HP-P003-1

SUPER VEXTA®

SC8800/SC8800E
Stepping Motor Controller

RS-232C Compatible

Operating Manual

Contents

Table of Contents

List of Figures

Introduction

The Oriental Motor (OM) SUPER VEXTA® SC8800/SC8800E programmable pulse generators are specifically designed to be RS-232C compatible stand alone stepping motor controllers. They are optimized for use with the Oriental Motor SUPER VEXTA® stepping motor and driver packages. They may also be used with drivers from other manufacturers that accept TTL level step and direction inputs.

Figure 1 shows a block diagram of the stepping motor control system.

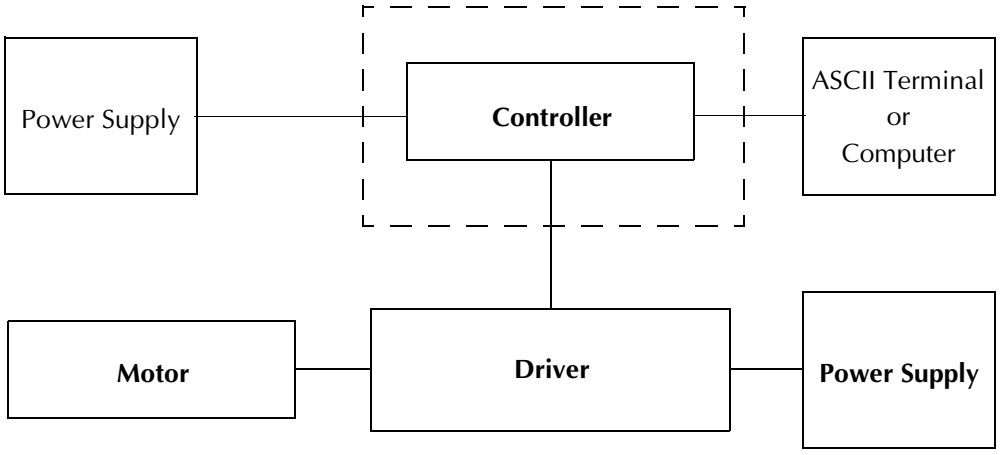


Figure 1 Block Diagram of Stepping Motor Control System

Model SC8800E differs from model SC8800 by having an encoder input which is used for position verification. Both models include a built-in control language which, with appropriate programming, allow the units to:

- Operate as stand alone controllers in open or closed loop configurations
- Be managed from a computer, ASCII terminal or stand alone PLCs
- Perform as application specific control devices with embedded program management

This comprehensive manual covers three major areas:

- Installation procedures
- Operation techniques and examples
- Control Language Programming References

The manual instructs you in precautions and proper practices for using this equipment safely.

The SC8800/SC8800E programmable controller includes the following components:

- SC8800/SC8800E controller (shown in Figure 2)
- Dual ended serial cable (Part # RS2740)
- 37 pin DIN connector and housing
- This manual (Part # HP-P003)

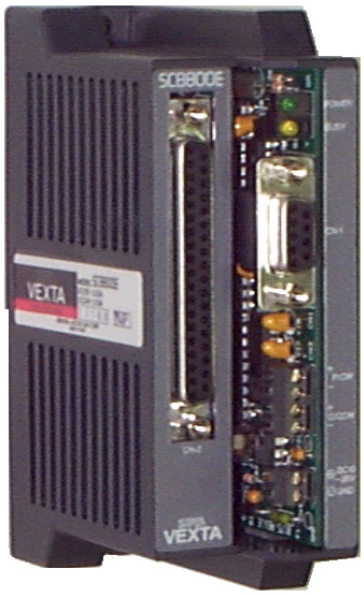


Figure 2 The SC8800/SC8800E Controller

Inspect the components for shipping damage. Report any damage to the carrier and to Oriental Motor USA. Notify Oriental Motor if any components are missing or incorrect.

CAUTION Do not install or apply power to any equipment that is damaged.

Accessing the Quick Start Guide

A Quick Start Guide is in Appendix A for users who are experienced in motor control systems and are cognizant of the electrical and mechanical precautions that must be observed. This Guide instructs the user in quickly connecting the SC8800/SC8800E controller to:

- A stepping motor driver
- A power supply
- A terminal or computer

It then directs the user into the first stages of using the interactive and programmable features of the unit.

System Configuration

A block diagram of the SC8800/SC8800E controller in a system configuration is shown in Figure 3.

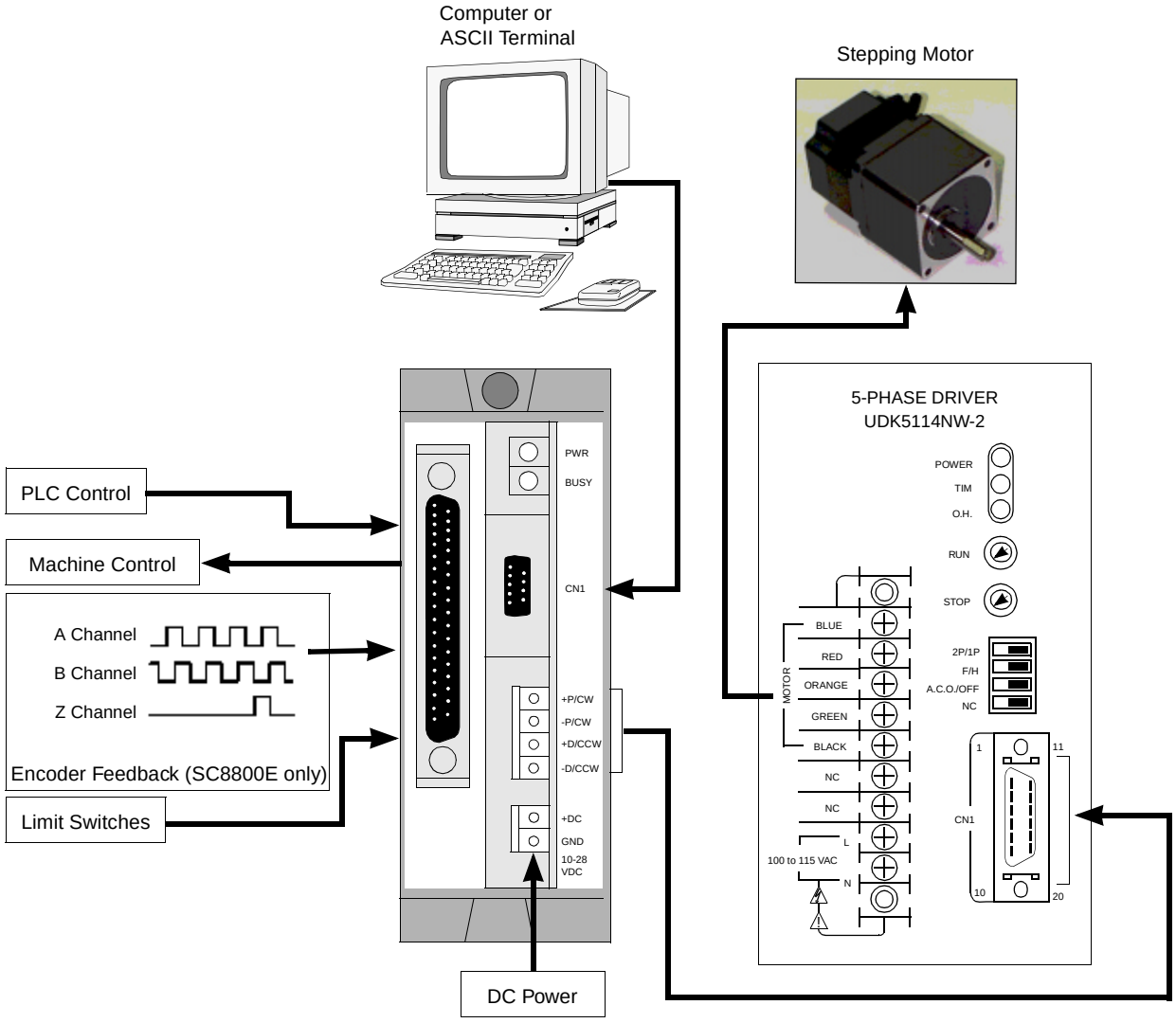


Figure 3 System Configuration

A description of how to connect the controller to a driver and motor, as well as to external control devices (i.e. PLC's, encoders, limit switches), is included in this section.

Installation

Installation areas for the controller (and associated motor and driver) should:

- Be free from dust, oil mist, salt or corrosive gas.
- Be free from excessive vibration or shock.
- Have an ambient temperature range between 32 to 104 °F (0 to 40 °C).
- Have humidity not exceeding 95%, noncondensing.
- Have at least one inch (25 mm) of open space between the controller and adjacent items.

Special considerations:

- Prevent contact between the controller and conductive material such as metal filings or pins.
- Do not store or use in direct sunlight.

Electrical Noise

Do not locate the controller near significant sources of electrical noise, such as high voltage lines, high voltage machines or switching-type power units. If this arrangement is not possible, insert noise filters in the source power lines or connect the controller to a separate circuit.

Mounting the Controller

The controller is designed to cool naturally by convection. Figure 4 shows the controller mounting hole positions and dimensions. When fastening the mounting hardware, take care not to over torque the screws (bolts) to avoid cracking the controller housing.

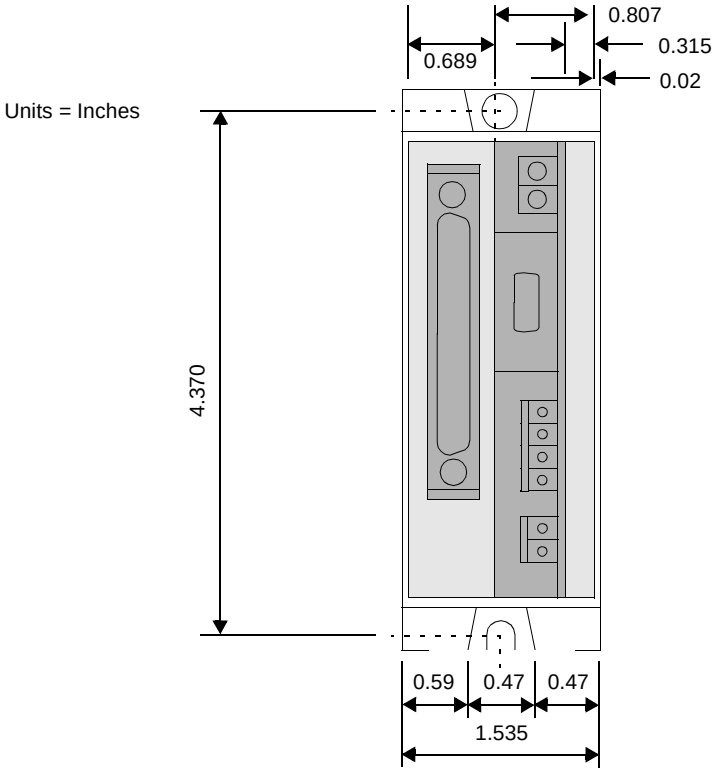


Figure 4 Controller Mounting Hole Positions and Dimensions

Figure 5 shows the proper mounting of the controller for horizontal or vertical installation. Secure the controller to a mounting plate within the system configuration. The mounting plate should be at least one-eighth of an inch (2 mm) thick and be made of steel, aluminum or other material having good thermal conductivity.

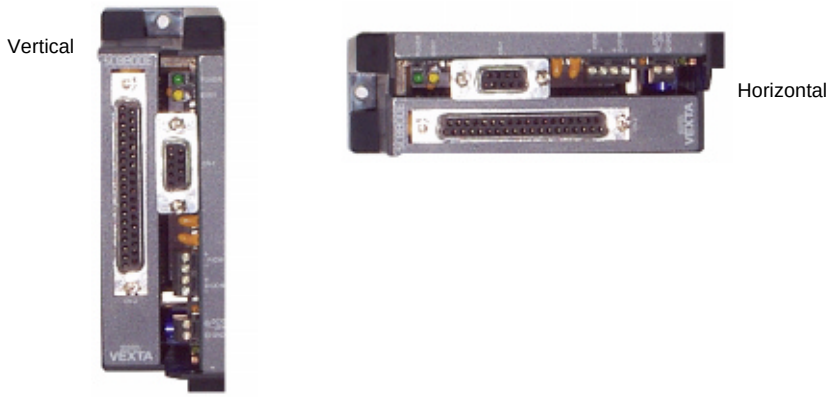


Figure 5 Controller Orientations

Front View of the Controller

Figure 6 shows the front panel of the controller's indicators and interface areas.

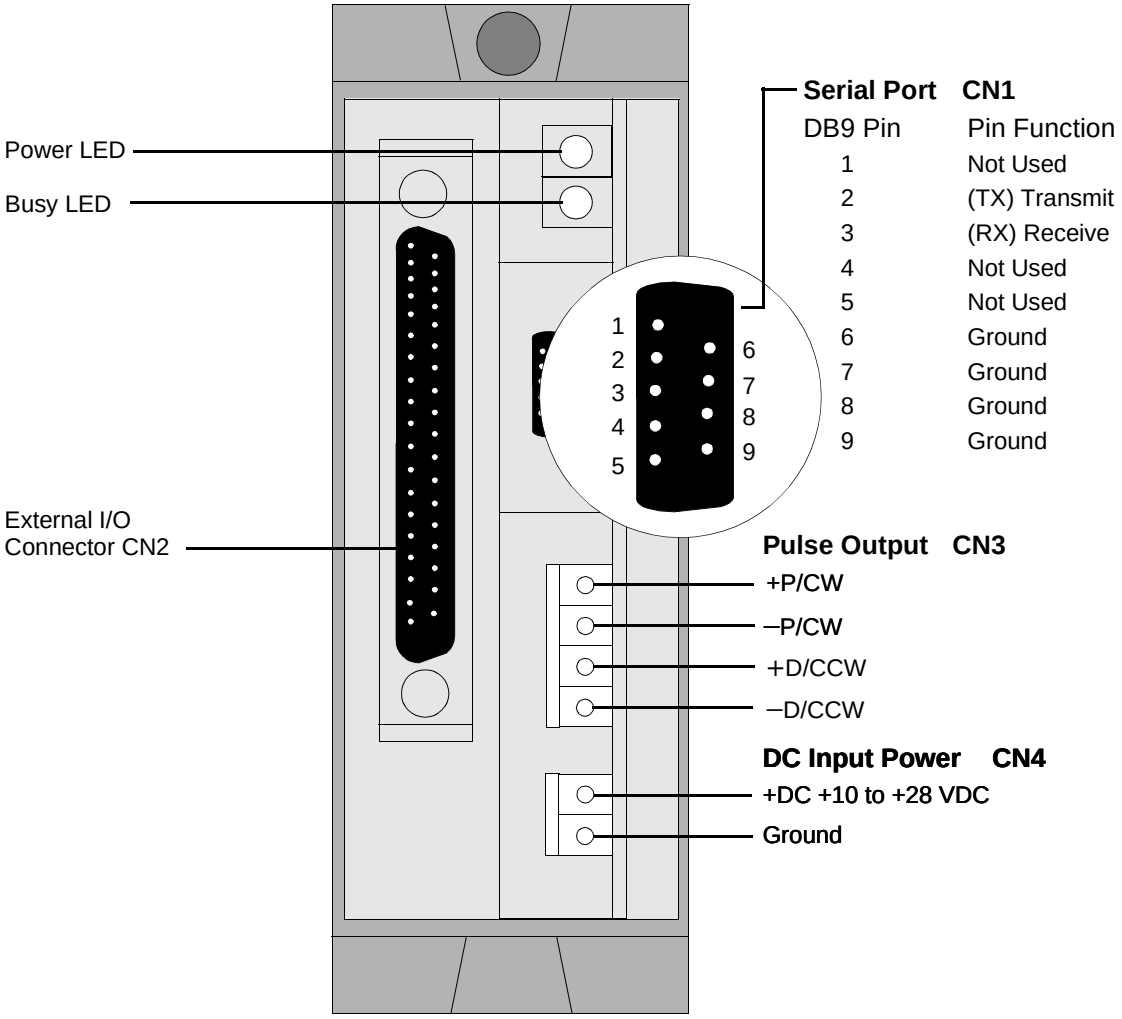


Figure 6 Front View, Controller Connections and Indicators

The four connectors associated with the power and control functions are listed below.

Function	Number of Pins	Type	Optically Isolated
DC Input Power (CN4)	2	Screw cinch terminals	No
Terminal or Computer (CN1)	9	RS-232C cable provided	No
Pulse Output (CN3)	4	Screw cinch terminals	Yes
External I/O (CN2)	37	DIN connector/housing provided	Yes

Subsequent sections contain details concerning:

- The wiring of these connectors
- The controller's indicator LEDs

Wiring Requirements for SC8800/SC8800E

- Use twisted pair hookup wire with a minimum diameter of 24 AWG for all signal control lines, and 18 AWG for all power lines.
- Strip no more than 0.2 inches (5.5 mm) of insulation from the lead wires to minimize the possibility of short circuits.
- Prevent excessive vibration or shock.

Wiring Precautions

- Keep the signal lines as short as possible.
- Keep the signal lines as far away as possible from power supply and motor lines.
- Shield the motor lead wires with conductive tape or wire mesh if noise generated on the motor lead wires causes interference with the signal lines.
- Always check the polarity of the DC power lines before turning on the power source.

The SC8800/SC8800E controller should be connected to a +10 to +28 VDC power source, as shown in Figure 6 on page 5.

WARNING Be sure the installation conforms to any applicable NEC, state and local codes.

Terminal or Computer Connection

The SC8800/SC8800E controllers can communicate with a host computer or a dumb terminal via an RS-232C port using standard ASCII terminal emulation, VT100 mode.

NOTE Other emulation modes may work, but some control commands (such as "ESC" or "CR + LF") may work differently.

Use the following terminal setup:

- Baud rate = 9600 (Transmit and receive)
- 8 data bits
- 1 stop bit
- No parity
- Full duplex mode (Transmit and receive independently)
- Handshake = NONE (NO hardware or Xon/Xoff protocol)
- CR translation = CR (Carriage return only, no "LF" line feed)
- Line wrap = ON (not essential)

Further terminal screen recommendations:

- 80 columns
- 16 to 24 rows

RS-232C Connection

Controller serial RS-232C port CN1 is connected via the provided shielded cable to a terminal, as shown in Figure 6 on page 5. The CN1 pin out is also listed in Figure 6 on page 5.

Motor Driver Connection

The SC8800/SC8800E controllers are connected to the driver module, as shown in Figure 7.

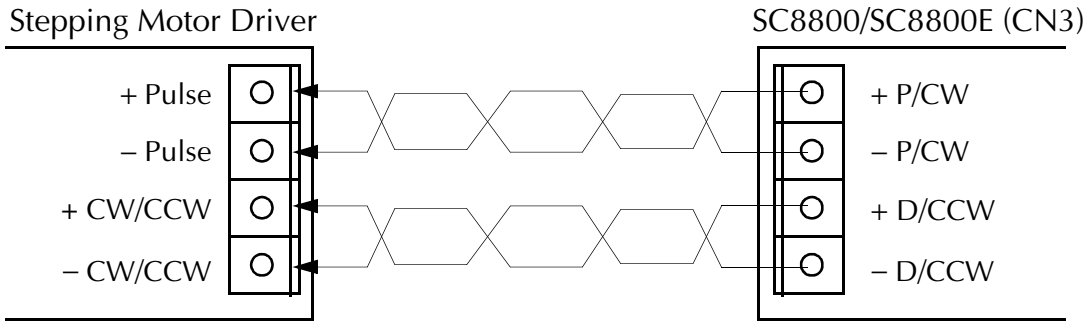


Figure 7 Connections between the Controller and the Motor Driver

NOTE Typical internal circuits for motor/driver I/O as well as all other Input and Output connections are shown in I/O Circuit Diagrams, beginning on page 10.

Controller Indicator Lamps

Figure 6 on page 5 shows the orientation of the controller's LED indicator lamps. The Power LED lamp is green and the Busy LED lamp is amber. These indicators have the following properties:

1. Upon applying power to the controller, the Busy lamp will flash on for an instant, and the Power lamp will turn on.
2. The Busy lamp turns on during program execution.

—If a user wants to have an indication of motor movement during a program, a signal from Output 1 or Output 2 can be used to drive an external indicator.

3. Both the Power and Busy lamps will alternately flash on and off to indicate a fault condition. If the controller encounters a problem, it can be with:

—Hardware or firmware

—Limit switches

—Software limit parameters

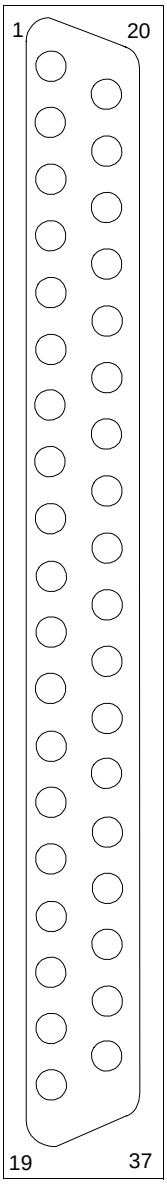
4. When the fault has been cleared, the lamps revert to their normal modes.

I/O Connections

External I/O Port

Figure 8 provides an enlargement of the CN2 port with pin assignments grouped for the four major control activities.

CN2



Inputs

5	Stop +
24	Stop –
12	Start +
31	Start –
11	Input 1 +
30	Input 1 –
10	Input 2 +
29	Input 2 –
9	Input 3 +
28	Input 3 –
8	Input 4 +
27	Input 4 –
7	Timing +
26	Timing –

Outputs

15	Output 1 +
34	Output 1 –
16	Output 2 +
35	Output 2 –
17	Fault +
36	Fault –
18	Busy +
37	Busy –

Limit Switches

6	Home +
25	Home –
13	CW Limit +
32	CW Limit –
14	CCW Limit +
33	CCW Limit –

Encoder Inputs (SC8800E only)

1	+ 5 VDC
20	Ground
2	Channel A +
21	Channel A –
3	Channel B +
22	Channel B –
4	Channel Z +
23	Channel Z –

Note: Pin 19 not connected

Figure 8 CN2 Pin Assignments

I/O Circuit Diagrams

Internal Circuits for Motor and Driver I/O

Figure 9 shows the internal circuits for motor and driver I/O.

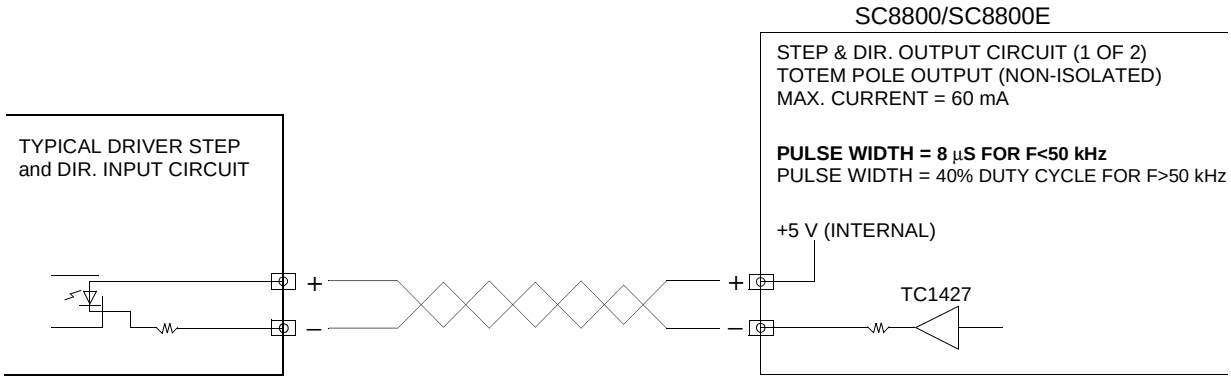


Figure 9 Internal Circuits for Motor and Driver I/O

Typical Input Connections

Figure 10 shows the typical input connections.

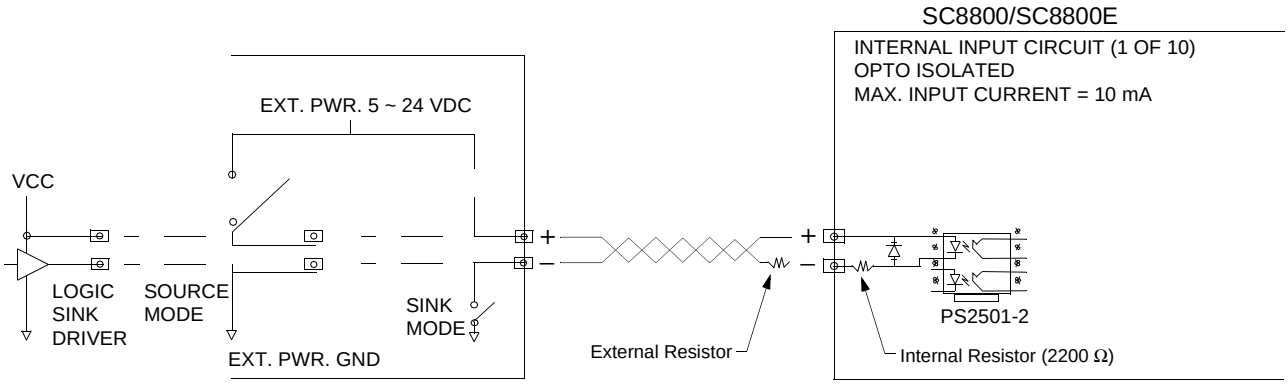


Figure 10 Typical Input Connections

SC8800/SC8800E Inputs

The SC8800 maintains ten input connections. They are designated:

Input 1	Programmable	START	Dedicated
Input 2	Programmable	STOP	Dedicated
Input 3	Programmable	CW Limit	Dedicated
Input 4	Programmable	CCW Limit	Dedicated
Timing	Dedicated	Home Limit	Dedicated

Inputs are:

- Optically isolated
- Activated by applying power
- Programmable
  - –Inputs can be read as high or low via software

Inputs support voltages from 5 to 30 VDC.
Input current **must** be limited to 10 mA maximum.

If an application requires an input voltage greater than 28 VDC, the user **must** provide an external resistor. The external resistor value can be calculated from the following equation:

$(V_{in} - 1.5\ V) = (R_{external} + 2200\ \Omega) \times (10\ mA)$

An application using the START input is shown on page 21.

Typical Output Connections

Figure 11 shows the typical output connections.

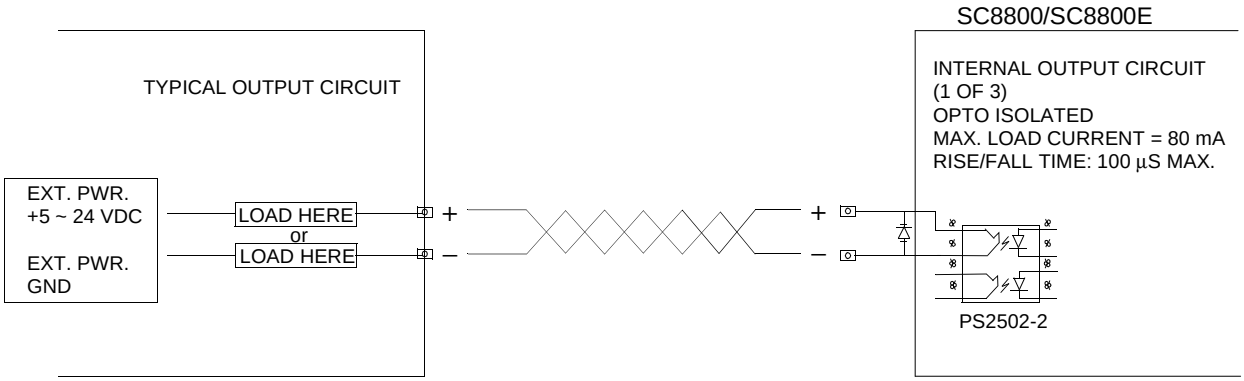


Figure 11 Typical Output Connections

SC8800/SC8800E Outputs

The SC8800 supports four outputs. They are designated:

Output 1	Programmable
Output 2	Programmable
FAULT	Status/Dedicated
BUSY	Status/Dedicated

Outputs are:

- Normally open
- In sinking output, open collector mode
- Rated at 80 mA maximum current
- Of a voltage specified range from 5 to 24 VDC
- Set high (ON) or low (OFF) via software

Typical Encoder Connections (SC8800E only)

Figure 12 shows typical encoder connections.

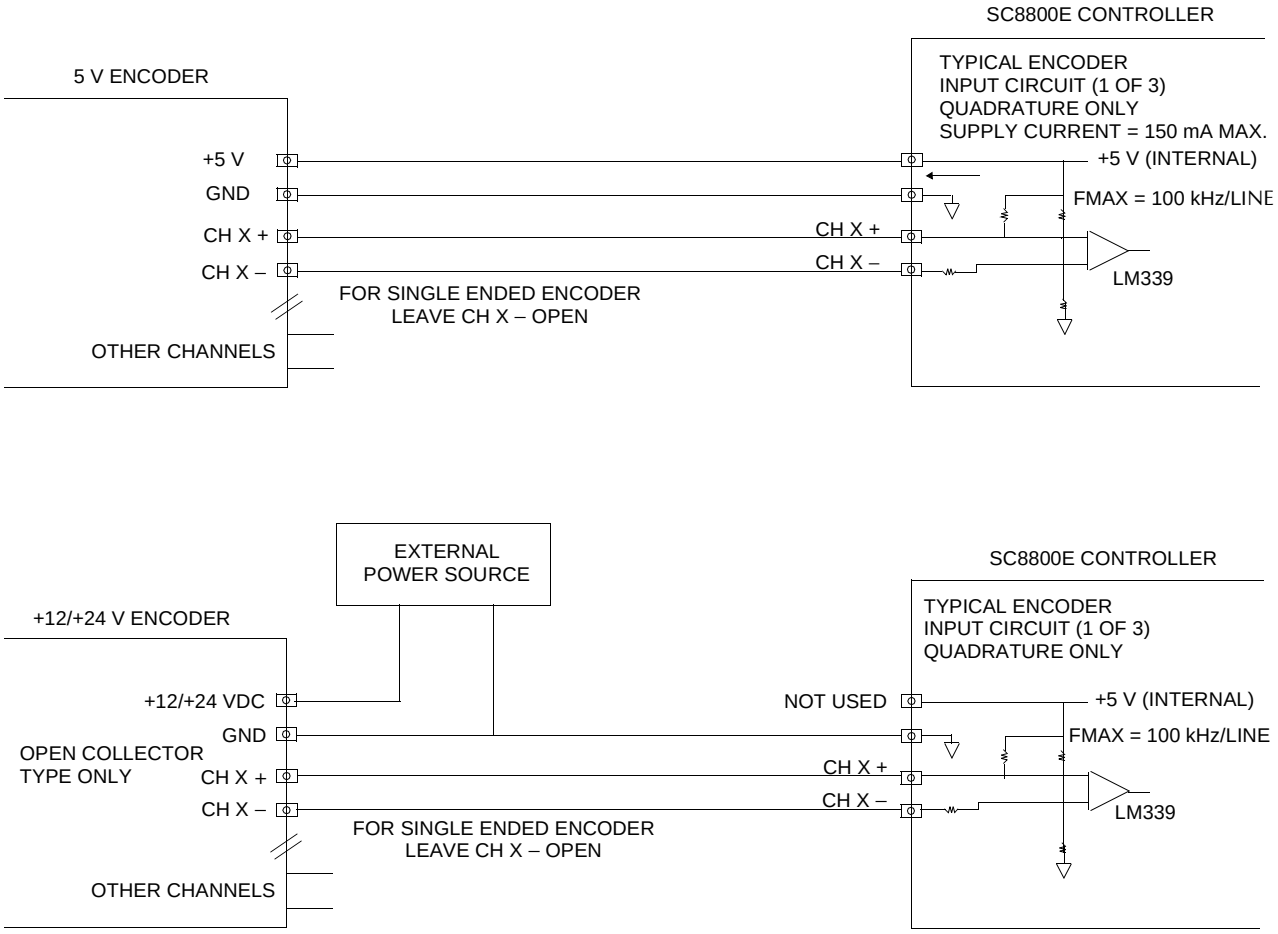


Figure 12 Typical Encoder Connections (SC8800E only)

Cabling for Daisy Chain Configuration

When a system's serial operator interface is connected to more than one SC8800/SC8800E, the controllers are organized in a string referred to as a daisy chain.

- Daisy chain cabling is a customer furnished item.
- All wiring requirements must be observed. See Wiring Precautions on page 6.
- Up to 35 controllers can be managed via a single terminal.
- The proper cabling technique for a daisy chain configuration is shown in Figure 13.

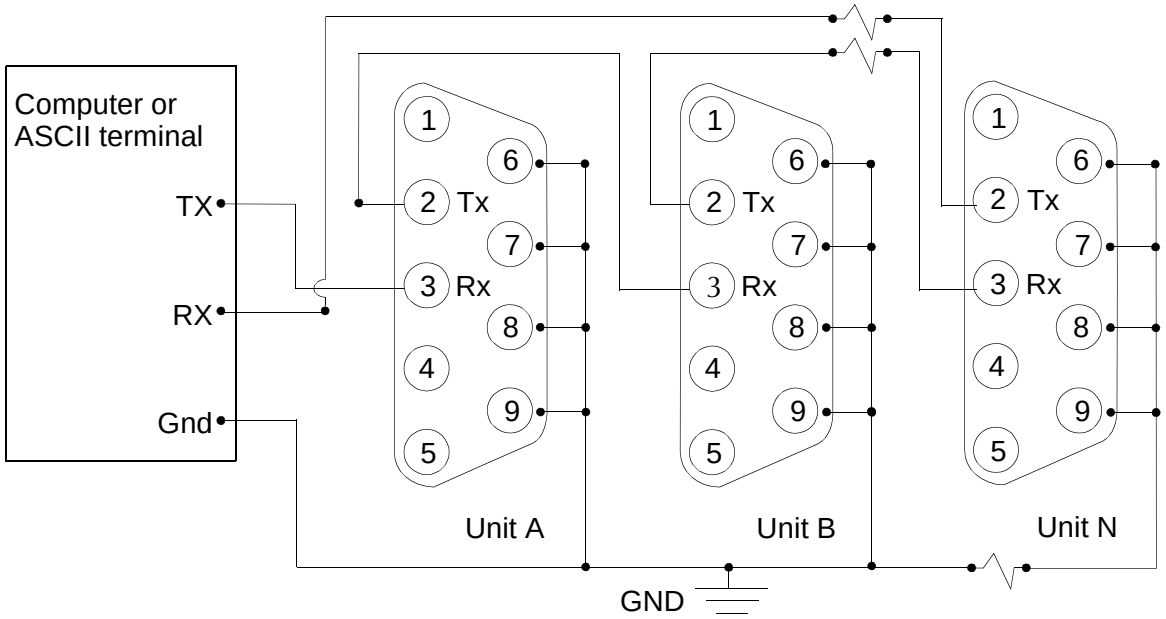


Figure 13 Cabling for Daisy Chain Configuration

Control Language Programming Reference

The SC8800/SC8800E contains within its onboard firmware:

- Extensive control language with an instruction set for motor/driver system management
- Interactive ability to execute keyboard input commands received via a serial communications link
- Complete editor for creating, storing, modifying, copying or deleting control programs
- Transparent execution module for performing the stored programs

Section Outline

This section contains:

- Execution Mode selection
- Programming conventions
- Using the interactive Immediate Mode
- Using the Program Mode with the embedded editor
- Using keyboard (KB) entry for variables
- Programming application examples
- Instruction set

Elements of the instruction set include:

- Motion commands
- Variables
- Parameters
- Dedicated keystroke entries

Details concerning the instruction set are as follows:

- A directory of commands is on page 25
- Conventions used in describing commands are on page 26
- A description of each command begins on page 27

Execution Mode Selection

SC8800/SC8800E controllers operate in either of two modes, *Immediate* or *Program Mode.*

Immediate Mode

- The controller executes commands received via the serial port (CN1) by keyboard entry from an ASCII terminal or computer.
- The controller initiates actions immediately, except if it is already executing a program.
- 18 commands are used exclusively from the keyboard in Immediate Mode.

Instructions for using the Immediate Mode begin on page 17.

Program Mode

- A sequence is created, altered or deleted using the embedded editor.
- Commands in a sequence are executed one at a time until the last one has finished.
- 16 commands are used exclusively by the editor in Program Mode.

Instructions for using the Program Mode embedded editor begin on page 18.

NOTE Thirty-six commands are used in both the Immediate and Program Modes.

Programing Conventions

Programming conventions used with the embedded editor for Program Mode, or with a terminal for interactive control in Immediate Mode, are as follows:

Syntax

- Commands are *not* case sensitive.
- A space or semicolon *must* separate multiple commands that are entered on the same line.
- A space may be used to separate commands followed by a digit field.
- Parameters that accept decimal values are fixed at three decimal places; additional decimal digits are ignored.
- Leading zeros of integers are ignored.

NOTE The underscore character (_) is used for clarity in this document to indicate a space between words or characters. It must not be used in sequences or in keyboard command entries.

Commands, Variables and Parameters

Commands

- Commands cause the controller to perform an activity.

Three Ways to Implement Command Elements

- From the keyboard in Immediate Mode
- By the editor in Program Mode
- From the keyboard (KB), in a hybrid mode, utilizing the KB interactive property as described on page 20.

Variables

- Variables hold information that define or report the state of the system.
- The SC8800/SC8800E control language provides four general purpose long integer variables: W, X, Y and Z.
- These variables may be used in conjunction with the instruction set specific variables PC, EC, V, VS, LOOP, D, CP or any constant within the range ± 2,147,483,647.

Arithmetic Operations with Variables

- Limited to only two terms at a time
- Valid arithmetic operators are: + , – , * , / or % (modulus)

Examples of arithmetic expressions are:

```
X = 100      'Set X to 100, this is an assignment
X = X +1     'Increment X as a counter by 1
Y = X / W    'Calculate Y from X and W
Z = PC       'Set Z to the current value of the Position Counter
```

Parameters

- Parameters are used to qualify elements of the instruction set.

Conditional Parameters

- Conditional parameters perform a comparison of two terms.
  —They must be enclosed in a pair of parentheses.
- Conditional test parameters have the form:
  (Variable 1 {Conditional} Variable 2 or constant)
  —The item appearing to the left of the conditional must be a variable.
  —The item on the right of the conditional can be a variable or a constant. Valid conditionals are:

```
=        'Equal to
!=       'Not equal to
<        'Less than
<=       'Less than or equal to
>        'Greater than
>=       'Greater than or equal to
```

Valid variables for conditional testing within the control language are:

```
IN       '4 programmable input bits (condition must always be = or != )
PC       'Position Counter value
EC       'Encoder Counter value
CP       'Compare Encoder and Position Counter values to Check Position
V        'Velocity
D        'Distance
W, X, Y, Z   'General Purpose Variables
```

Conditional testing makes decisions for the program when branching inside a program using conventional IF, ELSE, ENDIF statements, as well as WHILE and ENDW loop commands. Examples of conditional testing are:

```
IF (IN = xx01)        'Check for input #2 OFF and #1 ON
IF (PC > 200000)      'Check for Position Counter greater than 200000
IF (PC < -45000)      'Check for Position Counter less than -45000
WHILE (X!= 1000)      'While X is not equal to 1000, loop
WHILE (PC<X)          'While Position Counter less than X, loop
```

Using The Immediate Mode

The SC8800/SC8800E manages a stepping motor control system by keyboard entry, utilizing RS-232C serial communications between an ASCII terminal or a computer, and the controller. Some examples of this operating mode are described below.

Keyboard Input		System Response at Terminal
R		Displays current values of system parameters/status and I/O
CHKMEM		Displays the amount of available memory
HELP		Displays all the commands
MR500		Specifies number of pulses per motor revolution
VS100_V40000_T.1_d25000 MI		Sets motion variables and performs motion immediately, if not currently executing commands
RESET		Performs a system reset
<BKSP>	single keystroke	Erases last character in system buffer
<Esc>	single keystroke	Discards the current line and returns a system prompt to display
<ENTER>	single keystroke	Completes the line input and causes system to accept current line information
TALKC		Makes logical connection to Unit C in a daisy chain configuration

NOTE No special or external editors are required to program the SC8800/SC8800E.

Using The Program Mode

The Program Mode employs an embedded editor with 8 Kilobytes of nonvolatile memory available for creating and storing sequences.

Rules for Handling Sequences

- Sequences are identified *within the system* by number.
- Up to 50 sequences (0 to 49) can be stored and accessed via the RS-232C port.
- Only sequences identified as 0 through 15 are accessible from a PLC, for example, via the programmable inputs in conjunction with the dedicated START input signal. (See Start-Up Techniques on page 20.)
- Indirect access to sequences 16 through 49 are made by using the CALL or JMP SEQ commands.
- Sequences are designated utilizing the following conventions:

  —User assigns a number from 0 to 49.

  —User assigns an alphanumeric name of up to eight characters.
- Subsequently, user-named sequences are automatically assigned numbers by the system.

Editor Conventions

- The editor will generate line numbers automatically within a sequence.
- Syntax errors are not checked until the sequence is executed.
- When programming a new sequence, the editor prompts the user with the following message:

```
Empty.. Direct insert mode (ESC/Q=exit)
```

- Data is entered line by line with the <**ENTER**> keystroke terminating each line.
- An <**Esc**> or **Q** keystroke terminates the sequence and exits the Edit Mode.
- A current sequence is automatically saved when the user exits the Edit Mode.

There are three editor line commands:

- **Ax** or **ALTx** to alter the previously entered line x.
- **Ix** or **INSx** to insert new data; It has two forms:

  —Insert a new line between existing lines of a sequence.

  —Insert a new line as the final line of a sequence.

  If insert is selected and the last line number is used, the editor will automatically begin inserting new line numbers as required.
- **Dx** or **DELx** to delete the previously entered line x.

Using the Editor

The EDIT command is invoked to enter the editor in order to create or modify a sequence. Two examples for using the editor are shown below.

Example 1 Create a new sequence which is to be designated **5** by the programmer.

Prerequisite: System memory currently does not contain a sequence assigned as **5**.

Keyboard Input	System Response	Comment

EDIT_5	`Seq.5:`
	`Empty..Direct insert mode.`
	`(ESC/Q=exit)`
		User has entered the Editor Mode; system indicates that Seq. 5 is available for input

PC0	`(1) PC0`	Sets Position Counter to zero
TA2	`(2) TA2`	Sets Acceleration Time to 2 seconds
TD2.3	`(3) TD2.3`	Sets Deceleration Time to 2.3 seconds
VS500_V5000	`(4) VS500_V5000`	Sets Velocity parameters
D15000	`(5) D15000`	Sets Distance to 15000 steps
MI	`(6) MI`	Executes an Index Move
Q		Exits the Editor Mode

Sequence **5** has been programmed and saved to system memory. It is available for execution when invoked by a RUN command or START input if sequence **5** is selected by input.

Example 2 Alter an existing sequence designated as **9**.

Prerequisite: System memory currently contains a sequence assigned as **9**.

Keyboard Input	System Response	Comment

EDIT_9	`Seq. 9:`
	`(1) VS100_V10000_T.25`
	`(2) D10000`
	`(3) MI`
		User has entered the Edit Mode; system indicates that Seq. 9 exists and displays three lines of code

	`➜ Select Ax, Ix or Dx (Alt/Ins/Del/Q=exit)`	System asks for action of inserting, deleting or exiting
	`>>Command:`	Command prompt awaiting input

Sequence **9** can now be modified. When the user exits the editor, the modified sequence will be automatically saved.

Using Keyboard Entry for Variables

Sequences have an interactive property which allows program variable values to be introduced from the keyboard while the sequence is running.
Conditions for keyboard entry of variables are:

- SAS command is used to prompt the user for value to be entered
- Only numeric values are permitted
- Ranges of variables remain the same
  —Integer variables W, X, Y, Z= ± 2,147,483,647
  —Any decimal place values are discarded
- Pressing the <**Esc**> key with or without preceding values is interpreted as 0
- Pressing <**Enter**> or <**CR**> with no preceding value is interpreted as 0

Format examples for keyboard entry of variables are:

```
X=KB        'X takes the integer value entered from the keyboard
Y=Z+KB      'Y equals Z plus the value entered from the keyboard
```

A sequence example is shown below:

Example

```
(5) SAS Enter number of
boxes to build today: \0      'Prompt message asks user to enter desired number;
                              \0 suppresses CRLF at end of ASCII string leaving cursor on
                              same line containing message
(6)X=KB                       'Get input from keyboard, assign it to X
```

NOTE The KB interactive property has an *extended* capability. Information can be delivered to the controller from a touch screen, a PLC or any external device that generates ASCII characters via the RS-232C port.

Start-Up Techniques

There are three approaches for automatically starting system operations:

- Unattended under the `CONFIG` program control, no input is required.
- Unattended under the `CONFIG` program control, requiring a signal from one or more inputs.
- One Step Manual Control using the START input initiates program execution.

Unattended

To implement a totally unattended start-up, the programmer utilizes a special sequence named `CONFIG.`
The start-up plan is as follows:

- Immediately upon receiving power, the controller's firmware looks for `CONFIG` in the nonvolatile memory.
- If this sequence exists, it is executed.

The CONFIG sequence can be programmed to contain all the starting parameters and motion commands required to begin system activities. However, a preferred technique is to insert a single CALL statement in the CONFIG sequence. The CALLed sequence contains all the parameters and commands for start-up. See sequences CONFIG and BOB shown below.

```
Seq. CONFIG
(1) CALL BOB        'Execute sequence named BOB at power up or reset
Seq.31 BOB
(1) D1000 h+        'Set the distance and direction parameters
(2) VS100 V5000     'Set the velocity parameters
(3) MI              'Execute an index move
```

If the CALLed sequence contains *conditional* Input statements, then the start-up action can be synchronized with the status of up to 16 sequences. See sequences CONFIG and MIKE shown below.

```
Seq. CONFIG         'Automatic program activated at power up or reset
(1) CALL MIKE       'Execute sequence named MIKE at power up or reset
Seq.23 MIKE         'Sequence 23 named MIKE
(1) LOOP            'Infinite loop
(2) IF(IN2=1)       'Look to see if INPUT 2=1 (ON)
(3)   MGH           'Go Home
(4)   ELSE          'If INPUT2 is not=1, do nothing
(5) ENDIF           'End the statement of infinite loop
(6) ENDL            'END statement of infinite loop
```

One Step Manual Control

To implement automatic start-up by utilizing the START input, the programmer utilizes the controller's dedicated link between the four Programmable Input lines and the reserved sequences #0 through #15.
The start-up method is as follows:

- Upon power up, the controller checks the status of Inputs 1 through 4.
- It determines the decimal integer number equivalent to the binary status of the Inputs.
- It looks in the nonvolatile RAM memory to find a sequence with a matching number.
- If a sequence number match is found, it is executed immediately *when the START input is closed.*

Depending upon the status of the Inputs, this method allows up to 16 different start-up routines to be initiated. For reference, the possible Input states and the corresponding sequence numbers are shown below.

Input States				Equivalent Binary Input	Equivalent Sequence Number (Decimal)
4	**3**	**2**	**1**		
0	0	0	0	0000	0
0	0	0	1	0001	1
0	0	1	0	0010	2
0	0	1	1	0011	3
0	1	0	0	0100	4
0	1	0	1	0101	5
0	1	1	0	0110	6
0	1	1	1	0111	7
1	0	0	0	1000	8
1	0	0	1	1001	9
1	0	1	0	1010	10
1	0	1	1	1011	11
1	1	0	0	1100	12
1	1	0	1	1101	13
1	1	1	0	1110	14
1	1	1	1	1111	15

An example of this form of automatic start-up is shown below. It illustrates a situation where two completely different types of moves can be selected based upon the status of Input 1 and 2.

```
Seq. CONFIG          'Automatic program activated at power up or reset
(empty)              'No information, so no execution from here
Seq.1                'Sequence 1 executes when Input 1=1 and START input activated
(1) D1000 H+         'Sets distance and direction parameters
(2) VS100 V5000      'Sets velocity parameters
(3) MI               'Executes an Index Move
Seq.3                'Sequence 3 executes when Input 1 & 2=1 and START input activated
(1) D2000 H+         'Sets distance and direction parameters
(2) VS100 V5000      'Sets Velocity parameters
(3) MI               'Executes an Index Move
```

Application Sequences

To illustrate application specific sequences, the following two examples describe which sequences are used for closed loop position control. They employ the CP (**C**heck **P**osition) command.

Example 1 Record the largest position error in both positive and negative directions.

```
Seq.MAIN:
(1) X=0 Y=0          'Initialize X and Y
(2) LOOP100
(3) D50 MI
(4) DELAY.05         'Make sure that the motor settles
(5) CALL SUB         'Get encoder value from Seq. Sub
(6) ENDL
(7) X? Y?            'Print values of X and Y

Seq.SUB:
(1) Z=CP             'Capture signed value of CP
(2) IF ( X < Z)      'Update maximum positive direction error
    X=Z
(3) ENDIF
(4) IF ( Y > Z)      'Update maximum negative direction error
    Y=Z
(5) ENDIF
(6) RET              'Return to MAIN
```

Background: The CP (**C**heck **P**osition) command returns the error in motor steps by computing the difference in Encoder **C**ounter (EC) and **P**osition **C**ounter (PC) values using the following formula:

CP= ± Difference (motor steps)=(EC * MR/ER)-PC

where: MR = Motor Resolution
ER = Encoder Resolution

NOTE

The CP command returns a signed number in most cases except for when it is used in conditional branching. Inside the conditional brackets, CP returns an absolute value. The absolute value allows for a more simple operation of looking for a dead band range of ± x steps.

Example 2 Closed loop step motor with an error band of ± 5 steps.

```
Seq.CLOOP:                  'Subroutine CHECK ENCODER ERROR

(1) IF (CP >5)              'Error allowable is ± 5 steps, CP=abs (CP)
(2)     X=D
(3)     D=CP                'Get the step difference (signed)
(4)     MI                  'Make the correction move
(5) SAS ***                 Extra motion to correct step error
(6) ENDIF                   'Done
(7) RET                     'Return to MAIN
```

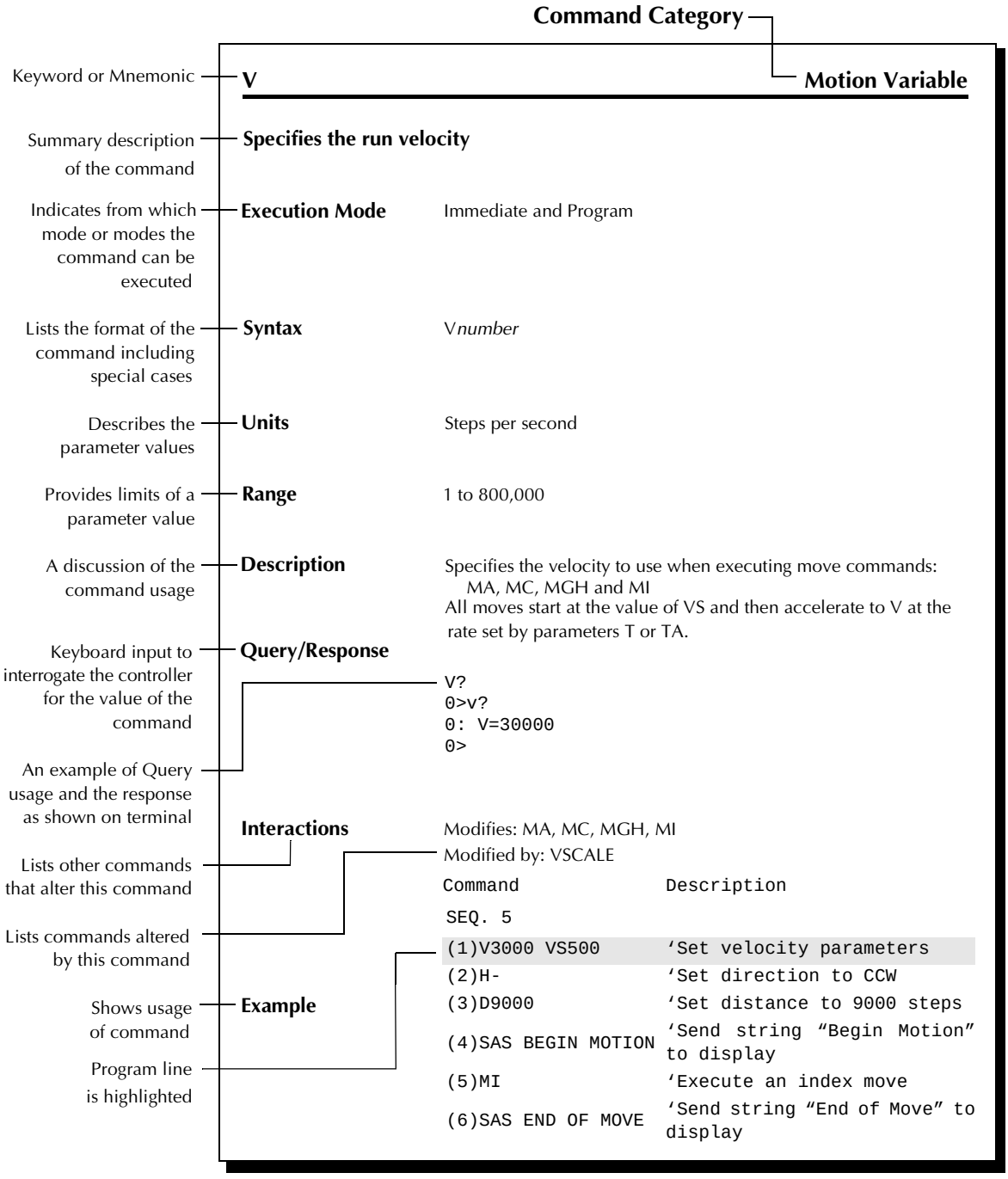
NOTE

In this example, D can also be operated like a variable X with one exception: When D is assigned to the variable X, the sign bit of X is used to modify the direction bit to provide the correct direction of travel for the error correction. Likewise, when X is assigned to D, it also takes the direction bit into account.

Instruction Set

Directory

Conventions



Instruction Set Commands

The instruction set for the SC8800/SC8800E is described on pages 27 through 103.

CALL — Program Control Element

Calls another sequence into the active program sequence

Execution Mode Program

Syntax Call sequence

Units Any stored sequence

Description Brings another sequence into the active sequence as a subroutine. Upon reaching the end of the subroutine, the program returns to the point of the previous sequence from which it left.

NOTE Particularly useful in Autostart mode with CONFIG file.

Example

Command	*Description*
`Seq. 5:`	
`(1) PC0`	‘Set the Position Counter to Zero
`(2) MI`	‘Do an Index Move
`(3) CALL TEST`	‘Execute Sequence named “Test”
`(4) SAS I’M BACK`	‘Echo Return to Original Sequence
`Seq. Test:`	
`(1) OUT1=1`	‘Turn On Output #1
`(2) DELAY2`	‘Delay 2 seconds
`(3) OUT1=0`	‘Turn Off Output #1
`(3) RET`	‘Return to Sequence 5

CHKMEM — Editor Command

Reports the amount of available memory

Execution Mode Immediate

Syntax CHKMEM

Units Bytes free

Description Keyboard entry command that returns the amount of memory space available in nonvolatile RAM.

Query/Response

```
u>CHKMEM <CR>
<CR>
xxxx bytes free
<CR>
<CR>
```

Example

Command	*Description*
`u>CHKMEM`	‘View available memory

Clears parameter display loop

Execution Mode Immediate and Program

Syntax CLR

Description Turns off the parameter display loop.

Example

Command	*Description*
`Seq. 3:`	
`(1) PC/`	'View Position Counter value
`(2) DELAY2`	'Delay 2 seconds
`(3) CLR`	'Turn off display loop

CLR_NVR — Editor Command

Removes the contents of nonvolatile RAM memory

Execution Mode Immediate

Syntax CLEAR_NVR

```
Are you sure? (Y/N)
```

Description Removes the contents of nonvolatile RAM. All stored sequences and variable values will be cleared. The CLEAR_NVR command will restore all parameters to their factory default values. Refer to Appendix F, *Default Values* on page 112 for the actual default values. The system will prompt the user for verification before actually clearing the nonvolatile RAM.

Example

Command	*Description*
`u>CLEAR_NVR`	‘Clear the RAM memory
`Are you sure? (Y/N)`	‘Prompt for verification

CONT — Motion Command

Continues a program after an interruption in a sequence

Execution Mode Immediate

Syntax CONT

Description Restarts a motion or program after a STOP input has been received or after a PAUSE, S or <Esc> command has been executed. The remaining portion of the interrupted command is executed.

Example

Command	*Description*
`u>D100000 MI`	‘Execute the Index Move
`u>STOP INPUT`	‘Stop Input is issued
`u>PC?`	’Check Position Counter value
`0> 2696`	’Value of Position Counter
`u>CONT`	‘Continue the stopped sequence; Motor will complete move (100,000 – 2,696 = 97,304 steps)

COPY — Editor Command

Makes a copy of a sequence

Execution Mode Immediate

Syntax COPY seqname1 seqname2

Where:

seqname1 is the Source sequence name

seqname2 is the Destination sequence name

Description Makes a copy of a sequence. The original sequence will still exist in memory.

If the destination sequence name already exists, a query message, `Overwrite (Y/N)`, is displayed to prompt the user for confirmation.

Examples

Command	*Description*
`u>COPY_5_10`	'Copy Sequence #5 to Sequence #10
`u>COPY_TEST1_TEST2`	'Copy Sequence TEST1 to Sequence TEST2

CP — System Control Command

Compares values of the encoder and motor position counters (SC8800E only)

Execution Mode Immediate and Program

Syntax CP

Units Motor steps

Range 0 to ±9,999,999

Description Compares the values of the encoder counter and the motor position counter. Each time the CP command is issued, the following equation is executed to determine the value of CP.

CP = Difference in motor steps = (EC * MR/ER) - PC

Where:

EC = Encoder counter value

MR = Motor resolution in steps

ER = Encoder resolution (quadrature value)

PC = Position counter value

NOTE The CP command can be used as part of a conditional expression. The CP command normally returns a signed number. However, when used in a conditional expression, only the absolute value is returned.

Query/Response

```
CP?
u>CP?
u:CP=number
```

Interactions Modified by: ER, MR, PC, EC

Example

Command	*Description*
`u>MR500`	'Set Motor Resolution to 500 steps
`u>ER2000`	'Set Encoder Resolution to 500 x 4 counts/rev
`u>PC0_EC0`	'Set counter values to zero
`u>D2000_MI`	'Execute an Index Move
`u>CP?`	'Check the Position error
`CP=-1`	'Indicates that the counter values are off by 1 step.

D — Motion Variable

Sets the distance to move for the MI move command

Execution Mode Immediate and Program

Syntax D±*number*

Units Pulses

Range 0 to 9,999,999

Description Determines the distance to be moved for the MI move command. The H command determines the direction of movement. The D± command also modifies the direction of the motor.

Query/Response

```
D?
u>d?
u:  D= 0
u>
```

Interactions Modifies: MI
Modified by: H, DSCALE

Example

Command	*Description*
`u>D2000`	‘Set distance to 2000 steps
`u>H_MI`	‘Execute a CW Index move
`u>D-8000`	‘Set distance to 8000 steps in the CCW direction
`u>MI`	‘Execute a CCW Index move

DELAY — Program Variable

Delays command execution

Execution Mode Program

Syntax DELAY*number*
DELAY (integer variable - W, X, Y, Z)

Units Seconds

Range 0.001 to 99,999.999 seconds

Description Causes the controller to wait a specified number of seconds before executing the next command. The value of the delay interval can be defined in a sequence.

NOTE Only used in Program Mode, not applicable to Immediate Mode.

Query/Response

```
DELAY?
u>delay?
u:  DELAY=number
```

Example

Command	*Description*
`SEQ. 3:`	
`(1) MI`	‘Execute an Index Move
`(2) DELAY5.3`	‘Delay 5.3 seconds
`(3) MGH`	‘Return to the Home Limit Switch

DEL — Editor Command

Deletes a sequence from nonvolatile memory

Execution Mode Immediate

Syntax DEL_seqname

Units Seqname = sequence name = sequence number or alphanumeric name

Range If number, 0 to 49
If an alphanumeric filename, A to Z, 0 to 9, up to 8 characters

Description Removes a program sequence from nonvolatile RAM. The system will request confirmation of the DEL action. A deleted file cannot be recovered.

Example

Command	*Description*
`u>LIST`	‘List the stored programs
`u>DEL_TEST1`	‘Delete the program TEST1 from memory
`U>ARE YOU SURE? Y/N`	‘Request confirmation

Lists the name and size of stored sequences

Execution Mode Immediate and Program

Syntax DIR

Description Lists name and size of all currently stored files in memory.
The DIR command is the same as the LIST command when LIST is not followed by a sequence name.

Query/Response

```
DIR?
u>DIR
0>dir

Sequence Directory

CONFIG                  27 bytes    Seq.25 TEST  268 bytes
Seq. 1                  17 bytes    Seq.27 QQ    56 bytes
Seq.26 TEST2            202 bytes

* 6843 bytes free
```

DSCALE — Motion Variable

Scales the move distance variable

Execution Mode Immediate and Program

Syntax DSCALExx

Units Pulses (or as redefined by DSCALE)

Range 0 to 2,147, 000, 000

Description Parameter that scales the distance to be moved for the MI move command. The value for DSCALE can represent other useful application variables such as revolutions/inch or steps/foot.

NOTE The internal distance, in steps, is calculated by the following equation:

D (steps) = D (input) * DSCALE

Query/Response

```
DSCALE?
0>dscale?
0:  DSCALE=0
0>
```

Interactions Modifies: MI, MA, PC/
Modified by: MT

Example

Command	*Description*
`u>DSCALE500`	'Set DSCALE = the motor resolution (Motor Resolution is 0.72°/step or 500 steps per revolution.)
`u>D1`	'Set distance to 1 revolution
`u>MI`	'Motor will move 1 revolution

EC — System Variable

Sets the internal encoder pulse counter (SC8800E only)

Execution Mode Immediate and Program

Syntax ECxxx

Units Encoder pulses

Range ±2,147, 000, 000

Description Sets the internal encoder pulse counter to any value within range.

Query/Response

```
EC?
0>ec?
0: EC = 2000
0>
```

Interactions Modifies: CP
Modified by: ER

Example

Command	*Description*
`u>EC0`	‘Set the encode counter to zero
`u>D2000`	‘Set the distance parameter to 2000 steps
`u>MI`	‘Do an Index move
`u>ec?`	‘What is the encoder value

ECHO — Editor Command

Echos RS-232C commands

Execution Mode Immediate and Program

Syntax ECHO*n*

Range 0 = off; 1 = on

Description Suppresses the display of any characters being sent to the screen.

CAUTION If the ECHO is turned *off* when more than one controller is used in a daisy chain configuration, the chain is broken and communication is lost.

Query/Response

```
ECHO?
0:  ECHO= 1
0>
```

Example

Command	*Description*
`u>TALK1`	'Talk to Unit #1
`u>ECHO0`	'Turn off the ECHO

Enters the sequence edit mode of the embedded editor

Execution Mode Immediate

Syntax EDIT xxxxxxxx

Range ASCII characters

Description Enters the edit mode where sequences are created or modified. When leaving the edit mode, a sequence is automatically saved.

Sequences may be named in one of three formats (listed below in highest to lowest priority):

1. Number only
2. Number + sequence name (up to 8 characters, excluding the preceding numbers)
3. Alphanumeric sequence name only (up to 8 characters)

—The first 16 sequences, from #0 to #15, are executable from the external I/O.

—Sequence numbers may not be duplicated.

—If no sequence number is present, the controller assigns the next available number, beginning from sequence #25. From that point on, that particular sequence can be referred to by either its sequence name or sequence number.

—Internal sequence numbers are allocated by the editor.

There are three editor line commands:

- **Ax** or **ALTx** to alter the previously entered line x
- **Ix** or **INSx** to insert new data; It has two forms:

  —Insert a new line between existing lines of a sequence

  —Insert a new line as the final line of a sequence

  If insert is selected and the last line number is used, the editor will automatically begin inserting new line numbers as required.
- **Dx** or **DELx** to delete the previously entered line x

Interactions Modified by: DEL, CLEAR_NVR, COPY

Examples

Command	*Description*
`u>EDIT 25`	‘Create (or modify) Sequence # 25
`u>EDIT OM`	‘Create Sequence OM, #26 is assigned
`u>EDIT 3TEST`	‘Create (or modify) Sequence 3TEST

ELSE — Program Control Element

Execution Mode Program

Syntax ELSE

Description Branches to an alternate operation if the conditional IF is not true.

Interactions Modifies: IF, ENDIF

Example

Command	*Description*
`SEQ.5:`	
`(1) IF(IN=0001)`	'If input #1 is on and others are off
`(2)  MA0`	'Return to 0 position
`(3)  ELSE`	'Branch on not true
`(4)  MGH`	'Go home
`(5) ENDIF`	'End of conditional statement

END — Program Control Element

Program end statement

Execution Mode Program

Syntax END

Description The END statement will terminate its associated program from anywhere it is detected; e.g. it is not required to be the last line for proper sequence execution.

NOTE

There are two reasons for using END:

- Debugging
- Documentation

Example

Command	*Description*
`(1) MI`	'Execute an index move
`(2) Delay 5.3`	'Delay 5.3 seconds
`(3) END`	'Exit the sequence (optional)
`(4) MGH`	'This line is ignored

ENDIF — Program Control Element

Completes an IF program segment

Execution Mode Program

Syntax ENDIF

Description Indicates the completion of a conditional IF statement.

Interactions Modifies: IF, ELSE

Example

Command	*Description*
`SEQ.5:`	
`(1) IF(IN=0001)`	'If Input #1 is on and others are off
`(2) MA0`	'Return to 0 position
`(3)  ELSE`	'Branch on not true
`(4) MGH`	'Go home
`(5)ENDIF`	'End of conditional statement

ENDL — Program Control Element

Completes a program loop segment

Execution Mode Program

Syntax ENDL

Description Indicates the end of a LOOP segment of a sequence.

Interactions Modifies: LOOP

Example

Command	*Description*
`Seq. 5:`	
`(1)LOOP 50`	'Loop the following 50 times
`(2)  MI`	'Do an index move
`(3)ENDL`	'End the loop

ENDW — Program Control Element

Completes a WHILE program segment

Execution Mode Program

Syntax ENDW

Description Indicates the completion of a WHILE conditional segment of a sequence.

Interactions Modifies: WHILE

Example

Command	*Description*
`SEQ. 5:`	
`(1)WHILE (IN=0001)`	'While Input #1 is on
`(2)  MI`	'Do index moves
`(3)ENDW`	'End of conditional statement

Sets the encoder resolution (SC8800E only)

Execution Mode Immediate and Program

Syntax ER

Units Number of encoder quadrature pulses per motor shaft revolution. Quadrature refers to number of counts that the controller receives per encoder line.

Range 0 to 65,535

Description Sets the resolution of the encoder that is used with the system.

NOTE The ER command is required for the SC8800E to work properly in conjunction with the motor resolution (MR) and check position (CP) commands. When no encoder is used, MR and CP are not valid.

Query/Response

```
ER?
0>ER?
0: ER= 2000
0>
```

Interactions Modifies: CP, MR

Example

Command	*Description*
`u>ER2000`	‘500 lines per revolution; 4 x 500 = 2000
`u>D8000`	‘Set distance to 8000 steps
`u>MI`	‘Execute an index move
`u>CP?`	‘Check position error

EV — System Control Command

Turns on event outputs on the fly

Execution Mode
Immediate and Program

Syntax
EVn OUTx=n @Txx.xx or @Dxxxxxxxx or @Vxxxxxx

Units
EV1 = event #1; EV2 = event #2

OUTx=nx = 1, 2 or 3; n = 0 (Off) or 1 (On)

@Txx.xxx Event Trigger, using time.
Time interval starts at beginning of motion.

@Dxxxxxxxx Event Trigger, using distance.
Distance interval in motor steps starts at beginning of motion.

@Vxxxxx Event Trigger, using velocity.
Valid when velocity is accelerating or constant.
Not valid when velocity is decelerating.

NOTE
EVn 0 is used to clear the event parameter. Clearing the event parameter does not clear or reset the outputs themselves.

Range
1 to 2

Description
Turns on outputs on-the-fly based on the occurrence of up to 2 events.

Once the event parameters have been declared, they remain active until they are invalidated by the EVn 0 command.

Query/Response

```
EV?
0>ev?
0:  ev1 out1=1 @t2
0:  ev2 out=110 @d5000
```

Interactions
Modified by: EVn 0, DSCALE, VSCALE

Example

Command	*Description*
`u>ev1 Out2=1 @v8000`	‘Turn on Output#2 when reach rate of 8000 steps/sec
`u>ev2 Out1=1 @T2`	‘Turn on Output#1 at interval of 2 seconds
`u>MC`	‘Execute a continuous move
`u>EV1 0 EV2 0`	‘Clear events number 1 and 2

H±

Motion Variable

Sets motion direction

Execution Mode Immediate and Program

Syntax H {NIL|+|–}

Range + = CW; – = CCW

Description Sets the direction of rotation of the motor shaft. During a continuous move (MC) the H± command is used to change motor direction on-the-fly.

Query/Response

```
H?
0>h?
0:  H= +
0>
```

Interactions Modifies: MA, MC, MGH, MI, MT
Modified by: D±, V±

Example

Command	*Description*
`u>V2000 t1 H+`	‘Set motion parameters
`u>MC`	‘Execute a continuous move
`u>V-1000`	‘Change speed and direction
`u>MC`	‘Execute changes
`u>H+`	‘Change direction back to CW

HELP — Editor Command

Requests system Help screens

Execution Mode Immediate

Syntax HELP [{NIL|1|2|3|4|5}]
HELP = HELP1 = Display all the commands
HELP2 = Display HELP for the motion elements
HELP3 = Display HELP for the system elements
HELP4 = Display HELP for the programmable elements
HELP5 = Display HELP for new functions

Description Displays 1 of 5 help menus. The first screen is an abbreviated list of all the elements in the instruction set. Screens 2 through 5 are grouped by function and provide a summary of the elements.

Query/Response

```
HELP?
u>HELP [{NIL|1|2|3|4|5}]
```

See Figure 14 through Figure 18 below.

```
* COMMAND LIST mnemonics. (Also see Help2, Help3, Help4, Help5)
 ---------- ( x= digit field. []= optional) ----------
  CALL x       ENDL         LOOP [x]     RET          X=
  CHKMEM       ENDW         LIM x1 x2    RESET        Y=
  CONT         END          MGH          RUN x        Z=
  COPY x y     ER x         MA +/-X      S            W=
  CLEAR NVR    EV n..       MC           STOP         EC/
  CLR          HELP x       MI           SAS          PC/
  CP           H+/-         MR x         SYS          CP/
  D x[.x]      ID x         MT x.x       TA x.x       V/
  DSCALE x     IF(..)       OUTx=x       TD x.x       IO/
  DELAY x.x    INx=x        OUT=x        T x.x        # or REM
  DEL x        IN=x         OL xxxx      TIM n        \
  DIR          INV X        PAUSE        TALK x
  EDIT n       JMP x        PC+/-x       TR n
  EC +/-x      JMP SEQ x    PULSE X      V x[.x]
  ECHO n       KB           R            VS x[.x]
  ELSE         LOCK x       RAMP X       VSCALE
  ENDIF        LIST [X]                  WHILE(..)
0>
```

Figure 14 Help Screen 1

```
*  COMMAND LIST 2.

   VSx[.x]  : Start velocity.     V x[.x]: Final velocity.
   TA x.x   : Accel time(sec).    TD x.x: Decel time(sec).
   T x.x    : Accel/Decel time.   Mt x.x: Total move time(Auto).
   D x[.x]  : Index distance.     D+/-x: Distance with direction.
   H+/-     : Direction CW/CCW.   RAMP n: 0=Linear,1=S-curve,2=Parabolic.
   MI       : Move incremental.   MA+/-x: Move absolute.
   MC       : Move continuous.    MGH: Move go home.
   S (ESC)  : Stop with decel.    CONT: Continue after a stop.
   DSCALE   : Distance multiplier.STOP    : Hard stop, no decel.
   TIMxx    : Timing signal "AND" Home sw.  VSCALE : Velocity multiplier.
   PULSEx   : Set pulse mode. 1=1P 2=2P
   INVx     : Invert step output (1P mode)
   PC+/-x   : Position counter set.
   EC+/-x   : Encoder counter set.
   CP       : Evaluate position error:"(EC*MR/ER)-PC"
   MR x     : Motor resolution.(used w/CP only)
   ER x     : Encoder resolution.(used w/CP only)
   LIM x1 x2: Software position limit.(x1<x2)
0>
```

Figure 15 Help Screen 2

```
*  COMMAND LIST 3.
   ------------------------------------------
   OUTn=x  : Set 1 output bit.
   OUT=xxx : Set output bits.
   OLxxxx  : Invert output bits.
   IDx     : Set unit ID code(0-9, A-Z)
   ID*     : Same as ID0 w/"0:" suppressed.
   TALKx   : Enable communication to unit x.
   \       : Global command(bypass ID code).
   R       : Report system status.
   SYS     : Report operating time.
   TR x    : Program execution trace(0-1)
   CLEAR NVR : Clear user storage memory.
   LIST x : List sequence content.
   DIR or LIST : List sequence directory.
   EDITx [name] : Create or alter a sequence.
   DEL x [name] : Delete sequence.
   RUN x [name] : Run sequence.
   CLR     : Clear error or display loop
   COPY x y: Duplicate sequence. x=source, y=destination
   PC/ EC/ CP/ V/ IO/: Variable display loop.
   EVn OUTx=n @Tx (or @Dx/@Vx): Output on the fly.(2max)
0>
```

Figure 16 Help Screen 3

```
*  COMMAND LIST 4. (Sequence Control)
   ------------------------------------------------------------
   CALL x     : Call sequence subroutine.
   DELAY x.x : Delay time.
   DELAY var : Delay using variable X, Y..(t=milli-sec)
   JMP x     : Jump to a sequence line number.
   JMP SEQ x : Jump to a new sequence.
   SAS       : Send ASC string.
   IF(cond)  : Conditional branching.
   ELSE      : Alternate branching when IF=false
    ENDIF    : End of IF..(ELSE) body.
   LOOP x    : Loop count.(NO digit=infinite loop)
    ENDL     : End of LOOP body.
   WHILE(cond): Loop while condition =true.
    ENDW     : End of WHILE body.
   END or RET: Sequence END or RETURN(optional).
   # or REM  : Remark.
   INn=x IN=xxxx : Input conditional for IF/WHILE.
   X Y Z W =x    : Long integer variables.
   -(Conditional): Applied to IN= IN!=.
    Others: X,Y,Z,W,Pc,Ec,D,V or Constant can be used in either term
  using relational operator = != > > = < < =
   -Two terms math can be applied to all vars above using + –* / %
0>
```

Figure 17 Help Screen 4

```
Var = KB   : Get input from serial port.
Var may be X, Y, Z or W. Exp. Y=KB
SAS ..\..  : \0: suppresses CRLF
           : \X \Y \Z \W: print variable value within SAS string
```

Figure 18 Help Screen 5

ID — System Control Command

Sets the selected unit to the desired identification

Execution Mode Immediate

Syntax IDx

Range 0 to 9, A to Z, *

Description This command has two usage modes:

- When more than one controller is planned for daisy chain configuration, the command *must* be used at installation time. This will **initialize** each unit by setting it to its appropriate identification number.
- When a single controller is used, command ID* can suppress the ID number from the screen display. Only the prompt (>) is shown. To return to the previously displayed prompt, type **IDx**.

NOTE When in a daisy chain mode:

- If a controller has an ID other than 0, it will not display the sign on banner at power-up or after a software RESET.
- TALKn command must be used to establish communication between the host and the selected unit.
- A backslash character (\) preceding a command assigns it as a global command, overriding the need for a TALK command.
- A maximum of 35 units can be supported in a daisy chain.

Query/Response

```
ID?
0>id?
0: ID = 0
0>

>id?
ID = *
>
```

Example

Command	*Description*
`u>IDC`	‘Set to Unit C
`u>TALK C`	‘Talk to Unit C
`c>MGH`	‘Go home
`c>ID*`	‘Suppress the ID print code
`>`	‘Only the prompt is displayed now

IF — Program Control Command

Initiates a conditional testing program segment

Execution Mode Program

Syntax IF (Variable 1 {Conditional Operator} Variable 2 or constant)

Description Provides branching to:

- Other sequence lines, or
- Other sequences

Valid properties for using the IF statement are:

- Program segments must be completed with an ENDIF statement
- An IF..ENDIF segment may be nested up to 4 times
- An ELSE statement may be used in the segment

Valid conditional operators are:

=	'Equal to
!=	'Not equal to
<	'Less than
<=	'Less than or equal to
>	'Greater than
>=	'Greater than or equal to

Valid variable elements are:

IN	'4 programmable input bits (condition must always be = or !=)
PC	'Position Counter value
EC	'Encoder Counter value
CP	'Compare Encoder and Position counter values
V	'Velocity
D	'Distance
W, X, Y, Z	'General Purpose Variables

Example

Command	*Description*
`SEQ. 7:`	
`(1) IF (PC>25000)`	'If position counter is > 25000 steps
`(2) SAS Error! Reset`	'Echo message
`(3) MGH`	'Return to home position
`(4)  ELSE`	'Or
`(5) SAS Everything ok`	'Echo message
`(6) ENDIF`	'End of the IF statement

Defines a programmable input

Execution Mode Immediate and Program

Syntax The IN statement has two forms:

- IN=xxxx (Global - Test each bit simultaneously)
- INx=n (Single - Test bit separately)

Inputs are numbered with the following convention:

IN4321

1 = Programmable Input #1
2 = Programmable Input #2
3 = Programmable Input #3
4 = Programmable Input #4

Range x = 1 to 4 n has three possible states:

0 = OFF
1 = ON
x = don't care

Description The IN statement has two application areas:

- Select a sequence to execute at power-up
- Conditional testing in IF or WHILE statements

NOTE Valid sequences for power up are 0 to 15.

Example

Command	*Description*
`SEQ. 8:`	
`(1)SAS PRESS START`	'Notify user to press start
`(2)IF(IN4=1)`	'If input #4 is on
`(3) MGH`	'Go home
`(4)ELSE`	'If input #4 is not on
`(5)  WHILE(IN2=1)`	'While input#2 is on
`(6) MI`	'Execute an index move
`(7) ENDW`	'End the WHILE loop
`(8)ENDIF`	'End the IF condition

INV — System Control Command

Inverts step pulse output

Execution Mode Immediate and Program

Syntax INV*n*

Range
0 = active high (default)
1 = active low

Description Inverts the logic level of the step pulse output.
Three characteristics of INV are:

- Command is valid when the controller is in the 1-pulse mode
- Command is not valid in the 2-pulse mode
- Value of the parameter is saved in nonvolatile memory

NOTE

Pulse mode definitions are:

- 1-pulse = step and direction
- 2-pulse = up-clock/down-clock
- See *PULSE* on page 76

Query/Response

```
INV?
0>inv?
0: INV= 0
0>
```

Example

Command	*Description*
`u>PULSE 1`	‘Set pulse mode to Step and Direction
`u>INV1`	‘Set pulse logic to active low
`u>MC`	‘Perform a continuous move

Displays the input and output status

Execution Mode Immediate and Program

Syntax IO/

The data response is of the form:
FE_DCB_A9_8765_4321

Output #1 is the right-most bit.

Character/bit definitions are shown below.

I/O Bit	Status of:	I/O Bit	Status of:
F	Clockwise limit	8	Input #4
E	Counter clockwise limit	7	Input #3
		6	Input #2
D	Home limit	5	Input #1
C	Timing signal input		
B	Encoder "Z" channel	4	Busy Output
		3	Fault Output
A	Start input	2	Output #2
9	Stop input	1	Output #1

Range 0 = Off; 1 = On

Description Continuously displays the status of controller inputs and outputs. This data is updated every 0.2 seconds.

Query/Response

```
IO?
0>io?
0: 00 000 00 0000 0001
0>
```

Interactions Modified by: INV, TIM, OL, OUT, IN, MA, MI, MC, MGH

Example

Command	*Description*
`u>io/`	'Turn on display loop for I/O
`- 00 000 00 0000 0001`	'Display of results

JMP — Program Control Element

Moves program control forward or backward in a sequence

Execution Mode Program

Syntax JMPxx

Range Any valid line number that exists within the current sequence

Description Moves program control unconditionally either forward or backward to another line within the current sequence.

CAUTION

JMP should not be used to exit a sequence loop segment like IF, WHILE or LOOP under normal operation.

Exception: When the JMP *simultaneously* terminates a loop *and* the program.

Examples

Command	*Description*
`SEQ. 6:`	
`(1)D6000 MI`	‘Perform an index move of 6000 steps
`(2)IF(PC!=6000)`	‘Check for step error
`(3)JMP7`	‘Jump to line #7 true
`(4)  ELSE`	‘If condition not true
`(5)  MGH`	‘Go home
`(6)ENDIF`	‘End the IF loop
`(7)END`	
`(8)SAS ERROR`	‘Send error message to screen
`SEQ. 5:`	
`(1)D1000 MI`	
`(2)JMP 5`	‘Used as a Debug Tool in this case
`(3)D20000 MI`	
`(4)D30000 MI`	
`(5)END`	

JMP_SEQ — Program Control Element

Moves program control to another sequence

Execution Mode Program

Syntax JMP SEQxx

Range Any stored sequence

Description Causes program control in the current sequence to move to another sequence. This newly invoked sequence is executed as the current sequence.

NOTE Unlike the CALL command, when the newly invoked sequence is completed, it will *not* return to the previous sequence. All loop counters are reset when a JMP_SEQ command is issued.

Example

Command	*Description*
`SEQ. 9:`	
`(1) MI`	'Do an index move
`(2) JMP_SEQ TEST`	'Jump to Sequence TEST

KB — Program Variable

Allows sequence data to be entered remotely from the system keyboard

Execution Mode Program

Syntax

- SAS command is used to **prompt** for the entry value
- {W|X|Y|Z} = KB
- Numeric values only
- Pressing **<Esc>** key with or without preceding values is interpreted as 0
- Pressing **<Enter>** or **<CR>** with no preceding value is interpreted as 0

Range

- KB = ±2,147,483,647, same as integer variables {W|X|Y|Z}
- Decimal values are not valid and are discarded

Description The KB command permits sequences to have an *interactive* feature. Values for integer variables can be entered from the system keyboard while the program is running.

NOTE The KB interactive property has an *extended* capability. Information can be delivered to the controller from a touch screen, a PLC or any external device that generates ASCII characters via the RS-232C port.

Interactions
Introduced By: SAS
Modifies: W, X, Y, Z

Examples

Command	*Description*
`(5) SAS Enter number of boxes: \0`	‘Prompt message asks user to enter desired number; \0 suppresses CRLF at end of ASCII string leaving the cursor on the same line containing the message
`(6) X=KB`	‘Get input from serial port, assign it to X

LIM — Motion Variable

Sets software position limits

Execution Mode Immediate and Program

Syntax LIM x1 x2

The following conventions are used:

- x1 must be smaller than x2
- x1 = minimum position
- x2 = maximum position

Units Motor steps

Range ±2,147,483,647

Description Sets the software position limits based on the value of the internal position counter. The following conditions apply:

- Software limits will not be used if x1 = x2 or x1 = x2 = 0
- The value of these limits are related to the DSCALE function
- If a software limit is encountered, the motor will stop immediately and an error message will be returned to the screen.

CAUTION Ensure that the limits are set appropriately.

Query/Response

```
LIM??
0>lim?
0: LIM= -2500 4000
0
```

Interactions Modified by: DSCALE

Example

Command	*Description*
`u>LIM -1000 6000`	‘Set software positions
`u>V5000 h-`	‘Set motion parameters
`u>MC`	‘Do a continuous move
`u>*Software CCW limit`	‘Error message

LIST — Editor Command

Displays the program lines of a sequence

Execution Mode Immediate

Syntax LIST {[x]|[sequence name]}

Standards for using LIST are:

- LIST followed by a sequence name will list that sequence
- LIST followed by nothing will display all stored sequences, similar to the DIR command

Range Any stored sequence

Description Displays the program lines of a sequence.

Examples

Command	*Description*
`u>LIST TEST`	'List sequence TEST
`u>LIST`	'List all sequences currently stored in NVRAM memory

Locks specified sequence

Execution Mode Immediate

Syntax LOCK {[xx]|[sequence name]}

Range Any available sequence

Description Sets a lock bit on the specified sequence so that it cannot be altered or deleted. When a DIR or LIST command displays the roster of stored sequences, all LOCK protected sequences will be marked with an asterisk (*). The UNLOCK command disables the LOCKed status of a sequence.

Interactions Modified by: UNLOCK

Example

Command	*Description*
`u>LOCK TEST`	‘Lock sequence TEST
`u>EDIT test`	‘Attempt to edit TEST
`u>** Seq LOCKED (NOT alterable)`	‘Message

LOOP — Program Control Element

Repeats a series of program statements

Execution Mode

Program

Syntax

LOOP*n*
Standards for using LOOP are:

- LOOP followed by a number *n* will repeat the segment *n* times
- LOOP followed by nothing will repeat the segment indefinitely with termination provided by:
  —Encountering an S or STOP command or <ESC> keystroke
  —Encountering an end of travel limit parameter and/or switch

Range

0 to 65,535

Description

Repeats *n* times the list of statements placed between a LOOPn and an ENDL command.

Interactions

Modified by: ENDL

Example

Command	*Description*
`SEQ. 3:`	
`(1)LOOP10`	‘Repeat the program segment 10 times
`(2)  H`	‘Set direction to CW
`(3)  D2000`	‘Set distance to 2000 steps
`(4)  MI`	‘Do an index move
`(5)ENDL`	‘End the first Loop
`(6)  LOOP`	‘Loop indefinitely
`(7)  H-`	‘Change direction to CCW
`(8)  MI`	‘Do an index move
`(9)ENDL`	‘Return to beginning of Loop

Performs move from one position to another based on position counter

Execution Mode Immediate and Program

Syntax MA*n*

Range ±999,999,999

There are two valid ranges for *n*, one with increased precision. They are based on the status of the DSCALE parameter as shown below:

DSCALE	Range of *n*
zero	999,999,999
not zero	±999,999,999.xxx

Description Performs a move index, MI, from the current position to a new position based on data from the position counter. The new position is determined by the *absolute* value of the parameter entered with the MA command. Motor movement, both direction and distance, is determined by controller computation at the time of the move.

Figure 19 describes a typical absolute position move.

Interactions Modified by: DSCALE, PC, RAMP, T, TA, V, VS, VSCALE

Example

Command	*Description*
`SEQ. 7:`	
`(1) PC0`	‘Set position counter to zero
`(2) V3000 VS500`	‘Set velocity parameters
`(3) MA - 1250`	‘Move CCW 1250 steps

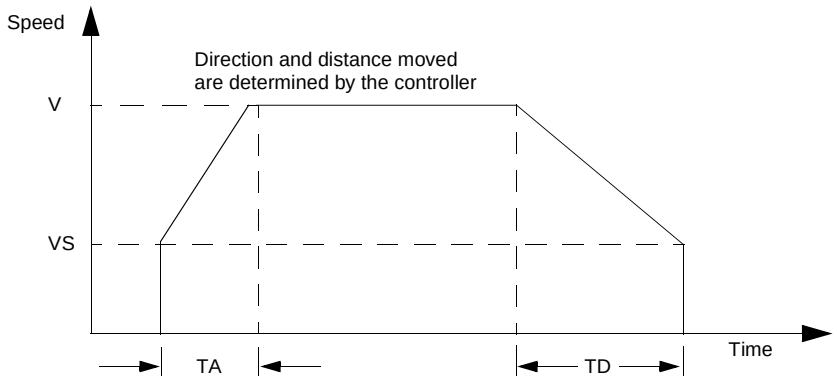


Figure 19 Speed/Time Profile for an Absolute Move

MC — Motion Command

Starts motor moving continuously based on current parameters

Execution Mode

Immediate and Program

Syntax

MC

Description

Starts motor moving continuously with the following logical standards:

- Uses current motion parameters, VS, TA, TD and V
- Upon reaching velocity, V, motor will remain at this speed while the controller executes the next command in the buffer.
- If no buffer commands exist:
  - — Motor continues to move at V
  - — Controller monitors direct input instructions and executes them
- Motion is terminated by direct input of any of the following instructions:
  - — PAUSE
  - — S
  - — STOP
  - — <Esc> keystroke
  - —Software limit
  - —End of travel limit

The MC command is mode specific. Rules for execution are as follows:

Mode	Execution Flow
Immediate	Commands that follow MC are invoked when the next MC command is executed for the first time only
Program	All commands that follow MC are immediately executed

Interactions

Modified by: H, RAMP, T, TA, TD, V, VS, VSCALE

Examples

Command	*Description*
`u>V3000 VS500`	‘Set parameters
`u>MC`	‘Begin moving continuously
`u>V6000`	‘Set new velocity
`u>MC`	‘Move at new velocity
`u>V4000`	‘Set new velocity and begin moving at the new rate

MGH — Motion Command

Causes motor to move to home position limit switch.

Execution Mode Immediate and Program

Syntax MGH

Description Causes the controller to seek the home position limit switch from either direction, depending on the direction specified by the H command.

- Home Position is defined as the position of a mechanical home limit switch.
- Encoder and position counters are reset to zero when the home limit switch is reached.

To gain more accuracy, MGH can be used in conjunction with the TIM timing signal and/or "z" channel signal of an encoder.

Diagrams which map the manner in which the system seeks the home position from various starting positions are shown in Figure 20 on page 68.

CAUTION For the MGH motion to operate safely and correctly, the limit switches (CCW, HOME, CW) must be connected to the controller. Injury or damage will occur if the limit switches are not connected.

Interactions Modified by: H, RAMP, T, TA, TIM, TD, V, VS, VSCALE
Modifies: EC, PC

Example

Command	*Description*
`u>V3000 VS500`	'Set velocity parameters
`u>H-`	'Set direction to CCW
`u>MGH`	'Move motor shaft to the Home position

Direction H+ Direction H-

Motion for when the device is positioned between the HOMELS and the CWLS.

Motion for when the device is positioned between the CCWLS and the HOMELS.

Motion for when the device is positioned on the HOMELS.

Motion for when the device is positioned on the CCWLS.

Motion for when the device is positioned on the CWLS.

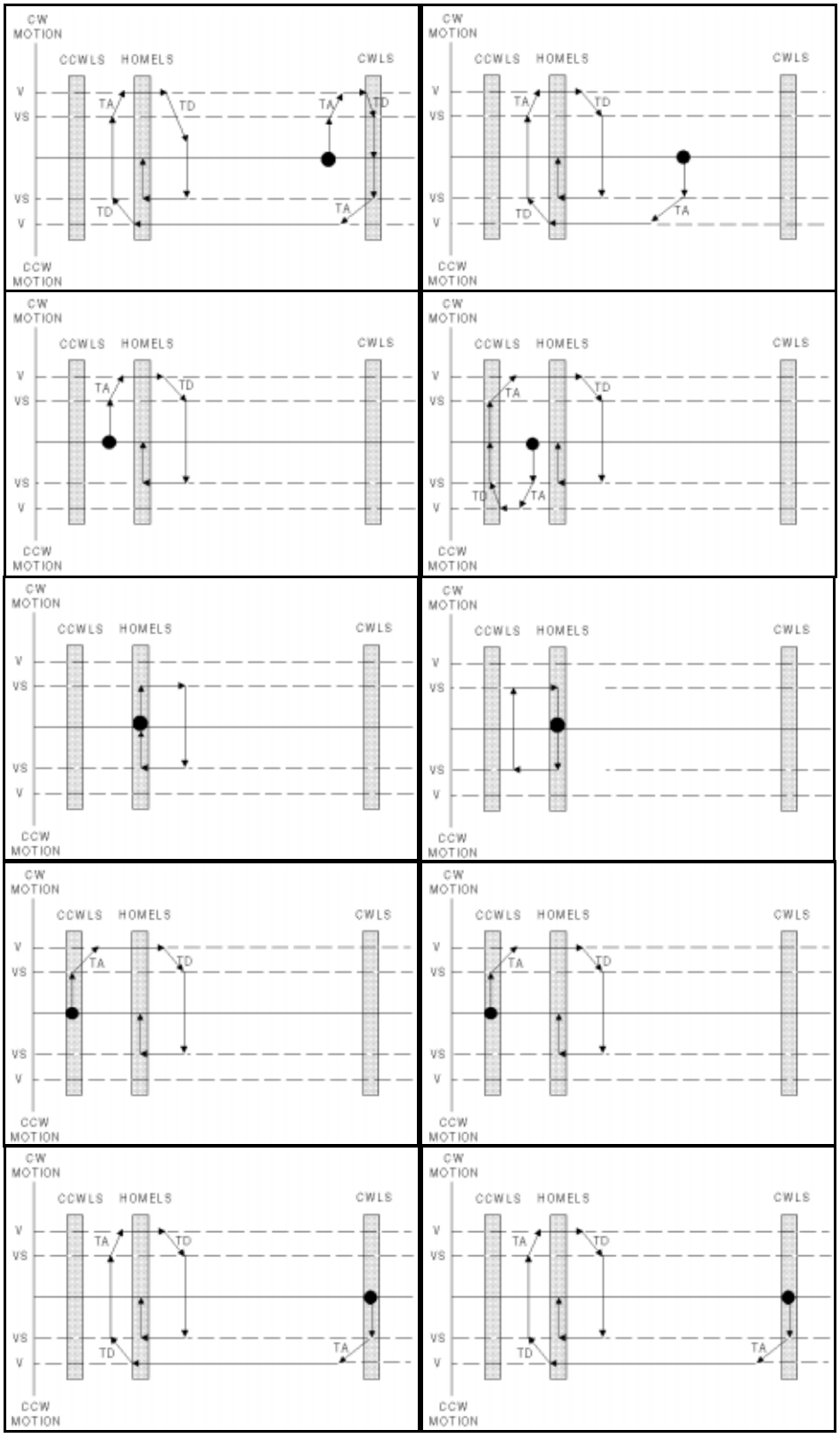


Figure 20 Home Hunting

Moves motor shaft a specified distance

Execution Mode Immediate and Program

Syntax MI

Moves the motor a distance established by the D parameter. The move is described as follows:

1. Specify the distance to move with D parameter.
2. Start moving initially at velocity VS.
3. Accelerate for time TA.
4. Run for period at velocity V.
5. Decelerate for time TD until reach VS velocity and stop.

The trapezoidal Speed/Time plot for this type move is shown in (Figure 21).

If the specified distance is not large enough to allow the motor to reach the specified velocity, V, then a triangular Speed/Time plot is created. Here the controller calculates V(peak). It begins deceleration as soon as V(peak) is attained. This is depicted in below in Figure 21(B).

Interactions Modified by: DSCALE, H, RAMP, T, TA, TIM, TD, V, VS, VSCALE

Example

Command	*Description*
`SEQ. 12:`	
`(1)V3000 VS500`	‘Set velocity parameters
`(2)H-`	‘Set direction to CCW
`(3)D18000`	‘Set distance to 18000 steps
`(4)MI`	‘Begin the index move
`(5)SAS FINISHED`	‘Echo that move is finished

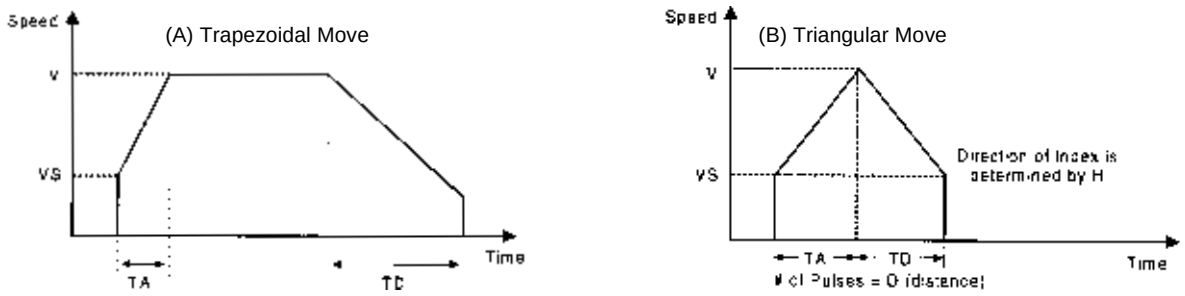


Figure 21 Speed/Time Profiles for (A) Trapezoidal and (B) Triangular Index Moves

MR — Motion Command Variable

Specifies the number of pulses per motor revolution (SC8800E only)

Execution Mode Immediate and Program

Syntax MR*n* Where *n* must be integer

Units Number of steps per revolution

Range 1 to 999, 999

Description Specifies the number of pulses per revolution for the motor. The MR parameter *must* be used when the controller is configured with an encoder. Without an encoder, MR is not valid.

Query/Response

```
MR?
0>mr?
0: MR = 5000
0>
```

Interactions Modifies: CP
Modified by: ER

Example

Command	*Description*
`SEQ. 20:`	
`(1)V3000 VS500`	‘Set velocity parameters
`(2)H-`	‘Set direction to CCW
`(3)ER2000`	‘500 line per revolution; 4 x 500 = 2000
`(4)MR500`	‘Set motor resolution to 500 steps/rev
`(5)D500`	‘Set distance to 1 revolution
`(6)MI`	‘Motor moves 1 revolution

MT — Motion Variable

Sets maximum amount of time for an Index Move

Execution Mode Immediate and Program

Syntax MTxx.xxx

Units Seconds

Range 0 to 64.99

Description Sets the maximum time allowed for index moves.
V and T are calculated and then executed.

NOTES

MT is valid under two conditions:

- Must be used with Index Move, the MI command
- MT can not equal zero

When MT is valid, the following rules apply for internal calculations:

- Maximum speed is set to 800,000 steps/second
- Ramp times, TA and TD, are forced to be symmetrical
- Motion variables D and VS are used with MT to determine acceleration time, deceleration time and final velocity

For the three types of RAMP profiles, these conditions apply:

Profile	No. of Segments	Segment Definition
RAMP0 (Linear)	3	3 equal times: t1=t2=t3=1/3rd MT
RAMP1(S-Curve)	3	t1=t3= time for acceleration and deceleration
RAMP2 (Parabolic)	2	2 equal times: t1 = t2 =1/2 MT t1 = t2 = time for acceleration and deceleration

See RAMP Command on 78 for drawings of the Ramp Profiles.

Query/Response

```
MT?
0>mt?
0: MT= 1.25
0>
```

Interactions Modifies: T, TA, TD, V, VSCALE, DSCALE, R

Example

Command	*Description*
`U>D20000`	'Set distance to 20000 steps
`u>MT0.8`	'Set move time to 0.8 seconds
`u>MI`	'Move 20000 steps in 0.8 seconds

OL System Control Command

Inverts the output levels

Execution Mode Immediate and Program

Syntax

OLb321

- Output #1
- Output #2
- Output #3 / FAULT
- b = BUSY output

Range

0 = non-inverted (Default condition)
1 = inverted

Description

Inverts the output levels of the three programmable outputs and the BUSY output signal. The status of OL is stored in the nonvolatile memory.

Query/Response

```
OL?
0>ol?
0: OL= 0101
0>
```

Example

Command	*Description*
`u>OL1010`	'Invert the Busy and Out#2 signals

Turns the three programmable outputs on and off

Execution Mode Immediate and Program

Syntax The OUT statement has two forms:

OUT*n*=x	OUTxxx
Single	Global
Set bit separately	Set all bits simultaneously

Outputs are defined as follows:

OUT321

- Output #1
- Output #2
- Output #3 / FAULT

Range x = 1 to 3
n defined by 0 = OFF; 1 = ON

CAUTION OUT3 is intended to be a system status output. If OUT3 is designated for some application function, the user will not be aware of a fault condition.

Description Turns the three programmable outputs on and off in either an individual or global manner.

Query/Response

```
OUT?
0>out?
0: OUT= 000
0>
```

Example

Command	*Description*
`SEQ. 13:`	
`(1)V3000 VS500 H-`	‘Set parameters
`(2)MI`	‘Do an Index Move
`(3)DELAY 2`	‘Delay 2 seconds
`(4)OUT=011`	‘Turn on outputs 1 and 2
`(5)DELAY 3`	‘Wait 3 seconds
`(6)OUT3=1`	‘Turn on output 3

PAUSE — Motion Command

Discontinues current motion or program

Execution Mode Immediate and Program

Syntax PAUSE

Description Discontinues the current motion or program by decelerating the motor to a stop. Paused programs are restarted under the following rules:

- Within a sequence at the next instruction by issuing the CONT(inue) command
- At the beginning of a program by issuing the RUN command

Example

Command	*Description*
`SEQ. 14:`	
`(1) D20000`	'Set distance to 20000 steps
`(2) IF(IN1=1) PAUSE`	'IF Input #1 is ON, PAUSE
`(3)  ELSE MGH`	'Otherwise, go home
`u> CONT`	'Continue the program

PC — System Variable

Sets the internal position counter

Execution Mode Immediate and Program

Syntax PC*integer*

Units Number of steps

Range ±999,999,999

Description Sets the internal position counter to the desired value.

Query/Response

```
PC?
0>pc?
0: PC+ 146239
0>
```

Interactions Modifies: CP
Modified by: DSCALE

Example

Command	*Description*
`u>PC0`	‘Set the position counter to 0
`u>MA -25000`	‘Move 25000 steps CCW direction
`u>PC?`	‘Check position counter value
`u:PC -25000`	‘Displays the current value of PC

PULSE — System Variable

Sets the type of pulses delivered by the controller to the motor driver

Execution Mode Immediate and Program

Syntax PULSE {1|2}

Range 1 = 1-Pulse mode; also referred to as *Step and Direction* mode.
2 = 2-Pulse mode; also referred to as *Up-Clock/Down Clock* mode.

Description Sets the type of pulses sent by controller to the motor driver.

- 1-Pulse (Step and Direction): The direction (CW/CCW) input being held high or low, causes the motor to rotate CW or CCW when pulses are received at the PULSE terminal. To avoid missed steps, the PULSE signal **must be inactive** when changing the direction (CW/CCW) signal.
- 2-Pulse (Up-Clock/Down-Clock): The motor will rotate in the CW direction when a step signal is received at the PULSE terminals and will rotate in the CCW direction when a step signal is received at the CW/CCW terminals. To avoid missed steps, the two step signals **must not be active** simultaneously.

Mode	Direction	Driver Connections	
		Direction Terminals	Pulse Terminals
1-Pulse (Step and Direction)	CW	High	Step Signal
	CCW	Low	Step Signal
2-Pulse (Up-Clock/Down-Clock)	CW	No Signal	Step Signal
	CCW	Step Signal	No Signal

CAUTION In 1-Pulse mode, PULSE signal must be inactive when changing DIRECTION signal. In 2-Pulse mode, Step Signals must not be active at the same time.

Example

Command	*Description*
`u>PULSE2`	‘Use 2-Pulse output mode
`u>MGH`	‘Move to home position

Shows current values of system status, system parameters and I/O states

Execution Mode Immediate and Program

Syntax R

Description Displays the current values of system status, system parameters and I/O states.

Example

Command	*Description*
u>R	'Report system status

```
o>r
Hardware status
CWL= 0 CCWL= 0
Home= 0 TIM= 01
Start= 0 Stop= 0
Prog.inputs= 0000
Prog.outputs= 000
Pulse mode= 1
Motion parameters:
VS= 5 V= 100000 VSCALE= 0
TA= 1.0 TD= 1.0 MT= 0.0
H(Dir)= + D= 100000 DSCALE= 0
MR= 5000 ER= 2000 RAMP= 0
TIM= 01 LIM= 0 0
Position:
PC= 149185
EC= 320964
System status: Busy
0>
```

RAMP — Motion Variable

Specifies type of acceleration and deceleration ramp

Execution Mode Immediate and Program

Syntax RAMP*n*

Range 0 to 2 Where ramp type is defined as:
0 = linear
1 = s-curve
2 = parabolic

Description Specifies the type of acceleration and deceleration ramp to use during motion. Three ramp types are available. The RAMP value is stored in nonvolatile memory (Figure 22).

Query/Response

```
RAMP?
0>ramp?
0: RAMP= 0
0>
```

Interactions Modified by: T, TA, TD, V, VS, VSCALE
Modifies: MT (See Figure 23 and refer to page 71.)

Example

Command	*Description*
`u>RAMP1`	‘Set ramp type to s-curve
`u>MI`	‘Execute an index move
`u>RAMP0`	‘Set ramp type to linear
`u>MGH`	‘Go home

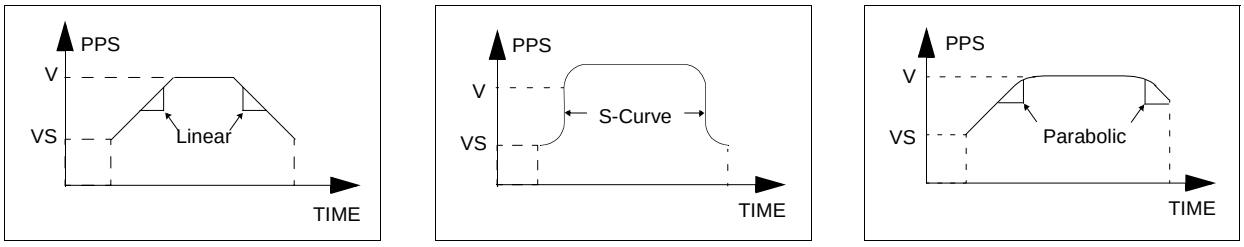


Figure 22 Linear, Parabolic and S-Curve Ramps

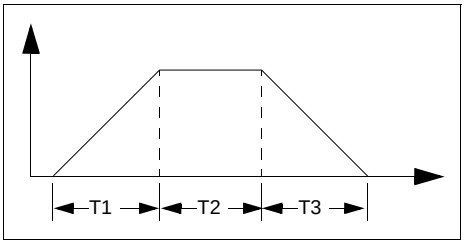


Figure 23 Segment Definition

REM — Program Control Element

Allows for insertion of program comment

Execution Mode Immediate and Program

Syntax Two forms are valid:

- REM [ASCII string]
- #[ASCII string]

Units ASCII character

Range ASCII string: {A to Z, 0 to 9}

Description Allows the programmer to insert comments into the program code to more clearly describe and explain the program logic. The following rules apply:

- Program lines that begin with REM or # are not executable.
- Commented lines are shown when a program is displayed via LIST command.

Example

Command	*Description*
`Seq. 3:`	
`(1)REM SET PARAMETERS`	‘Insert comment
`(2)H-`	‘Set direction to CCW
`(3)MGH`	‘Go home
`(4)# RESET COUNTERS`	‘Comment the program
`(5)PCO  ECO`	‘Reset counter values

RESET — System Control Command

Allows software to perform a system reset

Execution Mode Immediate and Program

Syntax RESET

Description Executes a system reset via software

NOTE

There are two equally valid ways to reset the controller:

- Power cycle the hardware by turning the power Off and then On
- Issue a RESET command
- Only unit 0 will return a sign on banner. All other ID values will perform the hardware reset, but not return the sign on banner.

Interactions Modified by: ID

Example

Command	*Description*
`u>RESET`	‘Reset the unit

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                  SC8800(E)                          *
*             Programmable Indexer                    *
*             Software Version 3.0                    *
*              Copyright 1995-1999                    *
*      ORIENTAL MOTOR U.S.A. CORPORATION              *
*             Torrance, California                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

RET — Program Control Element

Allows program control to revert to the main sequence

Execution Mode Program

Syntax RET

Description Allows program control to revert to the main sequence from a subroutine. It is not required, but provided for readability only.

Interactions Modified by: CALL

Example

Command	*Description*
`Seq. MAIN`	
`(1) PC0`	‘Set the Position Counter to Zero
`(2) MI`	‘Do an Index Move
`(3) CALL TEST`	‘Execute Sequence named “Test”
`(4) SAS I’M BACK`	‘Echo Return to Original Sequence
`Seq. Test:`	
`(1) OUT1=1`	‘Turn On Output #1
`(2) DELAY2`	‘Delay 2 seconds
`(3) OUT1=0`	‘Turn Off Output #1
`(4) RET`	‘Return to MAIN sequence

RUN — System Control Command

Executes any stored sequence

Execution Mode Immediate

Syntax RUNxx{sequence name}

Range xx can be from 0 to 49

Description Executes any stored sequence.

Example

Command	*Description*
`u>RUN TEST`	'Run the program "TEST"
`u>RUN33`	'Run program designated as sequence 33

Terminates motion immediately using current deceleration rate parameters

Execution Mode Immediate and Program

Syntax S

Description Terminates movement by decelerating the motor at the rate determined by parameters T or TD.

Interactions Modified by: T, TD, MT

Example

Command	*Description*
u>MC	‘Move continuously
u>S	‘Decelerate and stop

Directs an ASCII string to the terminal screen

Execution Mode Program

Syntax SAS[ASCII string] optional {\code}

Units ASCII character

Range ASCII string: = {A to Z, 0 to 9}
Optional \code = 0, W, X, Y, Z

Description Directs an ASCII string followed by a carriage return to the terminal screen. If no ASCII string is supplied, only the carriage return is sent. The optional \code qualifier has 2 applications:

- When **0** is placed at the end of the ASCII string, the carriage return is suppressed.
  This format is used when requesting a keyboard entry into a running program.
- The **\{W, X, Y, Z}** is used to imbed the current value of a program variable anywhere in the SAS line.

Example

Command	*Description*
`SEQ. 4`	
`(1) V3000 VS500`	‘Set velocity parameters
`(2) H-`	‘Set direction to CCW
`(3) D9000`	‘Set distance to 9000 steps
`(4) SAS BEGIN MOTION`	‘Send string “Begin Motion” to display
`(5) MI`	‘Execute an index move
`(6) SAS END OF MOVE`	‘Send string “End of Move” to display
`SEQ. 5`	
`(1) SAS Enter number of cans: \0`	‘Prompt for keyboard input, suppress CR
`(2) Y = KB` `...`	‘Set Y = to value entered
`(9) SAS there are \x boxes to go`	‘String reports how many units are left

Terminates motion immediately without deceleration

Execution Mode Immediate and Program

Syntax STOP

Description Stops the motor immediately without the use of any deceleration.

CAUTION The motor may lose steps when stopped in this manner.

Example

Command	*Description*
`u>MC`	‘Move continuously
`u>STOP`	‘Stop immediately

SYS System Control Command

Displays the total operating time and the number of resets

Execution Mode Immediate and Program

Syntax SYS

Description Displays the total system operating time and the number of system resets.

Interactions Modified by: CLEAR_NVR

Example

Command	*Description*
`u>SYS`	‘Display system operating times
`0>sys`	‘Decelerate and stop
`0: Total operating time (hr:min)=35:53`	
`0: Total resets= 53`	
`0>`	

Specifies the amount of time for acceleration and deceleration

Execution Mode Immediate and Program

Syntax T*number*

Units Seconds

Range 0.001 to 64.999

Description Specifies the amount of time used for *both* acceleration and deceleration when executing the move commands:

MA, MC, MGH and MI.

Setting the variable T results in variables TA and TD being equal.

Query/Response

```
T?
0>t?
0: TA = 1.2 TD = 1.2
0>
```

Interactions Modifies: MA, MC, MGH, MI
Modified by: MT

Example

Command	*Description*
`SEQ. 17:`	
`(1) PCO`	‘Set the position counter to zero
`(2) T2`	‘Set accel & decel time parameter to 2 seconds
`(3) VS500 V5000`	‘Set velocity parameters
`(4) D15000`	‘Set distance to 15000 steps
`(5) MI`	‘Execute an index move

TA — Motion Variable

Specifies the amount of acceleration time

Execution Mode Immediate and Program

Syntax TA*number*

Units Seconds

Range 0.001 to 64.99

Description Specifies the amount of time used for acceleration when executing the move commands:

MA, MC, MGH and MI

Interactions Modifies: MA, MC, MGH, MI
Modified by: MT

Example

Command	*Description*
`SEQ. 12:`	
`(1) PCO`	‘Set the position counter to zero
`(2) TA2`	‘Set acceleration to 2 seconds
`(3) TD2.3`	‘Set deceleration time to 2.3 seconds
`(4) VS500 V5000`	‘Set velocity parameters
`(5) D15000`	‘Set distance to 15000 steps
`(6) MI`	‘Execute an index move

TALK — System Control Command

Makes logical connection to unit in a multiple unit configuration

Execution Mode Immediate

Syntax TALK{u}

Range u = {0 to 9, a to z}

Description Makes logical connection to a specific unit in a multiple unit, e.g. daisy chain configuration. Then the specific unit can be uniquely addressed and programmed. If the unit ID is anything other than 0, the unit will not operate properly unless the proper TALK command is issued.

Interactions Modified by: ID, RESET

Example

Command	*Description*
`1>MGH`	'Unit 1 go home
`1>TALKC`	'Talk to Unit C
`C>MGH`	'Unit C go home

TD — Motion Variable

Specifies the amount of deceleration time

Execution Mode Immediate and Program

Syntax TD*number*

Units Seconds

Range 0.001 to 64.99

Description Specifies the amount of time used for deceleration when executing the move commands:

MA, MC, MGH and MI

Interactions Modifies: MA, MC, MGH, MI
Modified by: MT

Example

Command	*Description*
`SEQ. 12:`	
`(1) PCO`	‘Set the position counter to zero
`(2) TA2`	‘Set acceleration to 2 seconds
`(3) TD2.3`	‘Set deceleration time to 2.3 seconds
`(4) VS500 V5000`	‘Set velocity parameters
`(5) D15000`	‘Set distance to 15000 steps
`(6) MI`	‘Execute an index move

Specifies which signals should be used when executing an MGH move

Execution Mode Immediate and Program

Syntax

TIMtz
- z= Encoder "z" channel
- t= Timing signal

Range

0 = disable
1 = enable

Description Specifies if the driver Timing output signal and/or encoder "z" channel output are used when executing the MGH move.

NOTE To achieve a more accurate home position when executing the MGH command, the TIM command enables the logical combination of timing and/or "z" signals.

Query/Response

```
TIM?
0>tim?
0: TIM = 01
0>
```

Interactions Modifies: MGH

Example

Command	*Description*
`u>TIM10`	'Use the timing signal in MGH move
`u>MGH`	'Go home

TR — Editor Command

Displays the list of instructions as they are being executed

Execution Mode Immediate and Program

Syntax TR*n*

Range
0 = trace Off
1 = trace On

Description Displays the list of instructions within a sequence as they are being executed. This is used for debugging purposes.

Example

Command	*Description*
`u>TR1`	‘Turn trace mode On
`u>RUN TEST`	‘Run TEST program

UNLOCK — Editor Command

Resets access to previously locked sequence

Execution Mode Immediate

Syntax UNLOCK {[xx]}|[sequence name]}

Range Any available sequence

Description Resets the lock bit on a specified sequence so that it can be altered or deleted. Once the sequence is unlocked, the indicator asterisk is removed from the sequence name when a DIR or LIST command is issued.

Interactions Modifies: LOCK

Example

Command	*Description*
`u> LOCK TEST`	‘Lock sequence TEST
`u> EDIT TEST`	‘Attempt to edit TEST
`u>**Seq LOCKED (NOT alterable)`	‘Message
`u> UNLOCK TEST`	‘UNLOCK the sequence

V Motion Variable

Specifies the run velocity

Execution Mode Immediate and Program

Syntax V*number*

Units Pulses per second

Range 1 to 800,000

Description Specifies the velocity to use when executing move commands:

MA, MC, MGH and MI

All moves start at the value of VS and then accelerate to V at the rate set by parameters T or TA.

Query/Response

```
V?
0>v?
0: V= 30000
0>
```

Interactions Modifies: MA, MC, MGH, MI
Modified by: VSCALE, MT

Example

Command	*Description*
`SEQ. 5`	
`(1) V3000 VS500`	‘Set velocity parameters
`(2) H-`	‘Set direction to CCW
`(3) D9000`	‘Set distance to 9000 steps
`(4) SAS BEGIN MOTION`	‘Send string “Begin Motion” to display
`(5) MI`	‘Execute an index move
`(6) SAS END OF MOVE`	‘Send string “End of Move” to display

Specifies the starting velocity

Execution Mode Immediate and Program

Syntax VS*number*

Units Pulses per second

Range 1 to 800,000

Description Specifies the initial velocity when executing move commands:
MA, MC, MGH and MI
All moves start at the value of VS and then accelerate to V at the rate set by parameters T or TA.

Query/Response

```
VS?
0>vs?
0: VS= 500
0>
```

Interactions Modifies: MA, MC, MGH, MI
Modified by VSCALE

Example

Command	*Description*
`SEQ. 5`	
`(1) V3000 VS500`	‘Set velocity parameters
`(2) H-`	‘Set direction to CCW
`(3) D9000`	‘Set distance to 9000 steps
`(4) SAS BEGIN MOTION`	‘Send string “Begin Motion” to display
`(5) MI`	‘Execute an index move
`(6) SAS END OF MOVE`	‘Send string “End of Move” to display

VSCALE — Motion Variable

Defines a velocity scaling factor for use with motion commands

Execution Mode Immediate and Program

Syntax VSCALE*number*

Units Pulses per second (or as redefined by VSCALE)

Range 0 to 2,147,483,647

Description Modifies the velocity parameter V by a specified factor. The value of VSCALE can be used to represent important application parameters like revolutions per minute or steps per hour.

NOTE The controller calculates actual velocity by the following equation:
V (steps/sec) = V (input) * VSCALE

Query/Response

```
VSCALE?
0>vscale?
0: VSCALE= 0
0>
```

Interactions Modifies: MA, MC, MGH, MI, MT, V

Example

Command	*Description*
`u>VSCALE500`	‘Set VSCALE = 500 pulses per second
`u>V1`	‘Set velocity to 1 revolution per second
`u>MC`	‘Motor will move at 1 revolution per second

WHILE — Program Control Element

Initiates conditional testing for a looping program segment

Execution Mode Program

Syntax WHILE (Variable 1 {Conditional Operator} Variable 2 or constant)

Description Provides for conditional looping using the following rules:

- An ENDW statement must complete the WHILE loop.
- All instructions between WHILE and ENDW are repeated until the condition of the WHILE statement becomes false.
- WHILE and ENDW loops may be nested up to 4 times.

=	‘Equal to
!=	‘Not equal to
<	‘Less than
<=	‘Less than or equal to
>	‘Greater than
>=	‘Greater than or equal to

Valid variable elements are:

IN	‘4 programmable input bits (condition must always be = or !=)
PC	‘Position Counter value
EC	‘Encoder Counter value
CP	‘Compare Encoder and Position counter values
V	‘Velocity
D	‘Distance
W, X, Y, Z	‘General purpose variables

Interactions Modified by: ENDW

Example

Command	*Description*
`SEQ. 9:`	
`(1) WHILE (PC<25000)`	‘WHILE position counter is less than 25000
`(2) SAS MOTOR OK`	‘Echo a message
`(3) MC`	‘Move continuously
`(4) ENDW`	‘End the WHILE loop
`(5) MGH`	‘Go home

W, X, Y, Z — Motion Variable

Program Variables

Execution Mode Program

Syntax {W|X|Y|Z} = xxxx or KB — Decimal values are not valid
KB refers to *keyboard* entry during a running program

Range {W|X|Y|Z} = ± 2, 147, 483, 647

Description There are two forms of program variables, integer and fixed point. Both forms can be extended in their operation by using the keyboard entry facility.
Integer program variables are:

W, X, Y and Z

They are used in sequences as:

- Loop counters
- Statement values
- Parts of mathematical expressions

Query/Response

```
W?, X?, Y?, Z?
0>w?, x?, y?, z?
0: W = 0
0: X = 1000000
0: Y = 8000000
0: Z = 0
0>
```

Example

Command	*Description*
`SEQ. 4`	
`(1)X=1 Y=PC-X`	‘Set X and Y values
`(2)WHILE (X<20)`	‘Begin WHILE loop
`(3) D500 MI`	‘Do a 500 step move
`(4) X=X+1`	‘Increment the loop counter
`(5) ENDW`	‘End WHILE segment

Displays preceding parameter continuously

Execution Mode Immediate and Program

Syntax PC/ EC/ CP/ V/ IO/

Description A forward slash character (/) following certain variables causes the system to continuously display the value of those elements utilizing these rules:

- The five variables designated for continuous display are:
  PC, EC, CP, V, IO
- Up to four may be displayed simultaneously
- This data is updated every 0.2 seconds
- Keyboard inputs <ENTER> or <ESC> terminates the display loop
- Within a sequence, the CLR command turns off the display loop
- Regardless of order entry, variables are displayed in the following order:

1. PC = Position counter value
2. EC = Encoder position counter value
3. CP = Check position counter value against encoder counter value
4. V = Velocity
5. IO = Input/output status

CAUTION Do not confuse this special command with the division operator.

Example

Command	*Description*
`u>D2000`	‘Set distance to 2000 steps
`u>PC/V/`	‘Display position counter and velocity
`u>MI`	‘Execute an index move

\ Keystroke Command

Global command indicator

Execution Mode Immediate

Syntax \{command}

Range All valid commands

Description The backslash character (\) preceding a command causes the system to issue that command to all controllers in a daisy chain configuration. This global designation overrides ID and TALK commands that are directed to individual units.

Interactions Modifies: ID, TALK

Example

Command	*Description*
`u>TALK 2`	'Talk to unit #2
`2>MI`	'Unit #2 execute and index move
`2>\MGH`	'All units go home

Backspace (Destructive Backspace)

Execution Mode Immediate

Syntax <BKSP>

Description Pressing the <BKSP> key causes the system to perform a destructive backspace in the system buffer. It is assumed that the terminal, when receiving this character, will do the same. Thus, the terminal and system buffer will contain the same data.

<ENTER> Keystroke Command

Commands system to accept current line

Execution Mode Immediate

Syntax <ENTER>

Description Pressing the <ENTER> key completes the data input for the current line and causes the system to accept the information on the line.

NOTE

Not all terminals or communication packages have a single <ENTER> key/character facility.
The system will accept the following keystrokes as equivalent to <ENTER>:

<CR>
<LF + CR>
<CR + LF>

<Esc> Keystroke Command

Discards current line

Execution Mode Immediate

Syntax <Esc>

Description The <Esc> key has two meanings:

- When entering data from the keyboard, pressing <Esc> causes the information on the current line to be discarded.
- When a motion command is being executed, pressing <Esc> immediately causes the motor to stop. Current deceleration parameters are invoked.

Example

Command	*Description*
`u>ECO`	‘Set encoder to zero
`u>D2000`	‘Set distance parameter to 2000 steps
`u>V500 <Esc>`	‘Discard this line
`u>MC`	‘Perform a continuous move
`u><Esc>`	‘Decelerate and stop the move

Appendix A Quick Start Guide

This Guide enables you to connect and test the SC8800/SC8800E controller in a basic stepping motor system. Refer to the other sections in this Operating Manual for specific information.

CAUTION

- To ensure your *safety* and to prevent any equipment *damage,* observe all wiring conventions and environmental precautions. See page 6.
- Carefully observe all DC voltage polarities. See page 5.
- Do NOT apply power to any system components until Step number 7 below.

Check that you have the following items:

SC8800/SC8800E Components

- SC8800/SC8800E controller
- Dual ended serial cable
- D37 connector and housing
- Operating Manual

Customer Furnished Components

- Stepping Motor
- Stepping Motor Driver
- Terminal or PC with communications software such as ProComm® or Windows® Terminal
- Power Source #1: 10 to 28 VDC; 3 watts max.
- Power Source #2: Applicable for Motor/Driver

1. Connect controller's output terminals, +P/CW, -P/CW, +D/CCW and -D/CCW to the step and direction inputs of the motor/driver. Details: page 5.
2. Connect the stepping motor lead wires to the appropriate terminals of the motor/driver. Details: page 3.
3. Connect power source #1 to the controller's power input terminals. Details: page 5.
4. Connect power source #2 to the proper input terminals on the motor/driver. Details: page 3
5. Connect the 9 pin male end of the cable supplied into CN-1 of the controller. Connect either the 9-pin or 25-pin end of the cable into the RS-232C serial port of your computer or terminal. Details: page 7.
6. Start your communications program. Set the communications parameters to the following:

Baud rate = 9600	Parity = NO
Data bits = 8	Full Duplex
Stop bit = 1	Handshake = NO

7. Turn on the power to the SC8800/SC8800E. A sign on banner and a prompt will appear on the screen.
8. Turn on the power to the motor/driver.
9. Type **R** to view the current values of the motion parameters. Change any parameters needed and type **MI**. The motor should execute a move representing the values that were input. Details: page 17.
10. Type **HELP** to learn about the other available commands. Details: page 50.

Appendix B Specifications

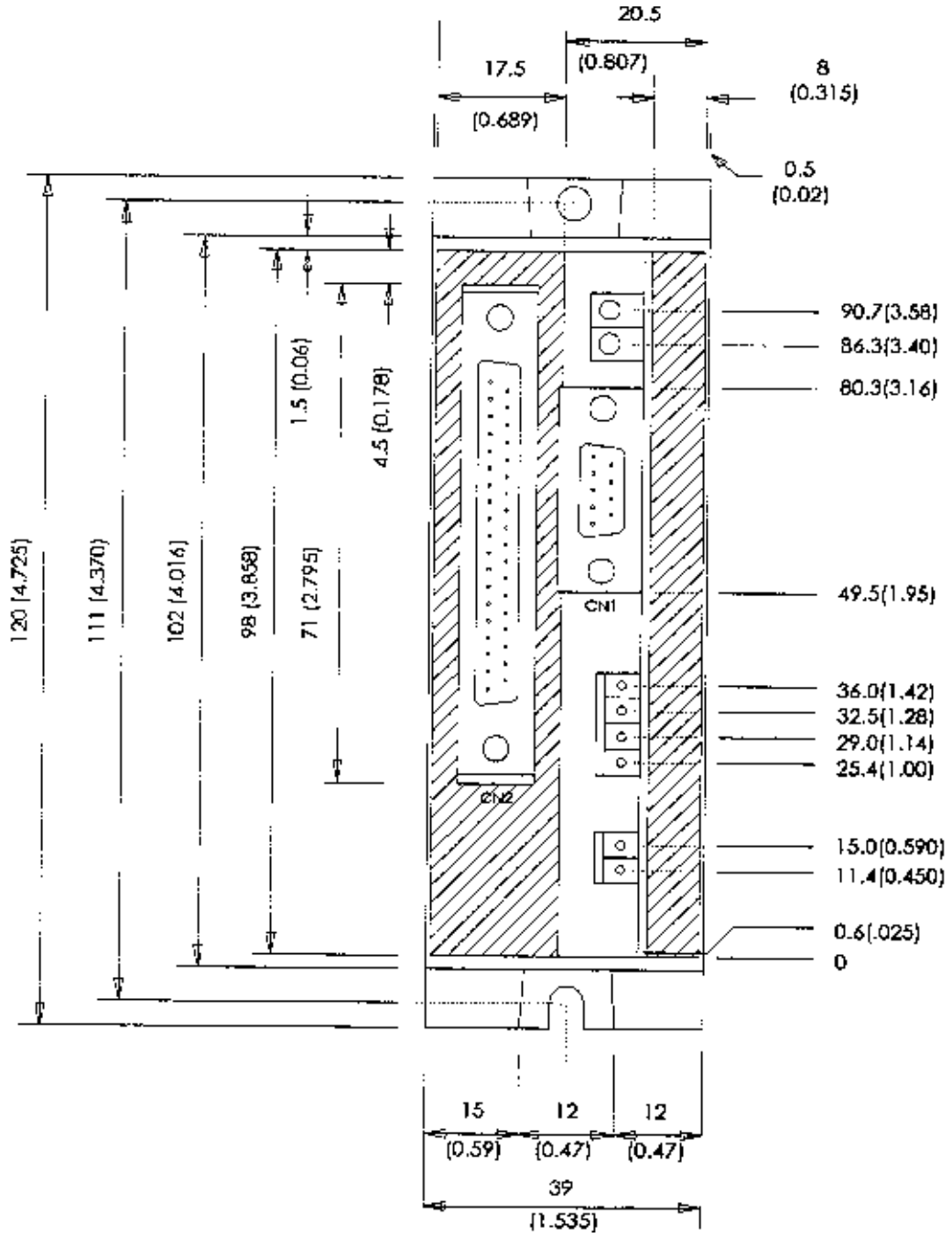
PARAMETER		VALUE
General		
	Part Number	SC8800, SC8800E, (E = Encoder interface option)
Input Power		
	Voltage	10 to 28 VDC 3.0 watts max.
Performance		
	Stepping Accuracy	± 0 steps from preset total
	Velocity Accuracy	± 0.05% of preset rate
	Velocity Repeatability	± 0.01% of maximum rate
	Position Range	0 to ± 999,999,999 steps, when DSCALE is active
	Velocity Range	1 to 800,000 steps/sec
	Acceleration Rate	0.001 to 10 sec
Motion Types		
	Absolute	Move to specified internal counter position
	Index	Move specified distance
	Continuous	Move specified speed until commanded to stop
	Go Home	Move to Home limit switch
	Move Time	Move specified distance in specified time
Sequence Execution		
	Via RS-232C	Sequence may be executed from RS-232C interface
with RUN command		
	Via Power-up Auto Run	Execute any sequence, 0~15, upon power-up
	Via Programmable Input	Sequences may be selected using an external device
Programming Language		Simple, high level programming language
Nonvolatile Memory		
	Sequence Length	8 KB or up to available remaining memory
	Number of Programs	50 max. or up to available memory

Specifications (Cont.)

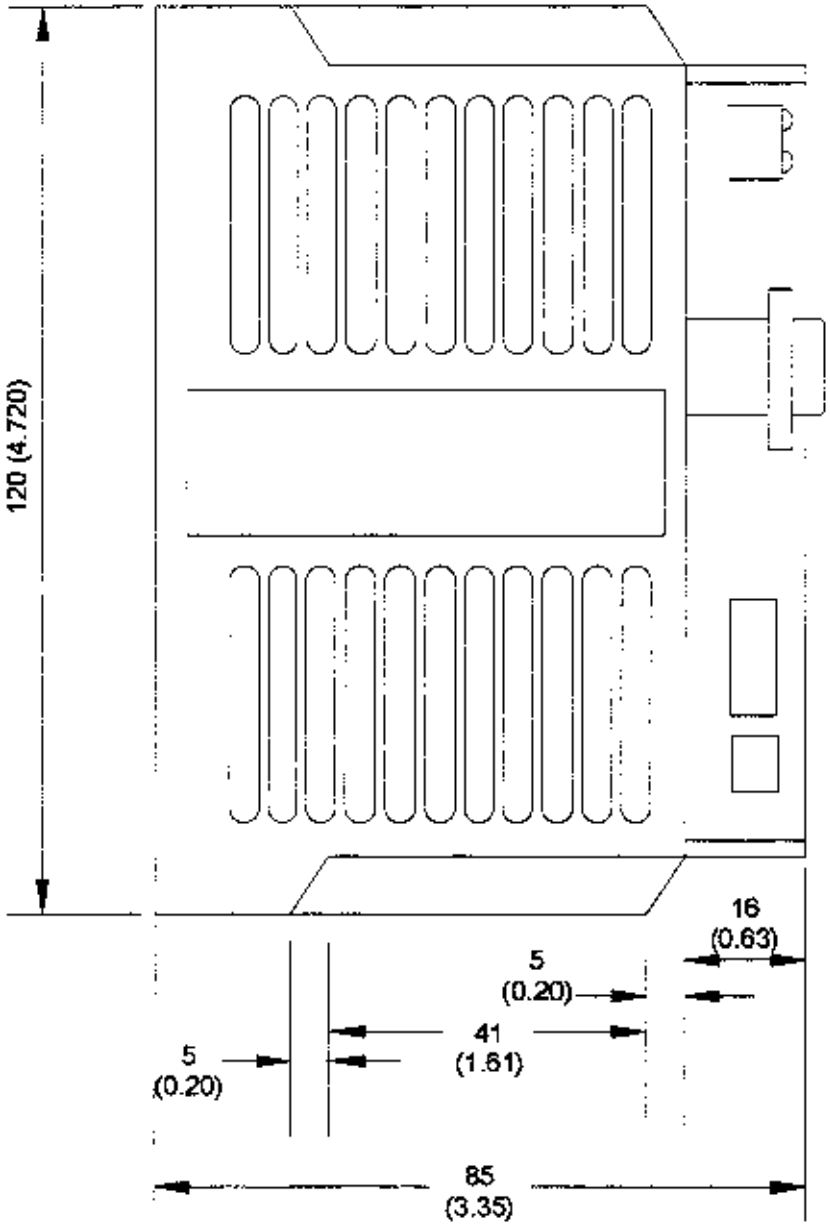
PARAMETER		VALUE
Inputs		
	Command Interface Type	RS-232C serial type, three wire implementation (TX, RX, Gnd)
	Parameters	Baud rate fixed at 9600, 8 data bits, 1 stop bit, no parity
	Configuration	35 units max. can be controlled by a single port of daisy chain configuration.
	CW, CCW, Home Limits	+5 to +30 VDC, optically isolated, max. current 10 mA
	Programmable Inputs	Four to be used for machine interaction and/or sequence selection, +5 to +30 VDC, optically isolated Max. current, 10 mA
	TIM	Phase 0 indicator, +5 to +30 VDC, optically isolated
	Encoder	Model SC8800E accepts 2 or 3 channel, two phase quadrature incremental encoders with differential or single ended outputs, 5 VDC TTL compatible, 400 kHz (quadrature), max.
Outputs		
	Step and Direction, Motion	TTL, High 4 to 5 VDC, Low 0 to 0.5 VDC, Pulse width 0.5 µsec min., Rise/Fall time: 0.2 µsec max.
	Programmable Outputs	Two, open collector, 5 to 24 VDC, 80 mA max.
	Status Outputs	Fault & Busy, Open collector, 5 to 24 VDC, 80 mA max.
Mechanical		
	Dimensions	3.35" L x 1.57“ W x 4.72“ H
	I/O connectors	Combination of fixed screw terminal and D-type
Environmental		
	Cooling Method	Natural convection
	Ambient Temperature Range	+32 °F ~ +122 °F (0 °C ~ +50 °C)
	Humidity	0 to 95% noncondensing
	Weight	11 oz. (0.31 kg)

Appendix C Dimensions

DIMENSIONS:
Units = MM (INCHES)



DIMENSIONS (Continued):
Units = MM (INCHES)



Appendix D Trademarks

Windows is a registered trademark of Microsoft Corporation.
ProComm is a registered trademark of ProComm Telecommunications, Inc.

Appendix E Transferring Sequences

Sometimes it may be necessary or convenient to transfer programs stored in one SC8800/SC8800E to another. Oriental Motor offers a software solution for this function. This function may be useful when building several identical machines.
To receive a free copy of this software, please contact Oriental Motor Technical Support at:

Phone: 1-800-468-3982
Email: Techsupport@orientalmotor.com

Appendix F Default Values

Listed below are the default values for the system parameters and I/O states. These values will be reset whenever a CLEAR_NVR command is issued.

```
Hardware status
CWL= 0 CCWL= 0
Home= 0 TIM= 0O
Start= 0 Stop= 0
Prog.inputs= 0000
Prog.outputs= 000
Pulse mode= 2
Motion parameters:
VS= 5 V= 10     VSCALE= 0
TA= 0.5 TD= 0.5 MT= 0.0
H(DIR)= + D= 0     DSCALE= 0
MR= 0   ER= 0   RAMP= 0
TIM= 00 LIM= 0.0
Position:
PC= 0
EC= 0
System status: Idle
```

ORIENTAL MOTOR U.S.A. CORP.

Atlanta:
TEL: (770) 716-2800
FAX: (770) 716-8515

Austin:
TEL: (512) 918-9438
FAX: (512) 335-5983

Boston:
TEL: (781) 848-2426
FAX: (781) 848-2617

Chicago:
TEL: (847) 240-2649
FAX: (847) 240-2753

Cincinnati:
TEL: (513) 563-2722
FAX: (513) 956-3183

Los Angeles:
TEL: (310) 784-8200
FAX: (310) 325-1076

New Jersey:
TEL: (973) 882-0480
FAX: (973) 740-0693

San Jose:
TEL: (408) 358-6900
FAX: (408) 358-8200

Canada:
TEL: (513) 563-2722
FAX: (513) 956-3183

Technical Support Line

TEL: 1-800-468-3982
8:30 AM to 8:00 PM EST
Email: Techsupport@orientalmotor.com