VELMEX, INC.
7550 State Route 5 & 20 Bloomfield, NY 14469-9389
FAX #: 716-657-6153 Tel.# 716-657-6151
http://www.velmex.com Email: info@velmex.com

8311,8312,8313,8314

USER'S GUIDE

8300 SERIES STEPPING MOTOR CONTROLLER/DRIVERS

January 29, 1985
Reformatted 1-16-98

Contents

8300 SERIES CONTROLLER/DRIVER

Introduction

The 8300 Series Control/Drivers are Microcomputer Controls for running one to four stepping motors from a host computer, programmable control, or terminal. Communication is through a full-duplex RS-232C Port. Data that is sent is interpreted by 8K of BASIC software. The BASIC Interpreter is the Rockwell International AIM-65 version developed by Microsoft. Velmex has developed 12K of custom firmware for interactive motor control. The combined 20K of ROM is contained on a 6502 Based Microcomputer along with 4K of RAM.

Programmable Input Features

* RS-232C Interface is a 3 wire connection (Tx,Rx,Gnd) configured as Data Terminal Equipment via a standard DB25 connector. The interface accepts ASCII character inputs which are echoed back to host.

* The Control sends a prompt ("^") when it is ready to accept an instruction or it can be polled for its status.

* Aux. RS-232C Port links multiple Controls together when more than four motors are needed, or if two or more motors must run at the same time. This port can also be used to send motor position, echo data received on the standard port, or to print any message to a printer or terminal.

* Baud Rate: switch settable from 50 to 9600 baud. Number of data bits and parity is switch settable.

* Absolute and Incremental Movements in Steps settable from -16,777,215 to (+)16,777,215 motor steps.

* Scaling: programmable divider to convert units from steps to inches, millimeters, revolutions, degrees, etc.

* Velocity: programmable from 16 to 4000 steps/sec. in 1 step/sec. intervals.

* Ramping: acceleration/deceleration is programmable from 3000 to 381,000 steps/sec./sec. in 3000 steps/sec./sec. intervals.

* Pauses: can be set from 0.1 to 838,860.7 sec.

* Base Speed: switch settable from 16 to 560 steps/sec.

* Outputs: four programmable outputs.

* Inputs: four individually readable inputs.

* Backlash compensation: a two line software routine will automatically finish every index, moving in the positive direction.

* Limit Switches: CW and CCW limits for every motor. Wired for direct connection to UniSlide "J" assembly. Limits can be used for homing.

* Software alterable motor direction designations: Normally a "+" direction signifies CW (UniSlide slider moves away from motor.) The user can reverse the designation of "+" in software to indicate CCW motion.

* User's Program in EPROM: Your final program can be externally burned into a 2716 or 2732 EPROM and inserted into an empty socket located inside the 8300 Control. The Control can be set to auto-load the program from the EPROM into RAM, and run.
For additional information, request document:
8300 PROGRAMS RESIDING IN EPROM.

Manual Input Features

* JOG: motors can be stepped one step CW or CCW from front panel.

* SLEW: motors can be run from 16 to 1500 steps/sec., CW and CCW from front panel.

* STOP/RUN: the Control can be put on "Hold" after a move ends, by pressing the STOP/RUN button located on front panel.

* PROGRAM INTERRUPTION (F1 Key): programs being executed can be interrupted.

Options

* Digital Displays: LED displays can be provided for each motor. The Readouts are front panel mounted. Scaling for different lead screw pitches is switch settable.

* Expansion: additional inputs, outputs, RAM expansion, and special programs in ROM are available.

Getting Started: We recommend that you read through this manual once before actual operation.

The AIM 65 BASIC incorporated in the 8300 series Control:

Every effort was made by Velmex to maximize the "command" set, "variable" names, and input/output capability of this simple to use, yet very powerful language. The user should refer to the AIM 65 BASIC LANGUAGE REFERENCE MANUAL by Rockwell International (included with this manual) for a complete description of this BASIC dialect.

The 8300 Control performs all the AIM BASIC Commands, Program Statements, Input/Output Statements, String Functions, and Arithmetic Functions except for ATN. The SAVE command performs a software reset which deletes the current program and all variables and reloads the BASIC Link program. The Control will be in the Manual Input Mode about 3 seconds after SAVE has been initiated. The USR statement is reserved for the 8300, therefore not available to the user. The NEW command should not be used. NEW makes the Control inoperable.

The Control converts the character "@" into the GOSUB statement. In the AIM 65 this character erases current line.

Inputs and Outputs: The words "print", "display", and "terminal" used in the AIM 65 Manual, refer to the 8300 Control's RS-232C Transmitter (Output); "keyboard" refers to the Control's RS-232C Receiver (Input); and "printer" corresponds to the Control's AUX. RS-232C Transmitter (Output). The words "printer on", "printer off" as used in the AIM Manual apply to the Control's Aux. RS-232C port. The printer status in the 8300 is normally "off".

Setup

* * CAUTION * *

*** HAZARDOUS VOLTAGE, DO NOT REMOVE CONTROL PANELS OR COVER ***

*** DO NOT CONNECT OR DISCONNECT MOTORS WHEN POWER IS "ON" ***

*** HIGH TEMPERATURE, CONTROL SHOULD BE KEPT AT LEAST 6 INCHES FROM ANY OBJECTS ***

*** AIR MUST CIRCULATE THROUGH AND AROUND CONTROL, DO NOT BLOCK AIR VENT IN BOTTOM OF CONTROL ***

*** NEVER USE IN AN EXPLOSIVE ENVIRONMENT ***

*** IN INDUSTRIAL ENVIRONMENTS, THE CONTROL MUST BE PROTECTED TO PREVENT METAL CHIPS FROM FALLING INTO IT ***

*** SEVERE ELECTRICAL DAMAGE MAY RESULT IF OBJECTS FALL INTO THE CONTROLS HEAT SINK ***

Before making any connections, Dip Switches inside the control should be set:

NOTE: Dip Switches should never be switched with the power "ON".

1. Remove the two plastic plugs on the side of the Control.

2. Set Dip Switches according to Figures 2, 4, and 5. Switch SW3 should have all switches OPEN. There is no need to set Switch SW2 if you don't have the Optional Digital Readouts.

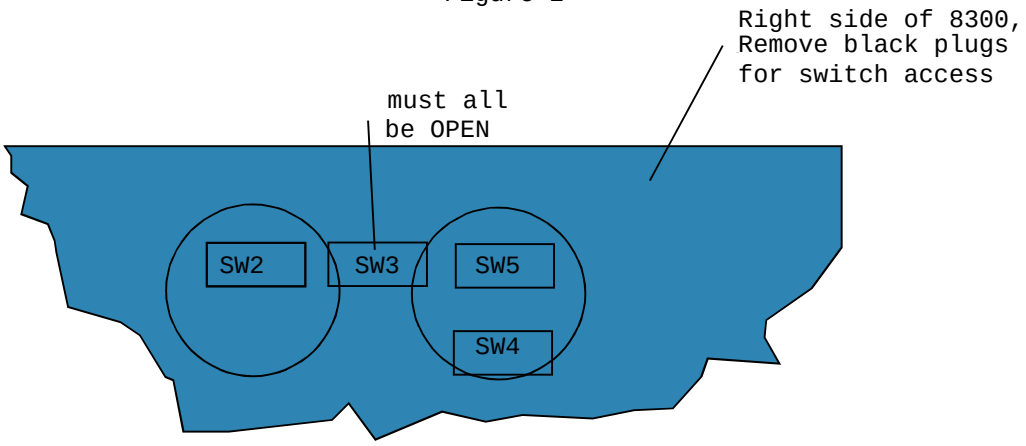
SW2

Lead Screw Pitch, Scaling for Displays

O=OPEN C=(CLOSED)

Divides Pulses to Display by	MOTOR 1		MOTOR 2		MOTOR 3		MOTOR 4		UniSlide LD Screws (Display in .001")	(Display in .01mm)
	1	2	3	4	5	6	7	8		
1	O	O	O	O	O	O	O	O	P5	Q2
2	C	O	C	O	C	O	C	O	P10	Q4
4	O	C	O	C	O	C	O	C	P20	
8	C	C	C	C	C	C	C	C	P40	

Figure 2



Dip Switch Locations

ADDENDUM:
April 25, 1986. Switch #8 of SW5 controls RS-232 echo. OPEN= 8300 echos characters back to host. "CLOSED"= no echo. 8300 does not send characters on the AUX. RS-232C when echo is off.

August 14, 1986. Switch #7 of SW5 controls the Auto Line Feed after a carriage return. OPEN= 8300 does not send a Line Feed. "CLOSED"= 8300 sends Line Feed after every carriage return.

SW4

RS-232C Parameters

O=OPEN C=(CLOSED)

BAUD Rate	1	2	3	4	5	6	7	8
50	O	O	O	O				
75	C	O	O	O				
110	O	C	O	O				
134.5	C	C	O	O				
150	O	O	C	O				
200	C	O	C	O				
300	O	C	C	O				
600	C	C	C	O				
1200	O	O	O	C	**< Use This Setting!**			
1800	C	O	O	C				
2400	O	C	O	C				
3600	C	C	O	C				
4800	O	O	C	C				
7200	C	O	C	C				
9600	O	C	C	C				

CAUTION: Above 1800 baud the Control may not be able to interpret data as fast as a host computer is sending. Refer to the following section on Direct and Indirect commands.

	5	6	7	8
Parity OFF	O			
Parity ON	C			
Parity ODD		O		
Parity EVEN		C		
7 Data Bits			O	
8 Data Bits			C	
1 Stop Bit				O
2 Stop Bits				C

Figure 4

SW5
Base Speed

O=OPEN C=(CLOSED)

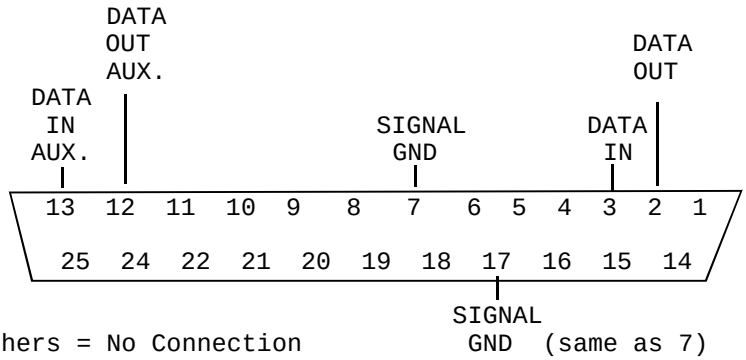
Base Speed (motor starting speed in Steps/sec.)	1	2	3	
16	O	O	O	
80	C	O	O	
160	O	C	O	
240	C	C	O	< Std. Setting, **< Keep At This Setting!**
320	O	O	C	
400	C	O	C	
480	O	C	C	
560	C	C	C	

Figure 5

3. Replace plastic plugs.
4. Connect cables to motors and limit switches.
5. Plug control into 120VAC outlet.
6. Hook up your computer or terminal as shown below: (Pins 12,13,17 are for networking Controls, refer to the following section on Aux. RS-232C.)

PC COM port			**8300 RS-232C**
25 pin	**9 pin**		
2	3	----------	3
3	2	----------	2
7	5	----------	7
(jumper 4&5, 6&20)	(jumper 4&6, 7&8)		

RS-232C Connector Looking at 8300 Front Panel



All Others = No Connection

Fig. 6

The 8300 acts as a DTE (data terminal equipment), it expects to be hooked up to and receive data from DCE (data communications equipment,) i.e. the Control expects data on line 3. If the data source is a DTE, then lines 2 and 3 must be crossed.

7. AUX. I/O connections: There are 4 programmable OUTPUTS. The OUTPUTS are unbuffered MOS level and should only be used with buffered-power-switching-equipment, such as manufactured by Opto-22, Gordos, and other manufacturers. **CAUTION:** these OUTPUTS will go "HIGH" momentarily when power is switched on.

There are 4 readable INPUTS that are normally "HIGH" (+5VDC). These are the INPUT and OUTPUT connections:

AUX. I/O Connection,
looking at 8300 front panel

(PIN Nos. are the same as INPUT and OUTPUT Nos.)

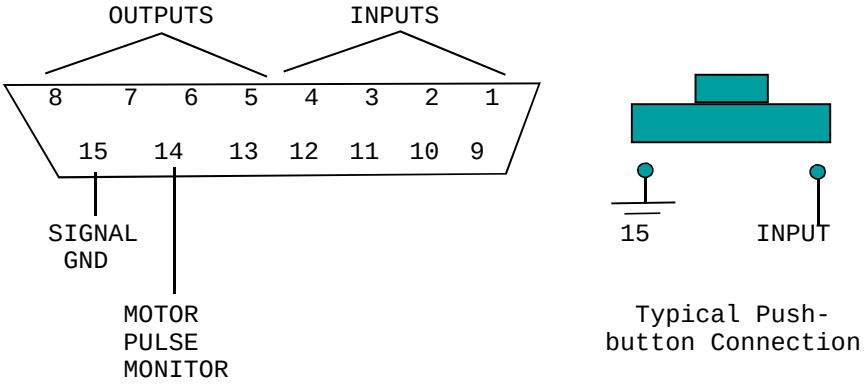


Fig. 7

The AUX. I/O is actually an 8-bit bidirectional port (Intel 8255A device.) Each line can be programmed as either an input or an output. This port has a Data Direction Register for specifying whether the peripheral pins are to act as inputs or outputs. When the 8300 is "powered-up" this register is automatically set to 147, which configures the port as shown above. The direction of the pins can be changed by altering the register using the BASIC POKE command.

This configures the port as all inputs:
POKE 34819,155

This configures the port as all outputs:
POKE 34819,146

This returns the port to normal:
POKE 34819,147

Pin 14 is a common Motor Pulse output. A negative-going pulse will appear here every time a motor is to move a step. This output is normally high (+5VDC), and will go low for 22 usec. before the next step. All outputs are capable of driving one standard TTL load only.

OPERATION

Manual Input Mode

NOTE: Turn on the host before or at the same time as the Control, otherwise a communications error may occur.

1. Turn ON the Control by pushing POWER switch.
The switch will "light", indicating "Power On".

2. The READY light will come ON in approximately 3 seconds. The motors can be run from the front panel.

3. Controls with optional Digital Displays should be "zeroed", by pressing the button next to the display.

4. MOTOR: Set motor to be run on the MOTOR thumbwheel.

5. SPEED and JOG/SLEW: To run the motor CW, (Slider on UniSlide moves away from motor) press the JOG/SLEW button labeled "+". A momentary closure of this button will JOG the motor one step. A maintained closure will SLEW the motor at 16 steps/sec. SLEW speed can be increased by pushing the SPEED switch upward while holding the JOG/ SLEW button down. The motor will accelerate, until the SPEED switch is released, up to a maximum of 1500 steps/sec.
NOTE: The 1500 steps/sec. can usually be obtained with zero load on the motor. Your actual top speed will be limited due to stalling, which is dependent on the motor type, friction, and the actual load. To decrease speed, push the switch down.
To run motor CCW, use the JOG/SLEW button labeled "-".

6. STOP/RUN: The STOP/RUN button is used to put a "running" program or a JOG/SLEW move on "Hold". When this button is pressed, the motor which is running will complete its move, the READY light will flash, and the control will remain on "Hold" until the STOP/RUN button is pressed again.
NOTE: The intended use of the STOP/RUN button is to give the user a convenient way to interrupt moves whenever a motor is running under program control.

7. F1: The JOG/SLEW (+) button functions as a program interrupt when the Control is operating in BASIC.
For more information on the F1 key, refer to page 301-1 of the *AIM 65 BASIC LANGUAGE REFERENCE MANUAL.*

RS-232C Input

1. Special Command Characters: The following characters have a special function:

CHARACTER	FUNCTION
E	Enters BASIC, the ON-LINE LED will "light", MANUAL INPUT is Disabled, and the Control sends the prompt "^" (ASCII 94).
&	Escapes BASIC and returns to Manual Input Mode. Any move in progress will be interrupted immediately. Also clears"ERROR" and "ON-LINE" lights. This character is not echoed to the host.
K	Interrupts current move or pause immediately, and resumes control to next command. NOTE: When motor is running at high speed, command "K" interruption may cause motor overshooting, resulting in loss of position.
D	Interrupts current move by decelerating the motor, at the set ramp, from its maximum velocity to a stop. Position register is adjusted to the current stopped position. Control resumes at the next command.
#	Polls Controller for its status. Control will send a "B" if it is "Busy" running a motor, or "^" if "Ready" for another command. Use of this command is optional since the prompt "^" is automatically sent after the execution of a move or pause.
S	Stops execution of a running program after a move or pause has completed. Prints "BREAK IN XXXX", where XXXX is the line number of the next statement to be executed. A "CONT" command can be used to continue running the program.
[	Opens communication to the Aux. RS-232C Port.
]	Closes communication to the Aux. RS-232CPort.
@	Same as a BASIC "GOSUB" statement. If no line number follows this command, the branch occurs to line "0" (Executes Move or Pause). The Control converts the "@" into a "GOSUB" when received from the host.

The preceding commands do not have to be followed by a Carriage Return.

ERROR: The "ERROR" light when lit, shows that either a Parity, Overrun, or Framing error occurred over the RS-232C . This light can be cleared by sending the "&" (escape BASIC) command.

Enabling Communication

2. BASIC Input Mode: To Enter BASIC send the letter:

E

The prompt "^" is sent to the host, "ON-LINE" light will come on, Manual Input will be disabled and the Control will be in BASIC Command Level.

BASIC Link Program: The BASIC Link program links the BASIC language with an assembly language subroutine. This assembly language routine performs the actual running of the motors. When the POWER is "turned on", the Link program below is automatically loaded into RAM. This program will always be used as a subroutine in your application programs. It is not necessary that the User understand how this routine works.

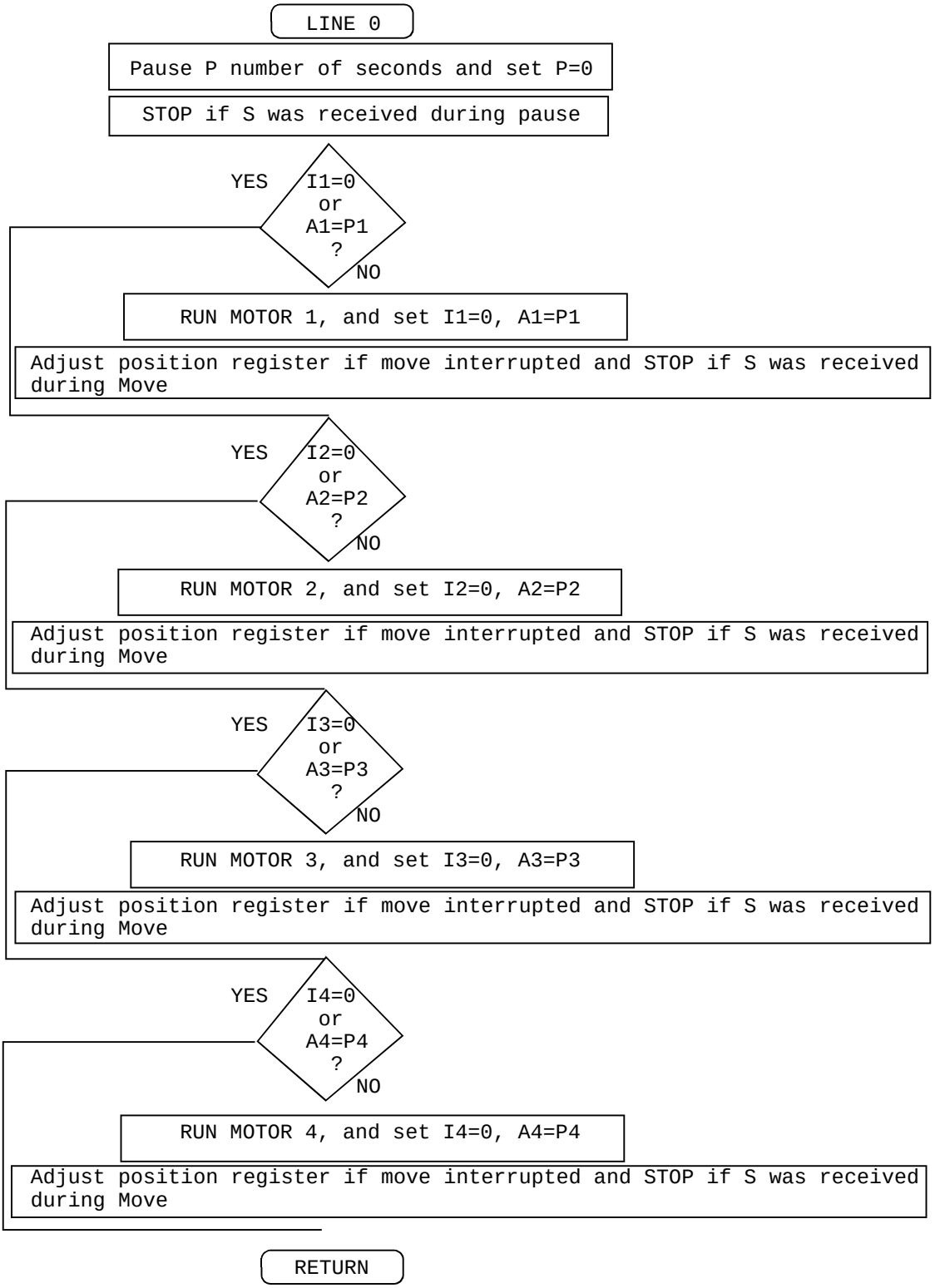
BASIC Link Program

(For Reference Only this is Built into the 8300!)

```
0 REM VELMEX 8314
28 IFP<>0THENI7=P*20:P=0:V5=0:A5=128:GOSUB55:IFFL$="S"THENSTOP
30 IFI1=0ANDA1=P1THEN36
31 IFI1=0THENI1=A1-P1
32 POKE228,1:I7=I1:I1=0:V5=V1:A5=R1:IFC1=0THENC1=1
34 I7=INT((I7/C1)+.5):I5=I7:IFI7=0THEN36
35 GOSUB55:P1=P1+(I5-V5)*C1:A1=P1:IFFL$="S"THENSTOP
36 IFI2=0ANDA2=P2THEN42
37 IFI2=0THENI2=A2-P2
38 POKE228,2:I7=I2:I2=0:V5=V2:A5=R2:IFC2=0THENC2=1
40 I7=INT((I7/C2)+.5):I5=I7:IFI7=0THEN42
41 GOSUB55:P2=P2+(I5-V5)*C2:A2=P2:IFFL$="S"THENSTOP
42 IFI3=0ANDA3=P3THEN47
43 IFI3=0THENI3=A3-P3
44 POKE228,3:I7=I3:I3=0:V5=V3:A5=R3:IFC3=0THENC3=1
45 I7=INT((I7/C3)+.5):I5=I7:IFI7=0THEN47
46 GOSUB55:P3=P3+(I5-V5)*C3:A3=P3:IFFL$="S"THENSTOP
47 IFI4=0ANDA4=P4THENRETURN
48 IFI4=0THENI4=A4-P4
49 POKE228,4:I7=I4:I4=0:V5=V4:A5=R4:IFC4=0THENC4=1
50 I7=INT((I7/C4)+.5):I5=I7:IFI7=0THENRETURN
51 GOSUB55:P4=P4+(I5-V5)*C4:A4=P4:IFFL$="S"THENSTOP
53 RETURN
55 POKE223,A5:A5=1:IFI7<0THENA5=0:I7=-I7
56 IFI7>16777215THENSTOP
57 POKE226,A5:IFI7>65535THENI6=INT(I7/65536):I7=I7-I6*65536
58 IFI7>255THENP=INT(I7/256):I7=I7-P*256
60 POKE220,I7:POKE221,P:POKE222,I6:P=0:I6=0:FL$=CHR$(USR(V5))
62 V5=PEEK(222)*65536+PEEK(221)*256+PEEK(220):IFA5=0THENV5=-V5
70 RETURN
```

Here is a flow diagram showing how this Link program works:

(For Reference Only this is Built into the 8300!)

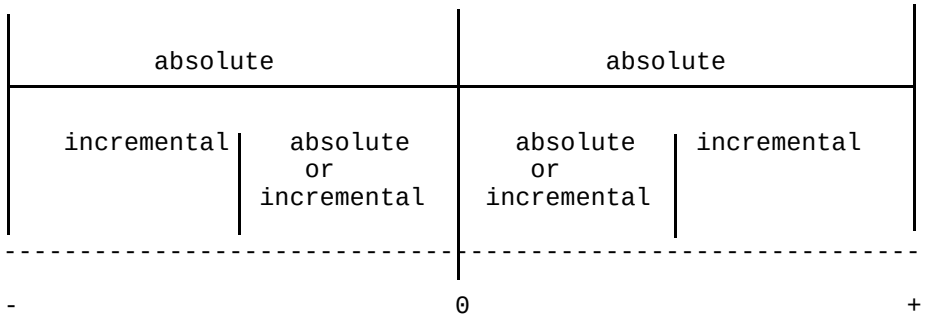


3. **BASIC Motor Variables:** Motor parameters are set by assigning values to the following variables.

NOTE: When the Control is turned on all variables are equal to zero. Variables are set to zero when a RUN or CLEAR command is used or when altering a line.

VARIABLE NAME	PARAMETER
I1	Incremental steps to move Motor 1. RANGE: -16,777,215 to (+)16,777,215 "+" being CW (UniSlide slider moves away from motor)."+" is CCW rotation of UniSlide rotary tables.
I2	Incremental steps to move Motor 2.
I3	Incremental steps to move Motor 3.
I4	Incremental steps to move Motor 4.
A1	Absolute steps to move Motor 1. RANGE: -16,777,215 to (+)16,777,215 steps. NOTE: Absolute zero is established at the current motor position when the power is turned"on", a CLEAR or RUN command is used, a line is altered, or if both A1 and P1 are assigned to zero.
A2	Absolute steps to move Motor 2.
A3	Absolute steps to move Motor 3.
A4	Absolute steps to move Motor 4.

A Representation of Absolute and Incremental Distance



C1 Conversion constant for Motor 1 which allows the user to work in units other than motor steps (i.e. inches, millimeters, degrees, revolutions, etc.). This constant is specific for the lead screw or rotary table employed, according to the table below. If no value is assigned to C1, units will be in steps.

Value to assign C1 (increment/step)	UniSlide lead screw or rotary table	Desired units for I1,A1,P1
0 or 1	all	steps(200/rev.)
0.005	WF & PF	inches
0.001	W2 & P5	"
0.0005	W1 & P10	"
0.00025	B & P20	"
0.000125	C & P40	"
0.02	K4 & Q4	millimeters
0.01	K2 & Q2	"
0.005	K1 & Q1	"
0.02	B4757TS (90:1)	degrees
0.025	B4872TS (72:1)	"
0.1	B4818TS (18:1)	"

C2 Conversion constant for Motor 2

C3 Conversion constant for Motor 3

C4 Conversion constant for Motor 4

V1 Velocity of Motor 1.
RANGE: 16 to 4000 Steps/sec.
in 1 Step/sec. increments.
Defaults to 4000 if set over 4000.

V2 Velocity of Motor 2

V3 Velocity of Motor 3

V4 Velocity of Motor 4

R1 Ramping (Accel./Decel.) of Motor 1.
RANGE: 1 to 127 (1 being 3000 steps/sec.^{2}, 2 being 6000 etc.
If no value set, defaults to 1)

R2 Ramping of Motor 2

R3 Ramping of Motor 3

R4 Ramping of Motor 4

The following are Position Indicating Variables and should never be set to any value by the user:

P1	Position of Motor 1. This is Accumulated Position computed in the BASIC Link program.
P2	Position of Motor 2.
P3	Position of Motor 3.
P4	Position of Motor 4.

Reserved Variables: The following variables are used by the BASIC Link subroutine as scratch variables and should never be used in your application programs:

A5
I5
I6
I7
V5

Calling the BASIC Link program: The BASIC GOSUB statement is used to branch to line 0 (start of BASIC Link program).
The BASIC Link program initiates execution of a move or pause.
The "@" character is equal to GOSUB.

Pause: Pauses can used by setting the variable "P" equal to the desired seconds to pause. The range is from 0.1 to 838,860.7 seconds.
The following example will produce a pause of 60.1 seconds (ends with a Carriage Return):

```
P=60.1:GOSUB
```

Since "@" is equal to GOSUB, this example is a shorter version of the above:

```
P=60.1:@
```

Indexing The Motors

(See examples on Page 31 and 32)

4. **Direct Commands:** Direct commands are one or more BASIC statements succeeded by a carriage return (<RETURN>). In this form they constitute a "line". Statements on the same line must be separated by a ":". Statements in the line are executed immediately following the <RETURN>. The maximum number of characters allowed in a line are 72 including the <RETURN>.

NOTE: The "@" command character occupies 5 character places in a line. The character is actually converted and stored as a GOSUB statement by the Control.

CAUTION: The use of direct commands determines the maximum baud rate at which the 8300 can interpret data sent from a host computer. When a line is sent from a host computer to the Control, the Control's BASIC requires a certain amount of time to interpret that line. The amount is dependent on the quantity and the particular statements that make up the line. At baud rates above 4800, the Control may lose data (the "ERROR" light will come on) if lines are coming in faster than the 8300 can interpret them. Therefore, to insure good communication use a low rate of 1800 or less. Only try the higher rates after all programing has been successful at the lower rates. NOTE: 1800 baud is recommended here in anticipation that you may use indirect commands discussed in the next section.

The following example, sent to the Control, indexes Motor 1 incrementally, 400 steps CW, at 1000 steps/sec. velocity, with a ramp of 6000 steps/sec./sec. (line is ended with <RETURN>). The 8300 sends the "^" prompt to the host indicating that the index has ended and the Control is ready for another command:

```
I1=400:V1=1000:R1=2:@
```

The "@" (GOSUB) causes a branch to line 0 of the BASIC Link Program and Motor 1 is indexed. The "RETURN" at line 70 causes a branch-back for another command. A ":" is used to separate statements on the same line. Spaces are not required between statements. See page 13 for definitions of the I1,V1,R1 variables.

Direct commands can be sent from the host singularly, as the following example illustrates (each command is followed by a Carriage Return):

```
R2=10
V2=2000
I2=-12000
@
```

To maintain the same velocity and ramp for succeeding indexes, it is not necessary to set these values for every index. Whenever the velocity and ramp are set, as in the previous examples, the velocity and ramp values are retained by the Control. However, these values will be set to zero when a RUN or CLEAR command is used, when altering a line, or when power is turned off.

The instruction below would index Motor 2, to absolute 375 steps, at the last set velocity and ramp value.

```
A2=375:@
```

This example would index Motor 2 back to its original starting position:

```
A2=0:@
```

Aux. Inputs and Outputs: The AUX. I/O can be addressed using the POKE, PEEK, and WAIT commands:

COMMAND	FUNCTION
WAIT 34818,1,1	Waits for Input 1 to go "LOW". Inputs are normally "HIGH" (+5VDC)
WAIT 34818,2,2	Waits for Input 2 to go "LOW".
WAIT 34818,4,4	Waits for Input 3 to go "LOW".
WAIT 34818,8,8	Waits for Input 4 to go "LOW".
POKE 34818,16	Turns "ON" Output 5 . (goes "HIGH" (+5VDC))
POKE 34818,32	Turns "ON" Output 6
POKE 34818,64	Turns "ON" Output 7
POKE 34818,128	Turns "ON" Output 8
POKE 34818,0	Turns "OFF" All Outputs
?(PEEK(34818)AND15)	Sends to the host "15" if all Inputs "HIGH", "14" if 1 "LOW", "13" if 2 "LOW", "11" if 3 "LOW", "7" if 4 "LOW", 0 if there all "LOW".

Position Report Back: Specific data can be sent to your computer or terminal by using the PRINT command.

This example sends the position of Motor 1 (ends with Carriage Return):

```
PRINT P1
```

Question Marks are equivalent to PRINT. You can use this statement instead of the above:

```
?P1
```

Sending A Program To The Control

5. **Indirect Commands:** Indirect commands are lines that are preceded by a line number. A line number is any integer from 0 to 63999. Lines 28 to 70 are occupied by the BASIC Link routine. The user must be careful not to unintentionally write over any of these lines. It is good practice to number the first line of your program "90."

Indirect commands make the Control truly programmable. A sequence of these commands constitutes a program. Entire programs can be sent by a host computer. These programs could be from text files residing in your computer, or a series of Print Statements from a BASIC program. A simple terminal is another method to enter programs. Programs can be typed in through a terminal connected to the 8300.

To execute a program, you would send the statement RUN "line number", where "line number" is the first line of your program currently in the Control's memory.

CAUTION: The use of indirect commands determines the maximum baud rate that can be used to communicate to the Control. The Control needs a certain amount of time to interpret and buffer each line sent to it. The time needed is dependent on the particular statements used within the line. Therefore, when indirect commands are to be used, the baud rate may have to be limited to a maximum of 1800. To insure good communication use a low rate of 1800 or less. Only try the higher rates after all programing has been successful at the lower rates.

Communication Errors

The "ERROR" light turns on if one or more of the following errors occurred:

a. Parity Error: caused by noisy or very long communication line.

b. Framing Error: due to baud rate or stop bit setting not the same on the host and Control, or turning on/resetting the host while Control is on.

c. Overrunning Error: caused by statements or lines that were sent faster than the 8300 could interpret them.

Example Programs

The following example programs were sent to the 8300 Control from a computer's text file, using communications software. (PC users use: “Terminal” program located in Accessories in Windows 3.1 or “HyperTerminal” with Windows 95, set baud to 1200, select parity, data bits, stop bits, and set flow control to None.

NOTE: Spaces and REM statements are used in these examples for clarity. These spaces and statements are not necessary.

Example 1

```
E
80 REM SET VELOCITY AND RAMPING
90 V1=900:R1=3:V2=900:R2=3
95 REM SETUP A LOOP OF 10, INCREMENT MTR2 200 STEPS
100 FOR Q=1TO10:I2=200:@
105 REM SETUP ANOTHER LOOP OF 10
110 FOR W=1TO10
115 REM INCREMENT MTR1 200 STEPS
117 REM TURN "ON" OUTPUT 6, PAUSE FOR 1.5 SECONDS
120 I1=200:@:POKE34818,32:P=1.5:@
125 REM TURN "OFF" OUTPUT 6 AND LOOP IF W IS NOT 10
127REM MOVE MTR 1 BACK AND LOOP IF Q NOT 10
128 REM MOVE MTR 2 BACK
130 POKE34818,0:NEXT W:A1=0:@:NEXT Q:A2=0:@
135 REM WAIT FOR A "LOW" ON INPUT 4 THEN GO AGAIN
140 WAIT34818,8,8:GOTO90
RUN90
```

The above program demonstrates an X,Y matrix operation. The "E" enables communication and the "RUN90" starts execution at line 90. The BASIC statement REM is utilized in this example to explain what each succeeding line is doing.

This the same program without REM statements:

```
E
90 V1=900:R1=3:V2=900:R2=3
100 FOR Q=1TO10:I2=200:@
110 FOR W=1TO10
120 I1=200:@:POKE34818,32:P=1.5:@
130 POKE34818,0:NEXT W:A1=0:@:NEXT Q:A2=0:@
140 WAIT34818,8,8:GOTO90
RUN90
```

Example 2

```
E
85 REM WAIT FOR "LOW" ON INPUT 1
90 WAIT34818,1,1
94 REM SET RAMPS,AND VELOCITY FOR X,Y
95 V1=1100:R1=4:V2=1500:R2=4:R3=3
97 REM SET SCALING FOR 5 PITCH LEAD SCREWS (P5)
100 C1=.001:C2=C1:C3=C1
105 REM GET ABSOLUTE X,Y FROM DATA
107 REM IF IT'S THE LAST COORDINATE, GO TO LINE 140
110 READ A1:READ A2:IF A2=0 THEN140
117 REM MOVE X,Y, THEN TURN "ON" PROBE
118 REM MOVE "Z" DOWN AT LOW SPD.,THEN TURN "OFF" PROBE
120 @:POKE 34818,16:A3=6.75:V3=500:@:POKE34818,0
125 REM MOVE "Z" BACK AT HIGH SPD. AND GO FOR ANOTHER X,Y
130 V3=1700:A3=0:@:GOTO110
135 REM MOVE X,Y BACK AND RESET "READ"
137 REM WAIT FOR INPUT 1, THEN RUN AGAIN
140 @:RESTORE:WAIT34818,1,1:GOTO110
145 REM ABSOLUTE COORDINATES IN INCHES FOR X,Y
150 DATA 1.125,1.5,3.125,1.5,3.125,4.75,1.125,5
160 DATA -.125,-5,2.125,7.771,-3.129,-8.340,2.001,3,1,.02,0,0
RUN90
```

The above program demonstrates a X,Y,Z probing operation, using READ and DATA statements to set A1 and A2. The "E" gets the Control "ON-LINE" and "RUN90" starts execution at line 90.

Here is the program without comments:

```
E
90 WAIT34818,1,1
95 V1=1100:R1=4:V2=1500:R2=4:R3=3
100 C1=.001:C2=C1:C3=C1
110 READ A1:READ A2:IF A2=0 THEN140
120 @:POKE 34818,16:A3=6.75:V3=500:@:POKE34818,0
130 V3=1700:A3=0:@:GOTO110
140 @:RESTORE:WAIT34818,1,1:GOTO110
150 DATA 1.125,1.5,3.125,1.5,3.125,4.75,1.125,5
160 DATA -.125,-5,2.125,7.771,-3.129,-8.340,2.001,3,1,.02,0,0
RUN90
```

Example 3

```
E
85 REM SET VELOCITY AND RAMP FOR MOTOR 1
90 V1=1000:R1=3
95 REM WAIT UNTIL A1 VALUE SENT FROM HOST
100 INPUT A1
105 REM RUN MOTOR 1 ABSOLUTE A1 STEPS
107 REM GO GET ANOTHER VALUE FROM HOST
110 @:GOTO100
RUN90
```

This a "prompt" example using the BASIC INPUT statement. The "INPUT A1" at line 100 requests a value for A1 from the host by sending a "?" as a prompt character.

Here is the same program without any remarks:

```
E
90 V1=1000:R1=3
100 INPUT A1
110 @:GOTO100
RUN90
```

Example 4

```
E
80 REM SET VEL.,RAMP,SCALE FOR P10 LD SCREW
90 V1=1200:R1=4:C1=.0005
92 REM SET INCREMENT VARIABLE TO .25 INCHES
95 INC=.25
97 REM USE GET TO GET VALUE FOR A$ FROM HOST
100 GET A$:IF A$="" THEN 100: REM LOOP IF A NULL
105 REM IF CHARACTER IS L THEN NEG. INCREMENT
110 IF A$="L" THEN I1=-INC
115 REM IF CHARACTER IS R THEN POS. INCREMENT
120 IF A$="R" THEN I1=INC
125 REM IF CHARACTER IS H THEN RETURN HOME
130 IF A$="H" THEN A1=0
135 REM INDEX MOTOR AND GET ANOTHER CHARACTER
140 @:GOTO100
RUN90
```

In this example the BASIC GET statement is used to get a single character from the host. If the host sends an "L" the UniSlide increments to the left, an "R" produces an increment to the right, and an "H" brings it back to home. NOTE: the quotation marks around the characters are not sent by the host.

This is Example 4 without remarks:

```
E
90 V1=1200:R1=4:C1=.0005
95 INC=.25
100 GET A$:IF A$="" THEN 100
110 IF A$="L" THEN I1=-INC
120 IF A$="R" THEN I1=INC
130 IF A$="H" THEN A1=0
140 @:GOTO100
RUN90
```

Limit Switches

Limit switch inputs are normally "HIGH" (+5VDC), active "LOW". When a limit switch is encountered, the motor halts immediately, control resumes at the next command, and the "limit flag" is set to a value of 1.

The PEEK command is used to check the status of the limit flag:

```
?PEEK(227)
```

If the above command prints a "1", the last motor "hit" a limit switch,
If the value printed is "0", no limit was "hit".

Limit switches may be used to establish a starting location ("home"). Using the UniSlide limits to establish a "home" yields a repeatability of about .001 inches.

The procedure to establish a "home" is:

1. Set a motor velocity of not more than 700 steps/sec.

2. Set an incremental or absolute move to a value the UniSlide cannot reach (i.e. run until limit switch is encountered).
WARNING: Make sure limit switch cables are properly attached.

3. Move "off" the switch a desired distance.

4. Establish this location as absolute zero by using a CLEAR statement, which sets all variables to zero.

This line is an example that moves Motor 1 to the "-" limit (UniSlide motor end) and backs "off" the limit 800 steps, where zero is then established:

```
90 V1=500:I1=-999999:@:I1=800:@:CLEAR
```

Backlash compensation: Mechanical errors resulting from changing motor direction can be eliminated by adding two lines to the BASIC Link program. The indirect commands below will run the motors 20 steps further only when a negative move is commanded, and then move 20 steps positive (i.e. all final moves are in the positive direction):

```
56 IFA5=0THENI7=I7+20
61 IFA5=0THENI7=20:GOTO55
```

These two lines will run the motors 20 steps further when a positive move is commmanded, and then move 20 steps negative (i.e. all final moves are in the negative direction):

```
56 IFA5=1THENI7=I7+20
61 IFA5=1THENI7=-20:GOTO55
```

Software alterable motor direction designations: Normally a "-" direction signifies CCW (UniSlide slider moves toward motor.) By adding the following indirect command (s) to the BASIC Link Program the motor (s) will move CW when requested to operate in the "-" direction.

This will reverse direction designation for Motor 1:

```
0 I1=-I1:IFA1<>P1THENA1=-A1
```

This will reverse direction designation for Motor 2:

```
5 I2=-I2:IFA2<>P2THENA2=-A2
```

This will reverse direction designation for Motor 3:

```
10 I3=-I3:IFA3<>P3THENA3=-A3
```

This will reverse direction designation for Motor 4:

```
15 I4=-I4:IFA4<>P4THENA4=-A4
```

User Defined Interrupt Characters: When a character is sent to the Control while a move or pause is in progress, that character will be assigned to the string variable "FL$". This variable is cleared to "0" at the start of every move or pause. This is how the conditional STOP in lines 35,41,46,and 51 of the BASIC Link Program responds to the special command character "S".

The Control will not assign the "#","K","D","&" command characters to FL$. You can setup your own relational tests for any character you wish, except the "#", "K","D","&".

The example line below would branch to line 500 of your program if an "I" was received during the last move:

```
200 IF FL$="I" THEN500
```

Aux. RS-232C

The Aux. RS-232C is used for the following purposes: 1. Linking 2 to 255 8300 Controls together, allowing communication with a host having just one RS-232 Port.

2. Sending messages or motor position to a printer or terminal.

3. View data that the Control normally echoes to the host, on a terminal or printer connected to the Aux. port.

Refer to figure 6 of the Setup section for proper connection to the Aux. port. The Velmex branching cable #8300-BC "brings out" the Aux. port via a standard 25D plug, for direct connection to the standard port of another 8300 Control.

NOTE: The Aux. RS-232C port is always at the same parameter settings (see Figure 4) as the standard port.

Linking Controls: Users that require more than four motors or must run motors simultaneously can link Controls together. To do this, Velmex branching cable #8300-BC or equal is required. One cable assembly is required for every additional Control. When Controls are linked, there is only one Control that is directly hosted by your computer. This Control is the host for a second Control and the second one is the host for a third, and the third a host for a fourth, etc.

Your computer or terminal can communicate to Control 2 through the Aux. RS-232C port of Control 1 if you enclose your data in brackets ("[ ]"). Control 1 must be "ON-LINE" and "READY" to communicate in this manner. If Control 1 receives a "[" from the host while it is "ON-LINE" and "READY", all three status lights on its front panel will go off, indicating it is in a "relay mode." In this "relay mode" all characters Control 1 receives will be relayed directly to Control 2. Data sent by Control 2 to Control 1 will be relayed back to your computer. The "relay" mode is terminated by a "]."

For example, this would enable and run Motor 1 of Control 2, then return control to Control 1:

```
[ E V1=900:R1=3:A1=4000:@ ]
```

Brackets can be "nested" up to 255 deep.

This would enable four Controls linked together:

```
E[E[E[E]]]
```

A Control can be programed to be a controller for other 8300s "down line" of it. The PRINT! statement is used to do this. Data following a PRINT! statement will be sent out the Aux. port.

This example sends the data between the " " to a second Control when Line 150 is executed:

```
150 PRINT!" V1=1300:I1=3001:@ "
```

Here is the same line in a shorter form:

```
150?!"V1=1300:I1=3001:@"
```

This line would index Motor 1 of a third Control:

```
150?!"[V1=1300:I1=3001:@]"
```

This line would index Motor 1 of a fourth Control:

```
150?!"[[V1=1300:I1=3001:@]]"
```

When Control 1 initiates the index of a motor of Control 2, as in the previous examples, Control 1 needs to know when Control 2 has completed the move before a new command can be sent. Control 1 must look for the "^" prompt (ASCII 94) that Control 2 sends when it finishes the index. This prompt will appear at Control 1's Aux. port receiver. Control 1 must use a "PEEK (35584)" command to access its Aux. port receiver.

This line will execute over and over until the "^" appears at the Aux. port:

```
170 IF PEEK(35584) <> 94 THEN 170
```

Printing Messages: Messages and data can be printed to a printer or terminal connected to the Aux. port.

This line prints "END OF TEST" to a printer:

```
999 ?!"END OF TEST"
```

This line prints the position of Motor 1:

```
320 ?!P1
```

Viewing the Controls Transmission: All communication that normally would go from the 8300 to the host can be diverted to the Aux. RS-232C port. This is accomplished by changing the "Out Flag register" (42001) from 0 to 128.

This diverts everything the Control transmits to the Aux. port:

```
POKE42001,128
```

This returns transmitting to normal:

```
POKE42001,0
```

For a table of ASCII character codes and complete information on the BASIC used in the 8300 Control, refer to the *AIM 65 MICROCOMPUTER BASIC LANGUAGE REFERENCE MANUAL* published by Rockwell International.
A copy of the AIM65 BASIC manual is included with the Control.

TROUBLESHOOTING PROCEDURE

SYMPTOM	POSSIBLE CAUSE	CORRECTIVE ACTION
POWER light does not "light" when Control is switched "on."	Blown fuse.	Check fuse located on back of Control.
Motors(s) do not operate in Manual mode.	MOTOR select thumbwheel not set.	Set MOTOR thumbwheel to a valid motor number to be run.
	Limit switch (es) in closed position.	Check limit switches for proper action.
Motors(s) do not operate and READY light is flashing.	STOP/RUN button was pressed.	Press the RUN/STOP button to release the Control from "hold."
Control does not come on-line when sent "E".	RS-232C not connected properly.	Trace Transmitted Data, Received Data, and Signal Ground wires from your computer to the 8300 Control.
	Your computer or terminal is not sending upper case letters.	Transmit only upper case letters. The Control will not respond to lower case.
	Your computer may require a high on its Data Terminal Ready (DTR) line. The 8300 does not implement the RS-232C control lines.	Check with the manufacturer of your computer to see if the DTR line must be artificially pulled high, or if it can be disabled in software.
Control does not come on-line when sent "E" and the ERROR light "lights."	RS-232Cparameters not set properly.	Match the RS-232C settings on the 8300 to those of your computer or terminal.
Your computer does not receive data from the Control.	Your computer may require a "high" on its Request To Send (RTS) line.	Check with the manufacturer of your computer to see if the RTS line must be artificially pulled high, or if it can be disabled in software.

SYMPTOM	POSSIBLE CAUSE	CORRECTIVE ACTION
Motor does not run when sent commands to index.	Control is not on-line (BASIC Input Mode).	The character "E" should be sent to put the Control on-line.
Motor operates in Manual mode, but not under computer control.	Velocity variable, or steps-to-move variable not set to a proper value.	Make sure you set the motor variables before executing the index.
	Line(s) accidentally deleted from the BASIC Link program.	Toggle power to reset the Control, or use SAVE to perform a software reset.
Motor does not run when sent commands, and ERROR light comes on.	Baud rate is too high, causing character overrunning.	Set baud rate to 1800 or less.
	Syntax error in command(s) sent, causing character overrunning.	Review the commands and statements you are sending for accuracy.
	Line(s) exceed 72 characters.	Keep lines under 72 characters (lines end with a <RETURN>).
	A program with Direct and Indirect commands is dumped to the 8300 Control, causing overrunning.	When dumping a program(more than one line) to the Control, use only Indirect commands.
A program is sent to the Control, the READY light goes off, and nothing appears to be happening.	The Control has been set for a long pause.	Send a "K" to the Control to interrupt pause or turn off power to restart Control.
	The 8300 is in an infinite loop.	Hold the F1 button down on Control's front panel, or turn off power to restart Control. Check your program for accuracy.

SYMPTOM	POSSIBLE CAUSE	CORRECTIVE ACTION
Motor stalls, it does not move at all.	Base speed too high.	Try a lower base speed.
	Inertia in system too high, or mechanism has seized.	Hand operate the system to locate any binding. A larger motor, or a different ratio in the mechanism may be required.
Motor stalls, after rotating slightly.	Acceleration too high.	Use a lower ramp rate.
	Motor cannot overcome friction or load.	Check mechanism for ease of movement. A larger motor may be required, or your load will have to be reduced or counterbalanced.
Motor stalls before reaching maximum velocity.	Motor torque decreases as its velocity increases.	Reduce velocity setting of motor.
Motor or system resonates (vibrates loudly)	The motor velocity is the motor or system's natural resonant frequency.	Increase or decrease speed to avoid resonance points. A damper or flywheel added to the motor shaft or lead screw may dampen the resonance.

SPECIFICATIONS

Functional

Packaged Controller/Driver, using Microcomputer control of stepping motors. One to four motors, one at a time.
Interactive limit switch inputs.
Four User inputs and four User outputs (TTL).
Programing through full-duplex RS-232C.
Manual control at front panel.
"ERROR", "ON-LINE", "READY" status lights.

Motor Compatibility

200 step per revolution, 10 amp max., stepping motors.
The Superior Electric M061-FD08, M062-FD03, M092-FD08 have a maximum velocity of 1500 steps/sec.
The M062-FD09, M091-FD09 will operate above 3000 steps/sec.
Note: Above max. speeds are for unloaded conditions .
Actual speed is dependent on friction and load.

Physical

Weight: 19.5 lbs.

Height: 8.5 inches

Width: 11.2 inches

Depth: 14.0 inches

Electrical Requirements

120VAC 60Hz, 350 watts

Environmental

32^0 to 120^0 F
Convection cooling

```
10 REM GWBASIC  EXAMPLE For sending Direct Commands to 8300
15 REM 2-17-92  /Velmex inc./ M.L.E.
20 REM ##############################################
30 REM #   Initialize COM port and put 8300 On-line #
40 REM ##############################################
50 REM Open COM1 port as file #1, 1200 baud, no parity,8 data bits,1 stop bit
60 OPEN "COM1:1200,N,8,1,CS0,DS0" FOR RANDOM AS #1
70 REM Put 8300 on-line by sending E
80 PRINT #1, "E"
90 REM Go get ^ out of COM1 input buffer that came from 8300
100 GOSUB 300
110 REM ##############################################
120 REM Index Motor#1 400 steps at 1000 sps and wait until Index done
130 PRINT #1, "V1=1000:I1=400:@"
140 GOSUB 300
150 REM Your routine for end of Index would go here!
160 REM Index Motor#1 to zero at 1000 sps and wait until Index done
170 PRINT #1, "V1=1000:A1=0:@"
180 GOSUB 300
190 REM Your routine for end of Index would go here!
200 REM Take 8300 off-line and end
210 PRINT #1, "&"
220 PRINT "End of Example Running 400 steps (2 revs) forward and back"
230 END
240 REM This routine waits until a  ^  is found in the COM1 port buffer
300 C$ = INPUT$(1, #1): IF C$ <> "^" THEN 300
310 RETURN
```

```
' QuickBASIC 4.5 EXAMPLE For sending Direct Commands to 8300
' 7-28-91 /Velmex inc./ M.L.E.

' ##############################################
' #   Initialize COM port and put 8300 On-line #
' ##############################################
' Open COM1 port as file #1, 1200 baud, no parity,8 data bits,1 stop bit
OPEN "COM1:1200,N,8,1,CS0,DS0" FOR RANDOM AS #1

' Put 8300 on-line by sending E
PRINT #1, "E"

' Go get ^ out of COM1 input buffer that came from 8300
GOSUB Waitprompt
' ##############################################


' Index Motor#1 400 steps at 1000 sps and wait until Index done
PRINT #1, "V1=1000:I1=400:@"
GOSUB Waitprompt

' Your routine for end of Index would go here!

' Index Motor#1 to zero at 1000 sps and wait until Index done
PRINT #1, "V1=1000:A1=0:@"
GOSUB Waitprompt

' Your routine for end of Index would go here!

' Take 8300 off-line and end
PRINT #1, "&"
PRINT "End of Example Running 400 steps (2 revs) forward and back"
END


'This routine waits until a  ^  is found in the COM1 port buffer
Waitprompt:
DO
        C$ = INPUT$(1, #1)
LOOP UNTIL C$ = "^"
RETURN
```