Vector Length
VM [<mode>]	Vector Mode
VR <x1>, <y1>, <x2>, <y2>, <angle>, <varid>	Vector Rotate
VS <varid> [, <value>]	Variable Set
VT <varid>	Variable Test
V< <varid>, <value>	Variable Less Than
V= <varid>, <value>	Variable Equal To
V> <varid>, <value>	Variable Greater Than
V+ <varid>, <value>	Variable Add
V- <varid> [, <value>]	Variable Subtract/Negate
V* <varid>, <value>	Variable Multiply
V/ <varid>, <value>	Variable Divide
V& <varid>, <value>	Variable AND
V½ <varid>, <value>	Variable OR
V! <varid> [, <value>]	Variable NOT/Exclusive-OR
WA <wait>	Wait
WD [<delay>]	Wait After Digital - Outputs Changes
Command Code	Command
WN <value> [, <which> [, <timeout>]]	Wait For Digital Inputs

XD [<id> [, <rept>]]	Execute Download Sequence
XI <id>, <value> [, <which> [,<else id>]]	Execute If
XU <id>, <value> [, <which>]	Execute Until
XW <id>, <value> [, <which>]	Execute While
ZM [<count>,<new>[, <which>]	Z-axis Multiple Mid-move Digital Outputs

Escape Sequences

Table 35 - Escape Sequences

Command Code	Command
ESC.(or ESC.Y	Programmed On
ESC.) or ESC.Z	Programmed Off
ESC.! [<Reset Code>] :	Change System State
ESC.@ [(<Logical Buffer Size>) ;(<DTR Mode>)] :	Set Communications Configuration
ESC.B	Output Buffer Space
ESC.E	Output Extended Error
ESC.H [(Block Size or Xoff Thr>) : (<Enquiry Character>) : (<Acknowledge String or Xon Trigger String>)] :	Set Handshake Mode 1
ESC.I [(Block Size or Xoff Thr>) : (<Enquiry Character>) : (<Acknowledge String or Xon Trigger String>)] :	Set Handshake Mode 2
ESC.J	Abort Output Request
ESC.K	Abort ACL Command
ESC.L	Abort Buffer Size
ESC.M [(<Turnaround Delay>) ; (<Output Trigger Character>) ; (<Echo Terminate Character>) ; (<Output Terminator 1>) ; (<Output Terminator 2>) ; (<Output Initiator Character>)] :	Set Output Mode
ESC.N [(<Intercharacter Delay>) ; (<Immediate Response String or Xoff Trigger String>)] :	Extended Output and Handshake Mode
ESC.O	Output Extended Status
ESC.R	Reset Handshake
ESC.S [(<selector>) [; (<id>)]] :	Output Resource Allocation

13 Appendix C Summary of Error Codes

ACL Errors

These are reported by OE; see Chapter 3.

Table 36 - ACL Errors

Code	Error Description
0	No ACL error has occurred since the last OE.
1	An unrecognized mnemonic has been received.
2	The wrong number of parameters has been received.
3	An out-of-range parameter has been received.
4	The X, Y, or Z Home switch was not found within 32767 motor steps during an FH or FZ command.
5	The download memory has overflowed or the Write Protect switch is not enabled during a CR or PE command or BD...ED sequence.
6	A position overflow has occurred; an AA, AR, AZ, MA, MR, SP, or SZ has attempted to move the carriage or Z motor outside the travel limits.
7	Download Sequences have called each other to a depth greater than 12. (This maximum depth includes interrupts.)
8	Patterns have been nested to a depth greater than 12, or an EP has been executed with no pattern active.
9	A disallowed command has been executed during a Continuous Path sequence, or too many physical action commands have occurred between BC and EC.
10	A Continuous Path sequence has exceeded the maximum allowable total length, or the carriage has been moved via the front panel controls between BC and EC, or a BC with a nonzero re-use> parameter was disallowed.

Communication Errors;

These are reported by ESC.E; see Chapter 6.

Table 37 - Communication Errors

Code	Error Description
0	No communications error has occurred.
10	An output command was received while another output command was executing. The original command will finish normally; the second one will be ignored.
11	An invalid character was received after the first two bytes, ESC and period, of an escape sequence.
12	An invalid character occurred in the parameter list of an escape sequence. The parameter containing the invalid character and all subsequent parameters will be defaulted.
13	An ASC parameter was outside the range 0 through 255. The parameter is defaulted and all subsequent are ignored.
14	Too many parameters were received. Excess parameters are ignored until a colon or illegal character is received, or until another escape sequence is begun.
15	A framing error, parity error, or overrun error has been detected in incoming data.
16	The input buffer has overflowed. As a result, one or more bytes of ACL data have been lost, and an ACL error will probably occur.

14 Appendix D Summary of Personality Parameters

Table 38 - Personality Parameters

Personality Parameter/Description	Factory Default
1: Arrow Single Step Delay	3000
2: Arrow Slow Step Delay	100
3: Arrow Fast Step Delay	3
4: X Arrow Direction Swap	0
5: Y Arrow Direction Swap	0
6: First Find Home X Back Off Distance	250
7: First Find Home Y Back Off Distance	250
8: First Find Home X Seek Step Delay	2
9: First Find Home Y Seek Step Delay	2
10: Second Find Home X Back Off Distance	100
11: Second Find Home Y Back Off Distance	100
12: Second Find Home X Seek Step Delay	100
13: Second Find Home Y Seek Step Delay	100
14: Find Home Max X Distance	32767
15: Find Home Max Y Distance	32767
16: Find Z Home Max Distance	32767
17: Antibacklash Vector X Offset	-15
18: Antibacklash Vector Y Offset	-15
19: Default X Calibration Factor	1.0000
20: Default Y Calibration Factor	1.0000
21: Default X Microstep Factor	8.0000
22: Default Y Microstep Factor	8.0000
23: Default Travel Limit <xmin>	0
24: Default Travel Limit <ymin>	0
25: Default Travel Limit <xmax>	32767
26: Default Travel Limit <ymax>	32767
27: User Definable 1	0
28: User Definable 2	0

29: X Motor Direction	0
30: Y Motor Direction	0
31: Z Motor Direction	0
32: Default Acceleration	193
33: Default Step Rate	10000
34: Alternate Acceleration	193
35: Alternate Step Rate	20000
36: Digital Inputs Trigger Enable	0
37: Digital Inputs Function Select	0
38: Digital Input Sense	255
39: Autostart Sequence I.D.	-1
40: Z Axis Arrow Slow Step Delay	200
41: Z Axis Arrow Fast Step Delay	25
42: Z Axis Arrow Direction Swap	0
43: Default Arrow Mode	0
44: Default Z Axis Step Rate	100
45: Default Z Axis Modulus	32767
46: Default Z Axis Half Step Mode	1
47: Default Z Axis Acceleration	40
48: Default Find-Z-Before-XY-Home	0
49: Step Verify Control	0
50: Slip Response Download Sequence ID	-1
51: Step Verify Delay	500
52: First Find Z Home Back Off Distance	0
53: Second Find Z Home Back Off Distance	150
54: First Find Z Home Seek Speed	750
55: Second Find Z Home Seek Speed	100
56: Find Z Home Acceleration	10
57: Z-head Identification Code	128
58: Calibration Map Code (Square or Rectangle)	0
59: Paused Digital Outputs (range 0-255)	0
60: Interlock Control (range 0-255)	0

15 Appendix E Speed Considerations

System Elements

A motion control system consists of several elements: the host computer, the RS-232C interface, the Automove System's internal processors, and the physical motor/load system itself. In general, the act of moving the carriage at a controlled acceleration and slew speed takes so much time that the speed of the other system elements is not a concern.

For example, at 386000 microsteps/second/second acceleration and 10000 steps/second slew speed, assuming 1000 microsteps per inch, a 0.5-inch move takes approximately 76 milliseconds (0.076 second) to execute. This is broken into 26 milliseconds of acceleration, 24 milliseconds of slewing at 10000 steps per second, and 26 milliseconds of deceleration.

With these same parameters, a 0.1-inch move never reaches slew speed; it takes a total of approximately 32 milliseconds to execute, evenly divided into acceleration and deceleration. Longer moves, of course, take longer to execute.

At 9600 baud with a parity bit and one stop bit per character, it takes approximately 13 milliseconds to transmit the sequence MA12000,11000; over the RS-232C interface. Because of its input buffer, the Automove System can receive these characters while still processing previous commands. Delay within the input buffer itself is negligible.

If your application uses lots of medium-to-long vectors and arcs (say, one-inch and up) its speed is probably limited by the physical motion. If you can, try increasing the acceleration and slew rate via the AC and SR commands. If you need a few fast vectors mixed in with a lot of lower-speed vectors, use the VM command to flag the fast ones.

Arrange your program so that moves are not wasted; keep the total travel distance as low as possible.

On the other hand, if your application uses a lot of short vectors, its speed may be limited by the other system elements. Read on!

Precomputation and Overlap

The Automove System typically requires about 13 milliseconds of precomputation time for each true-speed MA vector, and a little more for MR vectors. (Arcs take considerably more.) Add about 5 milliseconds if the XY Linearity Correction mechanism is enabled, and several more if the pattern rotation angle is nonzero. This time begins when the System fetches the last parameter's delimiter (usually a semicolon ";") from the buffer; it ends when the carriage actually begins to move.

All but 3 milliseconds of the precomputation can be time-overlapped with the immediately preceding vector or other physical motion command. This overlap is possible because the Automove System contains two internal microprocessors: one for receiving and pre-processing data from the host, the other for controlling "physical" things such as the motors, the Digital Outputs, and the front panel.

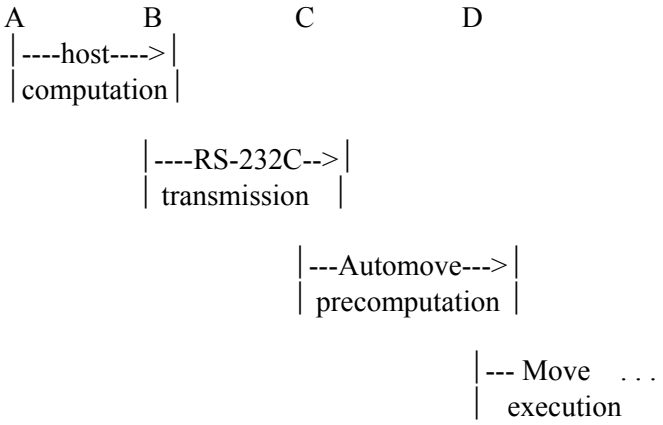
The precomputation time increases by about 0.5 millisecond for each character transmitted by the host during the precomputation. Due to the nature of the input handshakes (see Chapter 5) these characters tend to be transmitted in bursts. Thus any given vector or arc may take significantly longer to preprocess than its neighbors; this extra delay is very difficult to predict, but there is a way to eliminate most of the unpredictability. (See Tricks, below.)

The effects of the PD ("Pre-move Digital Outputs Changes") command occur after the precomputation and, hence, are precisely and repeatably time-related to the beginning of motion.

Throughput

When system vector and arc throughput is a consideration, there are two interesting special cases. The first, Case I, is a one-time startup delay at the beginning of a sequence of moves. It can be diagramed as follows, where the time axis is left-to-right:

CASE I -- STARTUP

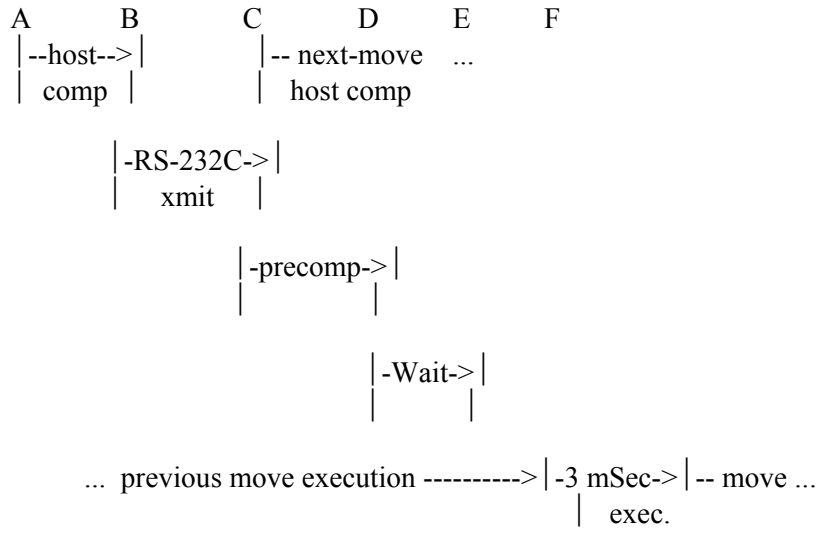


At time B the host finishes computing or looking up the coordinates of the next vector or arc, and begins transmitting them, in ASCII, to the Automove System. At time C the transmission is complete and the Automove System begins its precomputation. At time D the precomputation is complete and the carriage begins to move. The total time from B to D is simply the RS-232C transmission time plus the Automove precomputation time.

The host is free at time C to begin computing or looking up the next vector or arc. (This could be moved up to time B through the use of interrupt-driven host software.) The RS-232C channel is free at time C to begin transmitting the next command. The Automove System is free at time D to begin precomputing the next move.

This brings us to the second interesting case: the long term steady state where multiple moderate-length moves are being performed and the mechanical system is the slowest element. This case, Case II, can be diagramed as follows, assuming no computation/transmission overlap within the host:

CASE II -- STEADY STATE



This case is similar to the first, with the addition of the delay between time D and time E, during which the Automove System waits for the physical execution of the previous move to complete. The last 3 milliseconds of precomputation cannot occur until after E. As long as the typical move execution time exceeds the total from A through D, the system throughput is solely determined by the move length, the acceleration, and the slew stepping rate.

Of course, the typical application falls somewhere between the two cases. Time A-B is the least controllable element; if the host is very slow, each move is performed as a Case I move. Depending on host software, time A-B may depend on how long the host takes to produce each ASCII character from its internal numerical representation, how fast its disk drive is, and so on.

Time B-C depends on the baud rate, on whether a parity bit is being used, on the number of stop bits, and, perhaps most importantly, on how many "garbage" characters the host sends with each numeric parameter. If the host sends ten digits after the decimal point, or precedes each parameter with many blanks to make the "columns" line up as on a printer, throughput will suffer. For short vectors on most areas of the platen, the MR command requires fewer digit characters than MA, since the MR parameters tend to be smaller numbers. See Chapter 5 for recommendations on baud rate, parity, and stop bits.

Time C-D is fairly constant; it is usually in the range 10 through approximately 50 milliseconds. The worst case for vectors is where the following all happen at once: a burst of characters is being received during the precomputation for an MR command; and the MR parameters are negative and have digits after the decimal point.

If, for some reason, the host asks the Automove System to output something after each vector or arc, no overlap is possible. This is because the host must typically await the response before sending the next command. Also, most ACL output requests are not processed until the end of the preceding move. (Exceptions: OC, OE, OL, and OO do not wait.) The unsolicited "?" response to errors and Emergency Stops is intended to eliminate most of the need for this situation. See **Exceptional Conditions** in Chapter 5.

Tricks

If it is crucial that a particular short sequence of ACL commands be executed at a consistent rate each time, or with a minimum of delay between commands, the following trick can be used.

Using the PS command or the ESC.! command, place the Automove System into the Paused state. Then send the critical sequence of ACL commands, being sure that the first one is some sort of a "motion" command (or OA or OS or CD or WD ...) that does not execute if the System is Paused. The last command should be some sort of an ACL output command (NOT an escape sequence) that does not execute until the preceding command is finished, for example OS or OA.

The ACL commands will be stored in the Automove System's input buffer. After sending the last command the host should clear the Paused state, again via ESC.!, and the commands will then execute. Do not use the front panel during the sequence, or you will introduce delays. (To prevent this, the front panel can be locked out via the FP command.) When it receives the data from the final output command, the host is free to resume transmitting characters.

This programming trick prevents large uncontrollable delays due to data communications overhead and host computation time. However, internal coincidences will occasionally cause the Automove System to take a little more or less time to preprocess a given command.

16 Appendix F Choosing a Handshake

The decision about which handshake to use for a particular application must be based on a number of factors. Asymtek can only make general recommendations; the final decision must be made by the customer. If your host computer's application or driver ("BIOS") software already supports one type of handshake, the decision may be easy to make. Also, if you decide to use Asymtek's programming software, you do not need to make this decision.

From a software point of view, the Hardwired DTR handshake is the simplest to implement and the most reliable. This reliability stems from the fact that a noise glitch on the DTR line may cause a small speedup or slowdown of character transmission rate, but the Automove input buffer soaks up the difference since the buffer typically has room for several more characters at the time DTR goes False. The Xon/Xoff and Enq/Ack handshakes, on the other hand, fail catastrophically if a noise glitch renders a handshake character unrecognizable. (Software I/O timeouts within the host can provide recovery from some of these failures, but are messy to implement.)

The DTR handshake requires that the host RS-232C interface hardware or software be able to detect the state of some input signal; for example, Clear To Send. This signal, via cable wiring, is electrically connected to the Automove System's DTR (Data Terminal Ready) output. Thus, the DTR handshake precludes the use of a modem or a simple 3-wire RS-232C cable.

The host must be able to postpone data transmission while the DTR signal is False (Low). Some host RS-232C interfaces may have this ability built into the hardware. This postponement must not interfere with anything else that is going on within the host (parallel processes, for example).

If for some reason the DTR handshake cannot be used or if electrical noise is not a problem, the next best choice is probably the Xon/Xoff handshake. This handshake works well with either interrupt-driven or non-interrupt host software. It does not require the host to "chunk" the data or to know in advance how big the chunks will be (as does the Enq/Ack handshake). Xon/Xoff does not consume a lot of host processor time or a lot of data transmission time.

Finally, Xon/Xoff requires only 3 pins to be wired on the RS-232C connector: Transmitted Data, Received Data, and Signal Ground.

SAFETY, SHIELDING, AND GROUNDING CONSIDERATIONS MAY REQUIRE OTHER CONNECTOR PINS TO BE WIRED. ASYMTEK IS NOT RESPONSIBLE FOR THE WIRING OF ANY CABLE NOT MANUFACTURED BY ASYMTEK.

Some existing RS-232C cables may work with Xon/Xoff even though they do not work with Hardwired DTR.

The Enq/Ack handshake has most of the same advantages as Xon/Xoff but requires the host to separate the data into chunks (or blocks) of known maximum size. If this block size is set too small (for example, one character), excess handshake "chatter" degrades throughput.

Certain host software structures may be able to use Enq/Ack where they cannot conveniently use Xon/Xoff, and vice-versa. For example, is the host software able to test whether an Xoff character has been received, without actually hanging up waiting for one to arrive?

The Software Checking handshake should probably only be used as a last resort. Because of the large number of characters which must be transmitted in each direction, the handshake itself becomes the major burden on the RS-232C channel and on processor time at both ends of the channel. The host must chunk the data as for the Enq/Ack handshake. In a timeshared ("Big Computer") environment or a half-duplex modem environment where line turnarounds are slow, the Software Checking handshake would probably be far too slow.

There are other factors to consider in choosing a handshake. Is handshaking software already available? (Xon/Xoff is common.) Is vector throughput really important in your application? How will it be affected by the handshake? (See Appendix E Speed Considerations.) Does cable length dictate a low baud rate? If so, the number of characters transmitted for handshaking becomes more important.

Does an "antagonistic" I/O driver or host programming language prevent the application program from sending escape sequences? Does the driver send its own escape sequences which interfere with the ones sent by the application, or create output conflicts? (See ESC.E in Chapter 6.) Is the driver software accessible for modification?

17 Appendix G Example Program

Here is a simple BASIC program for controlling the Automove System from a Hewlett-Packard Series 200 computer. This program asks the user to enter 5 points with the TEACH button; then it "plays back" the 5 points.

In this version of the BASIC language, the **"!"** character indicates that the remainder of the current line is a comment. Variable names can have many characters; this program uses variables with the names Rs232, Ans$, Esc$, X, Y, Comma, and Dummy.

The OUTPUT statement is similar to PRINT, but sends its output to a specified port. ENTER is similar to INPUT, but gets its input from a port instead of the keyboard. The notation Ans$[m,n] means the m'th through n'th characters of the string named "Ans$", and Ans$[m] means the m'th through last characters of the string.

```
10   !  Automove example program.  Assumes
20   !  that the Hardwired DTR handshake is
30   !  automatically executed by the RS-232C
40   !  interface hardware.
50   !-----------------------------------------
60   DIM Ans$[30]    ! string for "answers"
70   DIM X(5),Y(5)   ! storage for taught points
80   Rs232=9         ! RS232C port select code
90   Esc$=CHR$(27)   ! the "Escape" character
100  OUTPUT Rs232;"IN;";   ! initialize Automove
110  OUTPUT Rs232;"FH;";   ! find Home switches
120  !-----------------------------------------
130  !  Now have the operator "teach" 5 points:
140  !-----------------------------------------
150  PRINT "Please enter 5 points with TEACH."
160  FOR I=1 TO 5
170    OUTPUT Rs232;"OT;"; ! request taught point
180    !  The following line hangs until the
190    !  user presses the TEACH button:
200    ENTER Rs232;Ans$    ! read the point
210    !  Check for "exceptional conditions":
220    IF Ans$[1;1]="?" THEN 540
230    Comma=POS(Ans$,",") ! where's the comma?
240    X(I)=VAL(Ans$[1,Comma-1])  !  X value
250    Y(I)=VAL(Ans$[Comma+1])    !  Y value
260    BEEP  !  give the user audible feedback
```

(Example program, cont'd.)

```
270    PRINT "Point #";I;" X=";X(I);" Y=";Y(I)
280  NEXT I
290  !-----------------------------------------
300  !  Now play the 5 taught points back:
310  !-----------------------------------------
320  PRINT "Press RETURN to play the points back."
330  BEEP
340  INPUT Dummy  !  wait for him to press RETURN
350  OUTPUT Rs232;"SR4000;";  !  step rate 4000
360  FOR I=1 TO 5
370    OUTPUT Rs232;"MA";X(I);Y(I);";";  ! vector
380    OUTPUT Rs232;"WA 0.5;";  !  wait 1/2 sec.
390  NEXT I
400  PRINT "All moves now in Automove buffer."
410  !-----------------------------------------
420  !  Now wait until the move sequence finishes
430  !  executing, just to see if an Emergency
440  !  Stop or motor slip has occurred:
450  !-----------------------------------------
460  OUTPUT Rs232;"OA;";   !  dummy request
470  ENTER Rs232;Ans$      !  await the answer
480  IF Ans$[1,1]="?" THEN 540
490  PRINT "Done -- no errors."
500  STOP
510  !-----------------------------------------
520  !  Routine to handle Exceptional Conditions:
530  !-----------------------------------------
540  BEEP
550  BEEP
560  BEEP
570  OUTPUT Rs232;Esc$;".E"; ! request Comm error
580  ENTER Rs232;Ans$        ! read error code
590  !  if error code has another "?", ignore it:
600  IF Ans$[1,1]="?" THEN Ans$=Ans$[2]
610  Err=VAL(Ans$)
620  IF Err=0 THEN 650
630    PRINT "Communications error ";Err
640    STOP
650  OUTPUT Rs232;"OS;";   ! request status byte
```

(Example program, cont'd.)

```
660  ENTER Rs232;Ans$      ! read the status
```

```
670  !  if status has another "?", ignore it:
680  IF Ans$[1,1]="?" THEN Ans$=Ans$[2]
690  Stat=VAL(Ans$)
700  IF (Stat DIV 512) MOD 2=0 THEN 730
710    PRINT "Motor slip has occurred."
720    STOP
730  IF (Stat DIV 16) MOD 2=0 THEN 760
740    PRINT "Emergency Stop has occurred."
750    STOP
760  OUTPUT Rs232;"OE;";   ! request ACL error
770  ENTER Rs232;Ans$      ! read the error code
780  PRINT "An ACL error ";Ans$;" has occurred."
790  END
```

18 Appendix H When it Doesn't Work: Debugging

When you think you've done everything right but it still doesn't work, you need to debug. Here we suggest a few ways of tracking down problems.

Nothing Happens

Check that the Automove power is on and that the RS-232C cable is firmly plugged in on both ends. If you supplied your own RS-232C cable, check the wiring. (See "The RS-232C Interface" in Chapter 5.)

Make sure that the host computer is actually sending characters to its RS-232C port. Check the host language statements which talk to the RS-232C port. Check the manual for the RS-232C interface to see if it needs to see a positive voltage on CTS (pin 5), DSR (pin 6) or DCD (pin 8) before it will send anything. If necessary, you can get a positive voltage from the Automove's RTS (pin 4). Try connecting a computer terminal, or another computer emulating a terminal, in place of the Automove System to verify that the computer is really sending characters, and that they make sense.

Check the baud rate and parity switches on the Automove System and on the host, to be sure both are in agreement. Make sure the host is sending 7-bit ASCII characters with one or two stop bits.

If you are using external motors, be sure the motor cables are correctly wired and are plugged in firmly. See the Automove System Operation Manual for information about wiring motor cables.

Be sure that you have not inadvertently pressed the STOP button or the PAUSE button on the Automove System, or closed the external Emergency Stop switch. (Look for a blinking light.) If the light in the STOP button is blinking, turn the Automove System off, then back on; if the light is still blinking see the next paragraph. If the PAUSE light is blinking, press the PAUSE button. If this does not turn the light off, either the button is broken or the button has been locked out via a software FP command from the host.

If the Emergency Stop light is blinking at power up then the internal non-volatile memory may be corrupt. See Appendix A for the procedure to fix this problem.

Caution: This erases your Download Sequences, personality parameters, and linearity correction table!

If the light in the PAUSE button stays on solidly (i.e. without blinking) then one of two things has happened: a) The FP command has locked out the PAUSE button and the button has subsequently been pushed. To fix this, execute an FP that doesn't lock out PAUSE. b) The STOP button is being pushed or an external Emergency Stop switch is currently actuated. To clear the situation you may have to take corrective actions such as closing a safety interlock switch. Check to be sure the Opto I/O cable is securely plugged into the I/O connector on the Automove back panel.

Be sure the machine is not in the Downloading mode; it gets in this mode when you send the BD command. To be sure it isn't, turn the power off, then back on.

Be sure you have not inadvertently put the machine in the Programmed Off state by sending ESC.) or ESC.Z.

Check the values of any personality parameters you have changed. If they are incorrect the Automove System may be physically disabled and give no other indication. The solution is to correct the bad values via the PE command, or simply reinstate all factory- set values via ESC.!6: . (See the caution, below!)

Connect a terminal (or a computer emulating a terminal) and see if the Automove System transmits a question mark "?" at power up. If so, one or more personality parameters have bad values. This can leave the System disabled. Consider sending ESC.!6: to reinstate the factory-set values.

Caution: Be careful -- some factory-set values may not be what you want!

Try running the Automove self test. Turn the power off. Then, while pressing both the Left Arrow and Right Arrow buttons, turn the Automove power on. The motors should run through a diamond- shaped self test pattern and the front-panel lights should go on and off.

It is a good idea to send the IN command at the beginning of each move sequence.

One Particular Command Doesn't Work

Use the OE command to see if the Automove System detected an ACL error. Check the syntax of the command and make sure the parameters are in range. If it is a motion command (MA, MR, AA, AR) make sure that the entire motion is within the travel limits; this is tricky with arcs. Use the OL command to find out what the current travel limits are.

Make sure all parameters are separated by spaces or commas, and are not run together. Make sure the command is terminated by a semicolon or by the mnemonic of the next command. Make sure there are no spaces in the middle of a number or between the minus sign and the first digit, and no semicolons or letters before the last parameter of a command.

Remember, if any parameter is bad, the whole ACL command is ignored. Also, any ACL error causes the System to ignore all characters until the next semicolon ";" or letter "A" through "Z".

If your programming language supports it, try sending the outputs to the computer's screen instead of to the RS-232C port. Again, a good way to find problems, if you happen to have a terminal or another computer, is to hook it up in place of the Automove System to see what your program is actually sending.

Remember, you are really debugging two programs: the host language (e.g. BASIC) program, and the sequence of ACL commands. Be sure that your host program is really executing the statements you think it is. Try putting PRINT statements in your loops and subroutines, putting a message on the screen to say "I got here and the value of X is 333".

If the Digital Outputs don't do what you want, look for an errant MD or VM command. Remember that MD affects all subsequent vectors, not just the first one.

If vectors are going fast when they should be slow, or antibacklash vectors have disappeared, look for a VM command.

Again, it is a good idea to send the IN command at the beginning of each move sequence.

The Carriage Crashes Into The Sides

Make sure your program does a Find Home operation at the start, or do a manual re-reference by pressing FAST with GO TO ORIGIN.

Make sure the travel limits have been set correctly for the size of your platen and the resolution; use the OL and TL commands. Check the applicable personality parameters with OP.

Make sure the resolution is what it should be, with the RE command. Check the applicable personality parameters with OP.

Be sure you are not sending an SP command which confuses the System about where the carriage is with respect to the platen.

The Move Pattern Is the Wrong Size or In the Wrong Place

Check the origin location with the OO command or by pressing the GO TO ORIGIN button.

Make sure the Origin hasn't been set to the wrong place by the BP command.

Check the calibration factors via the OF command. Check the applicable personality parameters with OP.

Make sure the resolution is what it should be, with the RE command. Check the applicable personality parameters with OP.

Be sure you are not sending an SP command which confuses the System about where the carriage is with respect to the platen.

Make sure your program does a Find Home operation at the start, or do a manual re-reference by pressing FAST with GO TO ORIGIN.

Be sure your program is sending the right values; try printing the numbers on the screen. Look for syntax errors described in **One Particular Command**, above.

It Works OK For A While, Then Goes Crazy

You may be getting a buffer overflow, due to an improper input handshake. Be sure you are using a handshake, either Hardwired DTR, Xon/Xoff, Enq/Ack, or Software Checking. See Chapter 5.

To check for buffer overflow, use the ESC.E command; see Chapter 6.

If you are using Hardwired DTR, be sure the Automove's Pin 20 is connected to the host's Pin 4 or Pin 5 (if the host is a DTE) and that either the interface hardware or the software stops transmitting when this handshake line goes low.

If you are using Xon/Xoff or Enq/Ack, be sure you have sent the correct escape sequences to set up the handshake; see Chapter 5 and 6. Also be sure your host software responds quickly to the particular Xoff character you have set up in the Automove System.

Be sure the host computer operating system or low-level driver software is not interfering with what the application program is trying to send to the Automove System. For example, the driver might not be passing through the escape sequences or handshake characters, or might be inserting Carriage Returns or Linefeeds in the middle of escape sequences.

Downloading Doesn't Work

Perhaps the rear-panel Write Protect switch is disabling non-volatile memory writes; flip it to the Down position to enable them.

The download sequences may be too large, or old download sequences (under different <id> numbers) might be occupying part of the memory. Use ESC.S2: to find out how much memory is available, or ESC.S5;n: to find out how much memory sequence "n" uses. Use ESC.!7: to be sure the memory is clear.

Part of the download memory may have become corrupt or inaccessible due to a reset or power failure during downloading. If you have erased sequences 0 through 255 (by sending BD 0; ED; BD 1; ED; ... BD 255; ED;) and some memory is still unavailable, use ESC.!7: to recover the lost memory.

You may have used the wrong download sequence number, or used the number twice, thus overwriting the old sequence with the new.

In general, it is a good idea to execute and debug your move sequence before downloading it. Some errors are not caught during downloading, but instead occur when the download sequence is executed. If you don't have a computer or terminal hooked up to the Automove System at that time, you won't see the "?".

Download Sequences Don't Seem To Execute Correctly

Try inserting debugging commands into the Download Sequences with the OU ("Output Literal String") command. Run the move sequence with a host computer connected to the RS-232C port, running a terminal emulator program.

The OU shows that execution reached that point. If you follow the OU with an OA, OE, OV, or other output command, you can get a better idea of what the move sequence is doing.

19 Appendix I Revision History

0.1	9/22/84	First edition.
0.2	10/4/84	FS becomes FH. Add VM and true-speed vectors. Allow parameters to be defaulted on many commands. MD's <new2> is never output before <new1>. Improve speed resolution. Travel limits default per platen size.
0.3	10/26/84	All time parameters are in seconds (affects CZ, WA, WD). Add the AB command, bit 2 of VM, and the <which> parameters of CD and MD. Allow TL to exclude the carriage position or the Origin position from the new limits (this also affects the GO TO ORIGIN and arrow buttons). The default Controller travel limits are 32767 instead of 32766.
1.0	11/9/84	Exceptional conditions send "?" instead of "@". Add the appendices. Add further description in Chapter 5. CZ has normal defaulting. CS no longer clears the Paused state. SO and TL parameters are in physical motor counts instead of Calibrated Units. Allow the carriage and Origin position to be outside the travel limits. Eliminate Error 5. Add the ESC.!: command.
1.1	2/5/85	First shipped with 1.2/1.3 ROMs. Add manual re-reference. Add AA, AR, FP, OI, OL. Add the No Reference status bit. Whole number parameters are rounded instead of truncated. FH does not reset the travel limits. OT waits until the button is pushed. The Output Terminator sequence defaults to CR-LF, not just CR.
1.2	3/8/85	First shipped with 1.3/1.4 ROMs. Add OF, OP, PE, RE, SP. Add Chapter 7: Changing the Personality. Add Appendix H: Debugging. Add non-volatile memory and personality parameters. MZ motion uses acceleration profiling. Change CZ parameter <delay> to <rate> and add <accel>.
1.3	3/19/85	First shipped with 1.4/1.5 ROMs. Add BD, ED, XD, the Execute Download Sequence front panel operation, ESC.S, and parameters 5, 6, and 7 on ESC.!.
1.4	6/21/85	First shipped with 1.6/1.5 ROMs. Add BP, CP, EP. Add ACL error 8. Add "ESC.S4:". Add debug suggestions for downloading. Reformat entire manual.
1.5	7/9/85	First shipped with 1.7/1.6 ROMs. Add personality parameters 29 through 33. Allow any AC up to 65535. Allow any SR up to 65535 for vectors, 10000 for arcs. AC and MD are diagonal measure, not long-axis.
1.6	8/15/85	First shipped with 1.8/1.7 ROMs. Add the MS command. Add negative parameter on the PM command. Allow fractional degree arcs. EXTERNAL MOTORS button does not do a RESET.
1.7	9/10/85	First shipped with 1.9/1.8 ROMs. Add personality parameters 34 and 35. The VM command's No SR bit affects both acceleration and step rate, and affects both arcs and vectors.
1.8	10/10/85	First shipped with 2.0/1.9 ROMs. Add Chapter 8. Add the AD, ON, TD, WN, and XI commands. Add personality parameters 36 through 39. Front-panel interrupts no longer abort the interrupted Download Sequence.

2.0	11/10/85	First shipped with 2.0/2.0 ROMs. Add the AM, AZ, and PD commands. Add bit 10 of the FP command. Add personality parameters 40 through 47. Decrease Z axis travel from 65535 to 32767 steps. Add Z axis Arrows. Add bit 7 of OS. Personality parameters are always "permanent". Decrease input buffer from 512 to 256 characters.
2.1	4/22/86	First shipped with 2.1/2.1 ROMs. Add the SZ, XU, and XW commands. Add the <only if needed> and <z first> parameters to FH and FZ. Add "ESC.S5;N:". Add personality parameter 48. Add front-panel Suppress Autostart and Z Arrow mode selection with Z Find Home and Z GO TO ORIGIN. Add non-volatile memory consistency check and bit 7 of ESC.O. Add bit 8 of OS. Change bit 10 of FP from EXTERNAL MOTORS to Interrupts. STOP and PAUSE are remembered if they occur while locked out via FP. Downloading and PE give an error if non-volatile memory is write-protected. OP sends 0 if <selector> is out of range. Exceeding maximum XD nesting depth halts Download execution. Change Z axis default to Half Step (PE 46 and CZ). "ESC.!6:" and "ESC.!7:" no longer go to power-up state. If personality parameter 14, 15, or 16 is zero the motor may travel forever. Delete the MS command.
2.2	5/23/86	First shipped with 2.2/2.2 ROMS. Add the Step Verify option: personality parameters 49, 50, and 51. Modify CS, FP, OS, PM, "ESC.!2:", and ESC.O.
2.3	7/2/86	First shipped with 2.3/2.3 ROMS. Add the MN and MT commands. IN sets Taught Point to 0,0. Maximum vector speed decreased from 10000 to 8000 full steps/sec. WA and WN are aborted by ESC.K.
2.4	10/29/86	First shipped with 2.4/2.4 ROMs. CS and "ESC.!2:" don't work while STOP switch is still actuated. Add bit 2 of 4.O. Add bits 8 and 9 of personality parameter 37 (binary-encoded and BCD-encoded Download invoke).
2.5	2/20/87	First shipped with 2.5/2.5 ROMs. Add the High- Resolution Z Axis option. RE does not set the No Reference status bit unless the resolution actually changes. Increase max Download Sequence <id> from 63 to 255. Decrease download chunk size from 63 bytes to 31 bytes, with consequent changes in ESC.S results. The manual Z Find Home sets the Position Change status bit.
2.6	8/10/87	Add the Z AXIS button and personality parameters 52 through 56 (FZ parameters). FZ is performed twice, with back-off and acceleration.
2.7	10/21/87	Add the CR, OR, and "ESC.!8:" commands (linearity correction). Change the factory default values for personality parameters 40, 41, and 54. Add Chapter 9.
2.8	5/9/88	Add the MM command.
2.9	6/28/88	Fix bug in MD/MM -- previous Digital Outputs not incorporated into MD/MM.
3.0	1/27/89	Add the BC, EC and "ESC.S6:" commands and ACL errors 9 and 10 (Continuous Path). Eliminate certain restrictions on arc speed and radius. Add a summary of the personality parameters as Appendix D, and rename the subsequent appendices. Change the factory default values of personality parameters 41, 53, 54, and 55. Decrease the minimum achievable Z axis acceleration (CZ command). The GO TO ORIGIN button sometimes performs a Find Z Home.

3.1	12/3/89	Add bit 10 of personality parameter 37. Eliminate the interdependence between the Digital Inputs and the Program Select thumbwheel and GO button.
3.2	7/11/90	Add ACL variables with the following ACL commands: SC, OV, VA, VC, VL, VR, VS, VT, V<, V=, V>, V+, V-, V*, V/, V&, V|, V!. Add Chapter 10. Add the <angle> parameter to the BP command, along with the OG command. Add the ES and OU commands. Increase the non-volatile memory from 8K to 16K and change from EE-PROM to battery-backed RAM; this speeds up downloading and eliminates the 10000-write limitation. Eliminate the 31-byte quantization (i.e. "chunks") of download memory. Increase the Linearity Correction table from 8 x 8 to 20 x 20. Add bit 3 ("No MN") to the VM command.
3.3	3/11/91	Internal changes. Speed up BP rotation.
3.4	4/22/91	Increase the maximum number of BC...EC physical action commands from 80 to 200. Add the <re-use> parameter to the BC command. Allow blanks between the ES or OU mnemonic and the first <quotechar> parameter. Add the "ESC.S7:" escape sequence in Chapter 6.
3.41	3/17/92	Add CK_XTAL subroutine to determine if PCB is new 4.9152 MHZ version and adjust time base for WAIT1MSEC subroutine.
3.42	7/27/92	Add AP command for two Aux I/O bits.
3.43	9/29/92	Fix CP bug that AP caused.
3.44	10/7/92	Add OB and OQ commands. Add AP0 to Esc .!5:. Fix front panel home problem with travel limits.
3.46	02/21/94	For REV 18 PCBs. Increase correction map to 50 x 50 for hubble.
3.47	03/29/94	For REV 18 PCBs. Add GU and GD commands. Add PE57. Remove 50 x 50 correction maps. Return to 20 x 20.
3.48	06/22/94	For REV 18 PCBs. Increase variables to 384 by decreasing available download memory.
3.5	10/7/92	Initial code updates for REV 19 PCBs. Implement paging scheme for download memory and increase download memory size. Add MC and MS ACL commands. Increase number of variables to 512. Increase calibration map maximum to 48 x 48.
3.51	10/5/93	Increase cal map to 50 x 50. Increase MM from 32 to 96. Add DJ command for fast MM. Add AF acceleration filter command.
3.52	12/23/93	Fix CP mnemonic table in JPARSER so that CP downloads work.
3.53	04/22/94	Fix Head and Tail Pointers by reducing variables to 384.
3.58	05/06/94	Make version number match JVGP. Add jerk control with JC command and circular buffer FIR. Allow both VG_FREQ for standard motion control or JC_FREQ for jerk control. Add retractable height sensor code from REV 18 PCBs 3.47 code. (Adds GU and GD commands. Adds PE57). Add JMP COMPUTE_KSLEW to CMD_JC to insure profile records are always correct. Add RCORT (PE58) to allow rectangular correction table.

3.60	11-06-95	Added AT, OX, ST, ZM, & * commands. Removed MC & MS commands. Modify MN commands for OR'ed inputs. Add PEP 59 for PAUSEd valve shutoff. Add PEP 60 for Interlock control. Add ESC.!35: to be used for ESC.!5,6,7,8 &9 Add ESC.Vn: and ESC.Wn;v: Remove need to press reset when new EPROMS installed.

20 Index