INTRODUCTION TO THE MODEL ONE/10 MANUAL

February 10, 1986

Components of the Model One/10

The Model One/10 Manual contains the following documents which explain how to use the Model One/10 system:

- Model One/10 Introduction and Installation Guide
- Model One/10 Command Reference
- Model One Error Messages Reference Guide
- Tektronix 4014 Emulation on the Raster Technologies Model One Graphics Controller
- Model One/10 and Model One/80 Programming Reference Card (2)

Release Notes

In addition to this Raster Technologies Model One/10 Manual, you should refer to the Model One/10 Release Notes, which explain the enhancements incorporated in the version of firmware included in your Model One/10. New features and any inaccuracies discovered since the printing of the Model One/10 Manual are documented in the Release Notes.

Raster Technologies

MODEL ONE/10 INTRODUCTION AND INSTALLATION GUIDE

Revision 2.0

Part # 552-00001-002

February 11, 1986

RASTER TECHNOLOGIES
MODEL ONE

This document corresonds to Version 2.1 of the Model One/10 firmware.

NOTICE:

The information contained in this document is subject to change without notice.

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual may cause interference with to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Table of Contents

Section Page

Part I. Introduction to the Model One/10

1. Using This Manual....1-1

2. Overview of the Model One/10 2-1

2.1 Firmware....2-2
2.2 Applications Development Features....2-4
2.3 System Architecture....2-5

3. Model One/10 Firmware....3-1

3.1 Communications....3-1
3.2 Alpha and Graphics Modes....3-1
3.3 Host Serial Communications Command Format....3-2
3.4 Local Command Format....3-2
3.5 Diagnostic and Development Features....3-2
3.6 Coordinate System....3-3
3.7 Coordinate Registers....3-4
3.8 Clipping Windows....3-6
3.9 Graphics Primitives and Area Fills....3-9
3.10 Pixel Value Registers....3-11
3.11 Defining and Executing Macro Commands....3-12
3.11.1 Using the Button Table to Execute Macros....3-13
3.12 Display List Firmware....3-14

4. Overview of the Model One/10 Display List Firmware....4-1

4.1 Display List Structures....4-1
4.1.1 Segments: Characteristics and Storage....4-1
4.1.2 Pick IDs....4-3
4.1.3 Segment Attributes: Visibility and Pickability....4-4
4.1.4 Segment Identity Information....4-5
4.1.4.1 Registers....4-5
4.1.4.2 Counters and Pick Buffer....4-5
4.1.4.3 Reading Back Segment Information....4-5
4.2 Creating and Modifying Segments....4-5
4.2.1 Defining Segments....4-6
4.2.2 Modifying Segments....4-6
4.3 Executing and Nesting Segments....4-6
4.3.1 Executing Segments....4-7
4.3.2 Nesting Segments....4-7

Table of Contents
(Continued)

Section Page

4.4 Execution Modes..4-9
  4.4.1 Normal Draw Mode..4-9
  4.4.2 Pick Mode..4-9
  4.4.3 Highlight Mode..4-9
  4.4.4 Unhighlight Mode..4-10

4.5 Picking...4-10
  4.5.1 Specifying the Center of the Pick Aperture..........4-11
  4.5.2 Specifying the Size of the Pick Aperture............4-11
  4.5.3 Specifying the Type of Information to Store in the Pick Buffer....................................4-11
  4.5.4 Entering Pick Mode.......................................4-12
  4.5.5 Executing (Picking) Segments..........................4-12
    4.5.5.1 Pick Buffer Overflows.........................4-12
  4.5.6 Reading Back Pick Information.......................4-13
    4.5.6.1 Using the RDPICK Command Only for Updating Registers...........................4-15
  4.5.7 Entering Normal Draw or Highlight Mode..............4-16
  4.5.8 Using Picking...4-16
4.6 Local Highlighting..4-16
4.7 Transformations..4-19
  4.7.1 Types of Transformations...............................4-19
    4.7.1.1 What You Have to Specify....................4-19
    4.7.1.2 Default Transformation: The Identity......4-20
    4.7.1.3 Translation With No Scaling or Rotation: XFORM2D XLATE...............................4-20
    4.7.1.4 Absolute Transformation: XFORM2D ABS......4-20
    4.7.1.5 Relative Transformation: XFORM2D REL......4-21
  4.7.2 Reading Back an Using The Current Transformation....4-22
  4.7.3 More About the Current Point and Transformations....4-22
4.8 Stacking..4-23
  4.8.1 Pushing and Popping Transformations.................4-24
  4.8.2 Pushing and Popping Transformations: The Current Point...4-24
4.9 Display List Firmware Commands by Functional Category......4-26

Table of Contents
(Continued)

Section Page

5. FORTRAN Library....................5-1

5.1 The Model One/10 FORTRAN Library....................5-1
5.1.1 Conventions....................5-2
5.2 Installing the FORTRAN Library on the VAX....................5-2
5.2.1 Documentation Included on the Library....................5-2
5.2.1.1 Suggestion....................5-2
5.2.2 How to Install the FORTRAN Library....................5-3
5.2.2.1 Question 1: Copying Files from Tape....................5-3
5.2.2.2 Question 2: Default Device Type....................5-4
5.2.2.3 Question 3: Optional Model One/380 Demonstration Program....................5-4
5.2.3 Results of Using the INSTALL Command....................5-4
5.2.3.1 Sub-directories....................5-5
5.2.3.2 Command Procedures....................5-5
5.3 Using the VAX FORTRAN Library....................5-5
5.3.1 Compiling User Programs....................5-6
5.3.1 Linking the User Programs....................5-7
5.3.2 Specifying the Model One/10 for Serial I/O....................5-7
5.4 Verifying the VAX Library....................5-7

6. Central Processor....................6-1

6.1 Processor Memory....................6-2
6.2 Input/Output Ports....................6-2
6.3 Control and Status Registers....................6-2

7. Vector Generator....................7-1

7.1 Pattern Register....................7-1

8. System and Image Memory....................8-1

8.1 Write-enable Masks....................8-1
8.2 Read Masks....................8-1
8.3 Image Memory Output....................8-1
8.3.1 Look-up Tables....................8-2
8.3.2 Video Amplifiers....................8-3
8.4 Using the LUT16, RMSK16, VAL16, and WRMSK16 Commands....................8-3

Table of Contents
(Continued)

Section Page

Part II. Installation of the Model One/10

9. Display Control....................9-1

9.1 Screen Origin....................9-1
9.2 Display Scaling....................9-1
9.3 Flooding the Screen....................9-1

10. Interactive Device Support....................10-1

10.1 Digitizing Tablet....................10-1

11. Unpacking and Cabling the Ports....................11-1

11.1 Unpacking....................11-1
11.2 Environmental and Technical Specifications....................11-2
11.3 Overview of the Model One/10's Ports....................11-2
11.4 Connecting the Monitor, Keyboard, and Tablet....................11-3
11.5 Printer Interface....................11-5
11.5.1 Using the Printer....................11-5
11.5.2 Printing Text....................11-5
11.6 Using the DIAG Command....................11-5

12. Installing PROMs....................12-1

12.1 Variations in Model One/10s....................12-1
12.2 Warnings....................12-1
12.3 Basic Tasks....................12-1
12.3.1 Removing the Board....................12-2
12.3.2 Replacing the PROMs....................12-4
12.3.3 Inserting the Board and Reconnecting the Power......12-6

Table of Contents
(Continued)

Section Page

13. Selecting Terminal Features Using Setup Mode....................13-1

13.1 Entering Setup Mode......................................13-1
13.2 Viewing the Status of Setup Features.....................13-1
13.3 Selecting Setup Features.................................13-3
13.4 Exiting Setup Mode.......................................13-4
13.5 Setup A Features Defined.................................13-5
12.5.1 Tabs...13-6
12.5.2 Screen Brightness....................................13-6

13.6 Setup B Features Defined.................................13-6
13.6.1 Host Baud..13-7
13.6.2 IEEE Mode..13-7
13.6.3 Host Parity Type.....................................13-7
13.6.4 Host Parity..13-7
13.6.5 Host Data Bits.......................................13-7
13.6.6 Overstrike...13-9
13.6.7 Tablet Type..13-9
13.6.8 Host Graphics Mode...................................13-9
13.6.9 XON/XOFF...13-9
13.6.10 Newline...13-10
13.6.11 Autowrap..13-10
13.6.12 Cursor Type...13-10
13.6.13 Reverse Screen......................................13-10
13.6.14 Control Codes Displayed.............................13-11
13.6.15 Display Format......................................13-11
13.6.16 Solid Background....................................13-11
13.6.17 Text Blinking.......................................13-11
13.6.18 Background Color....................................13-12
13.6.19 Blinking Text Color.................................13-12
13.6.20 Normal Text Color...................................13-13

13.7 Changing Model One/10 Special Characters..................13-13

Table of Contents
(Continued)

Section Page

14. Using Keyboard Controls and LED Indicators....................14-1

14.1 Keyboard Controls..14-1
14.1.1 SET-UP.......................................14-1
14.1.2 ESC (Escape).................................14-3
14.1.3 CTRL (Control)...............................14-3
14.1.4 SHIFT..14-3
14.1.5 CAPS LOCK....................................14-3
14.1.6 NO SCRL (No Scroll)..........................14-3

14.1.7 TAB..14-3
14.1.8 BACK SPACE...................................14-4
14.1.9 RETURN.......................................14-4
14.1.10 LINE FEED...................................14-4
14.1.11 BREAK.......................................14-4
14.1.12 DELETE......................................14-4
14.1.13 Space Bar...................................14-4
14.1.14 Cursor Movement Keys........................14-4
14.1.15 PF1, PF2, PF3, and PF4......................14-5
14.1.16 Other Keypad Keys...........................14-6
14.1.17 Keys With Special Functions in Setup Mode...14-7

14.2 LED Indicators...14-7
14.2.1 ON LINE......................................14-8
14.2.2 LOCAL..14-8
14.2.3 KBD LOCKED (Keyboard Locked).................14-8
14.2.4 L1, L2, L3, and L4...........................14-8

14.3 Function Key Programming................................14-8

15. Using Control Codes and Escape Sequences......................15-1

15.1 Control Codes..15-1
15.2 Escape Sequences...15-3
15.2.1 Cursor Movement Escape Sequences.............15-4
15.2.1.1 Index (<esc> D)........................15-5
15.2.1.2 Reverse Index (<esc> M)................15-5
15.2.1.3 Next Line (<esc> E)....................15-5
15.2.1.4 Conditional Cursor Movement Sequences....15-5

Table of Contents
(Continued)

Section Page

15.2.1.4.1 Cursor Up (<esc> [n A)......15-6
15.2.1.4.2 Cursor Down (<esc> [n B)....15-6
15.2.1.4.3 Cursor to the Right (<esc> [n C)................15-6
15.2.1.4.4 Cursor to the Left (<esc> [n D)................15-6
15.2.1.4.5 Cursor Address (<esc> [row;col f or (<esc> [row;col H)..........15-6

15.2.2 Tab Escape Sequences.................................15-7
15.2.2.1 Set Tab at Current Column (<esc> H)......15-7
15.2.2.2 Clear Tab at Current Column (<esc> [g)..15-7
15.2.2.3 Clear All Tabs (<esc> [3 g).............15-7

15.2.3 COLDSTART Escape Sequence (<esc> c)................15-7

15.2.4 Define Scrolling Region Escape Sequence (<esc> [row1;row2 r)..............................15-7

15.2.5 Character-Erase Escape Sequences..................15-8
15.2.5.1 Erase to End of Line (<ESC> [K).........15-8
15.2.5.2 Erase to Beginning of Line (<ESC> [1 K).15-8
15.2.5.3 Erase Line (<ESC> [2 K).................15-8
15.2.5.4 Erase to End of Screen (<ESC> [J).......15-8
15.2.5.5 Erase to Beginning of Screen (<ESC> 1 J).15-8
15.2.5.6 Erase Screen (<ESC> 2 J).................15-8

Table of Contents
(Continued)

Section Page

15.2.6 Video Attribute Escape Sequences....................15-9
15.2.6.1 Bold On (<ESC> [1 m).....................15-9
15.2.6.2 Underline On (<ESC> [2 m)................15-9
15.2.6.3 Blink On (<ESC> [3 m)....................15-9
15.2.6.4 Reverse Video On (<ESC> [4 m)............15-9
15.2.6.5 Video Attributes Off (<ESC> [m)..........15-9

15.2.7 Set/Clear LEDs Escape Sequence
(<ESC> [parameter q)..................................15-9

15.2.8 Report Sequences....................................15-10
15.2.8.1 What are you? (<ESC> [c
or <ESC> [Z)........................15-10

15.2.8.2 What is the terminal's status?
(<ESC> [5 n)...........................15-10
15.2.8.3 What is the alphanumeric cursor position?
(<ESC> [6 n)...........................15-11
15.2.8.4 What are the terminal parameters?
(<ESC> [0 x or <ESC> 1 x)......15-11

15.2.9 Mode Escape Sequences..............................15-12
15.2.9.1 Set New Line Mode (<ESC> [20 h.........15-13
15.2.9.2 Reset New Line Mode (<ESC> [20 l).....15-13
15.2.9.3 Set Local Echo Mode (<ESC> [12 h)......15-14
15.2.9.4 Reset Local Echo Mode (<ESC> [12 l)....15-14
15.2.9.5 Set Cursor Key Mode (<ESC> [? 1 h).....15-14
15.2.9.6 Reset Cursor Key Mode (<ESC> [? 1 l)...15-14
15.2.9.7 Set Reverse Screen (<ESC> [? 5 h)......15-14
15.2.9.8 Reset Reverse Screen (<ESC> [? 5 l)....15-14
15.2.9.9 Set Origin Mode (<ESC> [? 6 h).........15-15
15.2.9.10 Reset Origin Mode (<ESC> [? 6 l).......15-15
15.2.9.11 Set Autowrap Mode (<ESC> [? 7 h).......15-15
15.2.9.12 Reset Autowrap Mode (<ESC> [? 7 l).....15-15

Table of Contents
(Continued)

Section Page

15.2.10 Color Escape Sequences....................................15-15
15.2.10.1 Select Normal Text Color (<ESC> [? 0 r;g;b m)..................15-16
15.2.10.2 Select Blinking Text Color (<ESC> [? 2 r;g;b m)..................15-16
15.2.10.3 Select Background Color (<ESC> [? 3 r;g;b m)..................15-16

15.2.11 Display-screen Formats....................................15-17
15.2.11.1 Select Screen Format (<ESC> [? fmt ~).......................15-17

15.2.12 Blinking and Normal Text..................................15-17
15.2.12.1 Set Text Blinking Mode (<ESC> [? 10 h).......................15-17
15.2.12.2 Reset Text Blinking Mode (<ESC> [? 10 l).......................15-18

15.2.13 Solid and Graphics Backgrounds...................15-18
15.2.13.1 Set Solid Background (<ESC> [? 11 h).......................15-18
15.2.13.2 Reset Solid Background (<ESC> [? 11 l).......................15-18

15.2.14 Overstrike Mode..................................15-18
15.2.14.1 Set Overstrike Mode (<ESC> [? 31 h....15-18
15.2.14.2 Reset Overstrike Mode (<ESC> [? 31 l..15-18

Appendix A.
Model One/10 Command Set....................................A-1
A.1 New Commands....................................A-2
A.2 Non-applicable Commands....................................A-3
A.3 Modified Commands....................................A-4

Table of Contents
(Continued)

Section Page

Appendix B.
Programming Example: Model One/10 Function Key Definition.......B-1
B.1 Example of Model One/10 Function Key Programming...........B-2

Index...I-1

List of Figures

Figure Page

2.1 Model One/10 Integrated Graphics Terminal.......................2-1
2.2 Data Paths and Functional Blocks................................2-6
3.1 Corners of Image Memory and Graphics Display....................3-3
3.2a Default Clipping Window Location in Image Memory................3-7
3.2b Polygon in Full-screen Clipping Window..........................3-8
3.2c Polygon in Smaller Clipping Window..............................3-9
3.3 Polygon Fill..3-10
3.4 Area Fill...3-11
6.1 Central Processor Addressing and I/O............................6-1
8.1 Image Memory Output Diagram.....................................8-2
11.1 Model One/10 Back Panel...11-4
13.1 Setup A Display...13-2
13.2 Initial Setup B Display..13-3
13.3 Model One/10 Byte Configurations................................13-8
13.4 Setup Mode B Display: Background Color Defaults.................13-12
13.5 Setup Mode B Display: Blinking Text Color Defaults..............13-13
13.6 Setup Mode B Display: Normal Text Color Defaults................13-13
14.1 The Model One/10 Keyboard.......................................14-2

List of Tables

Table Page

2.1 Model One/10 Solutions to Software Development Problems.........2-4
2.2 Sample Command Stream Translator Output.........................2-5
3.1 Corners of Image Memory and Graphics Display....................3-3
3.2 Coordinate Register Assignments..................................3-5
3.3 Value Register Assignments.......................................3-8
3.4 Function Key/Button Mapping......................................3-9
4.1 XFORM2D: Eight Relative Rotations...............................4-21
4.2 XFORM2D: Three Absolute Scalings...............................4-22
4.3 Display List Firmware Commands.................................4-27
14.1 Codes Transmitted When You Press the Cursor Movement Keys......14-5
14.2 Codes Transmitted (With Alternate Keypad Mode Off) When You Press the Function Keys...............................14-5
14.3 Codes Transmitted (With Alternate Keypad Mode Enabled) When You Press the Keypad Keys.................................14-7
15.1 ASCII Code Chart...15-2
15.2 Control Code Summary..15-3
15.3 LED Parameters..15-10
15.4 Reporting Terminal Parameters.................................15-12
15.5 Mode Escape Sequences and Their Parameters...................15-13
15.5 Text Color Escape Sequences and Their Parameters..............15-16
15.7 Available Screen Formats......................................15-17

Part I

INTRODUCTION TO THE MODEL ONE/10

1. Using This Manual

The purpose of this manual is to describe Raster Technologies' Model One/10 integrated graphics terminal. It provides an overview of the Model One/10 system architecture and a summary of the features and specifications of the display system and terminal.

Part II is an installation guide designed to help users with unpacking, setting up, and using special programming of the Model One/10 terminal.

You can use this manual to gain an understanding of the Model One/10's features, both in relation to the rest of the Model One family and as a unique integrated graphics terminal. In addition, if you are an applications programmer, this manual's overview of the system's firmware commands, along with the Model One/10 Command Reference, will get you started with applications development.

Part I includes ten sections. Section 1 briefly discusses this manual's purpose. Section 2 provides an overview of the Model One/10. Section 3 explains the Model One/10's standard firmware command set. Section 4 describes the Display List firmware in some detail. Section 5 discusses the host FORTRAN subroutine library.

Section 6 describes the central processor. This section talks about processor memory, input/output ports, control registers, and status registers. Section 7 explains the vector generator, touching upon pattern registers. Section 8 details the Model One/10's image memory. These topics are included: write-enable masks, read masks, memory bandwidth, and image memory output.

Section 9 covers display control and Section 10 discusses interactive device support.

Part II includes Sections 11 through 15. These sections provide detailed installation assistance, along with guidance for setting up the Model One/10 and terminal programming. Section 11 talks about unpacking the Model One/10 and cabling its ports, and using diagnostics. Section 12 describes a procedure for installing PROMs (for firmware updates only; firmware is pre-installed in new systems). Section 13 details the Model One/10's Setup mode. This section describes the terminal's numerous features and how to select them. Section 14 lists the Model One/10's keyboard controls and their functions. Section 15 explains terminal programming using control codes and escape sequences.

Last, Appendix A lists and describes new commands unique to the Model One/10, and Appendix B provides an example of a program for the Model One/10.

2. Overview of the Model One/10

The Model One/10 is an integrated graphics terminal -- display monitor, keyboard, and controller. This terminal provides the local intelligence to off-load interactive tasks from a host computer system. The Model One/10 is a medium-resolution, high-performance, graphics processing system with the following configuration:

o 640 x 480 x 10 pixels (1024 simultaneous colors).

As for peripherals, the Model One/10 can be configured with an optional mouse or data tablet.

The Model One/10 fully supports general purpose graphics, interactive computer-aided design (CAD), business graphics, and many other applications. Figure 2.1 shows the Model One/10 integrated graphics terminal.

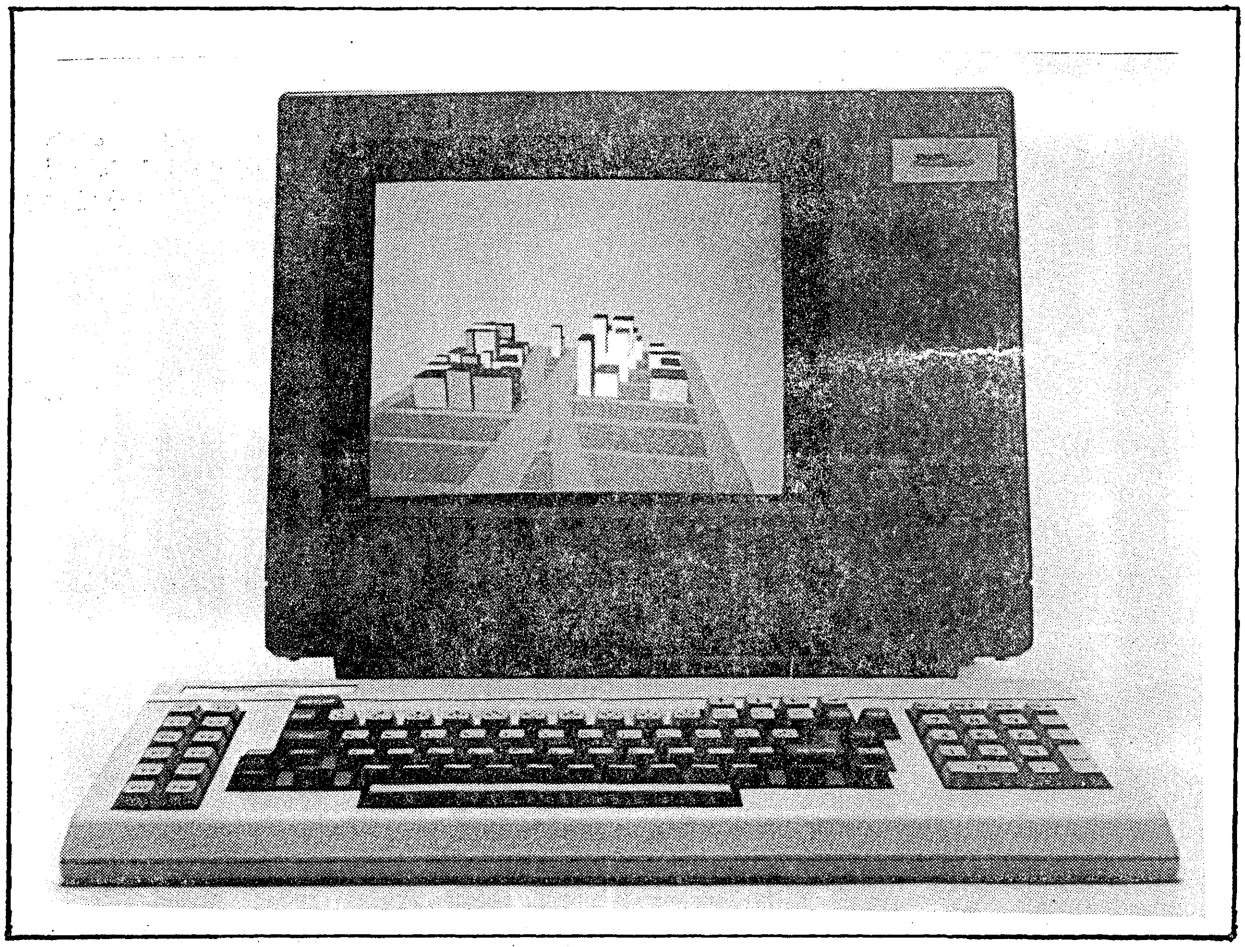


Figure 2.1 Model One/10 Integrated Graphics Terminal

The Model One/10's red-green-blue (RGB) outputs drive its own 640 x 480, 60 Hz, non-interlaced RS-343 monitor. The system includes three 10-bits-in, 8-bits-out programmable Look-up tables (LUTs). These LUTs allow 1024 simultaneous colors out of a total palette of 16.7 million displayable colors.

In operation, the host computer sends commands to the Model One/10 to update the displayed image and to communicate with the local interactive devices. The Model One/10 has a standard RS-232C host interface which operates at speeds ranging from 75 baud to 19.2 K baud.

Multiple processors within the Model One/10 rapidly interpret and execute the host command stream. The 16-bit Z8001 microprocessor and the custom VLSI vector generator provide extremely fast execution of graphics commands.

2.1 Firmware

The Model One/10 offers an integrated set of over 150 firmware graphics commands. The command set includes:

- o Graphics primitives commands

  These commands create and manipulate lines, polygons, circles, arcs, rectangles, horizontal and vertical text, and points. The Model One/10 supports several addressing modes when drawing primitives, including relative and absolute modes.

- o Area fill commands

  These commands fill arbitrary areas with either patterned or solid colors.

- o Display control commands

  This group of commands is used for panning, read and write-protect masks, clipping windows, and crosshairs.

- o Look-up table commands

  These commands program the video look-up tables.

- o Data readback commands

  This set of commands is used for readback of image memory windows, pixel values, current point, and other values.

- Image transmission commands

  These commands allow you to select from several data formats for the transmission of pre-computed pixel-by-pixel raster images from the host to the Model One/10.

- Interactive device support commands

  These commands provide interactive device support, such as blinking and button table functions.

- Macro programming commands

  These commands are used to define, execute, and delete macros.

- Register operation commands

  These commands are for control of the coordinate and value registers.

- Software development commands

  These commands aid in software development. Included in this set are a command stream translator and a local debugger.

- VT-100/Alphanumeric emulation

  These commands allow the monitor to be used as an alphanumeric terminal with attached keyboard. You can specify multiple independent scrolling windows. They can be any size and placed wherever desired.

- Display List Firmware commands

  These commands locally handle scaling, rotation and translation with quite a few transformation options, all user-definable. Picking and highlighting of selected geometry is also available with many variations.

Model One/10 firmware is discussed in Section 3. Appendix A describes the command set differences between the Model One/10 and the rest of the Model One family. In addition to the standard firmware, an optional library of FORTRAN-callable subroutines is available with the Model One/10 for host-level access to all firmware commands.

2.2 Applications Development Features

The Model One/10 includes features for the easy development of applications programs. These development tools are designed specifically to minimize typical problems encountered in applications development and support, as shown in the following table.

Table 2.1 Model One/10 Solutions to Software Development Problems

Software Development Problems	Model One/10 Solutions
Off-loading graphics tasks from the host program: clipping, scrolling windows, cursor tracking.	Firmware commands allow the Model One/10 to perform operations previously requiring host intervention.
Training new graphics programmers for the Model One/10.	Programmers use English-like command mnemonics for local programming.
	Local mode allows programmers to test commands locally, without having to write host programs.
Testing new graphics concepts and sequences without writing or rewriting host programs.	Programmers use local mode.
Cost of programming microprocessor intelligence for graphics use.	Programmers use macro command sequences and alphanumeric windows.
Applications menu management and ongoing support.	Programmers use macro command sequences.
Break-pointing new applications programs during development: stepping through programs under local control.	Complete local debugger allows breakpointing, stepping through programs, local command execution and listing of defined macros.
Local management of cursors, buttons and other interactive devices.	Programmers use macro command sequences and interactive device support.

The Model One/10 supports macro programming. It can store up to 256 macros at a time. The macros you define can be easily executed with the Model One/10's button table and its keyboard's function keys (see Section 3.11). Thus, you can execute a macro by pressing a single key. Macros can include text strings to be sent to the host computer, so that a single function key can issue any frequently-used host command, as well as perform any local graphics function.

The One/10's local debugger allows you to step through program execution, list defined macros, execute graphics commands locally, and return to normal execution when desired. The command stream translator is especially useful when combined with the local debugger. When you invoke the command stream translator, it translates the command stream from the host into easy-to-read mnemonics (the same mnemonics used when entering commands locally) and prints these mnemonics and their associated parameters at the local terminal. Once application development is complete, you can disable the command stream translator and local debugger while they remain available for diagnostic use. Table 2.2 shows sample output from the Model One/10's command stream translator.

Note: Executing the INSPID, DELPID, or RDPID commands while using the command stream translator will cause the command stream translator to be exited.

Table 2.2 Sample Command Stream Translator Output

Command Stream From Host	Local ALPHA Output
...0100040006..	MOVABS 0004 0006
...0600FFFF..	VALUE 00 255 255
...0E002D..	CIRCLE 100
...0200E300D4..	MOVREL 100 105
...9005C8E5ECECEF..	TEXT1 Hello

2.3 System Architecture

The hardware architecture of the Model One/10 includes these functional blocks:

- o Central processor
- o Hardware vector generator
- o Image memory
- o Look-up tables
- o Video output

Figure 2.2 shows the data paths and relationships between these functional blocks.

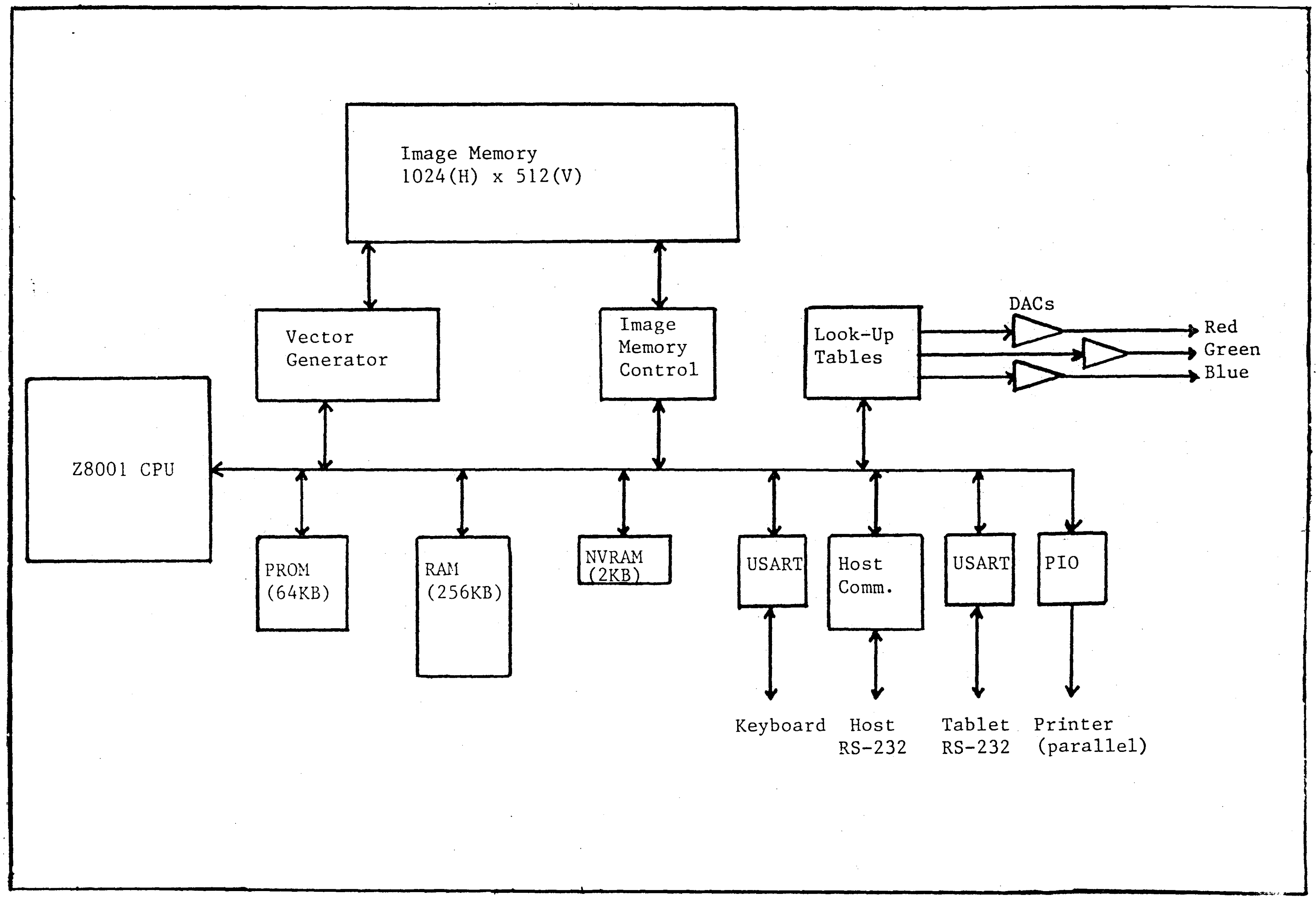


Figure 2.2 Data Paths and Functional Blocks

The Model One/10 central processor is a general-purpose 16-bit microcomputer whose principal function is to interpret and decode graphics commands received by the Model One/10 and generate appropriate commands to the hardware vector generator and other system units.

The hardware vector generator is a custom VLSI processor which executes a digital-line-generating algorithm. The central processor processes incoming commands while the vector generator simultaneously makes real-time updates to image memory. The vector generator also performs the display list picking function (see Section 4).

3. Model One/10 Firmware

The Model One/10 firmware, a set of over 150 commands, is the control program run by the Model One/10's central processor.

This firmware has two major functions:

- o Controlling communications, and
- o Interpreting and processing graphics commands.

The hardware supports serial I/O with both the host computer and with optional graphics input devices.

3.1 Communications

The Model One/10 provides one firmware-configurable RS-232C serial port for the host. In Model One/10 Setup mode or with the SYSCFG command, you can configure the host port for parity, baud rate (from 75 baud to 19.2K baud), XON/XOFF protocol, and others. All input and output through this port is buffered and interrupt-driven. The keyboard, although locally connected at a fixed baud rate, is viewed as a separate port by the system firmware.

The tablet/mouse port has its baud rate fixed at 9600 baud and is set for: eight data bits, no parity, and one stop bit.

3.2 Alpha and Graphics Modes

The mode of the ports determines whether characters received at those ports are interpreted as alphanumeric data or a graphics commands. The host and keyboard ports are always in either Graphics mode or Alpha mode. Only one of the Model One/10's ports can be in Graphics mode at a time.

Initally, the host and keyboard ports are in Alpha mode by default. To enter local Graphics mode, type CTRL-D, the default special control character for the ENTERGRAPHICS command. If you want to change this control character, use the SPCHAR command to select an alternate character.

Once you enter Graphics mode, all subsequent data is interpreted by the graphics command interpreter until you exit this mode with the QUIT command.

Binary graphics commands are composed of a one-byte opcode followed by a string of bytes which are operands. The opcode specifies the graphics command, for example, 0E (hexadcimal) specifies the CIRCLE command. You supply the data needed for command execution by specifying the required operands, the radius of a circle, for instance. The local Graphics mode accepts ASCII mnemonic names and ASCII decimal or hexadecimal parameter values for all Model One/10 commands.

3.3 Host Serial Communications Command Format

Commands sent over the host serial interface are normally sent in 8-bit binary format. In binary format, the opcode for each command takes one byte and is sent in the first character, with each operand byte sent in subsequent 8-bit characters.

For those host computers which cannot be programmed to transmit 8-bit binary characters over their serial lines, the Model One/10's host port can be configured to accept ASCII-hexadecimal format. In this format, each 8-bit binary character is replaced by two hexadecimal digits, sent as two successive ASCII characters.

Note that the 8-bit binary format is twice as efficient as the ASCII-hexadecimal format and should be used by any host computer capable of transmitting 8-bit binary characters. Data from the host can also be sent in Alpha mode (same as local format) after you first execute the ASCII command.

3.4 Local Command Format

Local input in alphanumeric form allows you to test a sequence of commands without writing and compiling a host computer program to generate the commands. Use the Model One/10's keyboard to enter local commands.

When you type the ENTERGRAPHICS special character (default is CTRL-D; the user can select another ENTERGRAPHICS special character by using the SPCHAR command), the Model One/10 responds with a graphics command prompt "!" on its screen. You then talk directly to the Model One/10's command interpreter until you type QUIT.

The Model One/10 accepts graphics commands from its keyboard in an English-like mnemonic form, rather than in binary or hexadecimal form. You use commands, for instance, such as CIRCLE, POINT, MOVREL, and TEXT. As for operands, you can enter them either in hexadecimal (e.g., #1FF) or decimal (e.g., 511).

3.5 Diagnostic and Development Features

Special debugging features are included with the Model One/10 to aid you in developing and debugging applications. A local debugger and a command stream translator are available. Also, error messages can be sent either to the host or to the Model One/10's display. In addition, the REPLAY command replays the last 32 characters that were sent over the host interface.

You can enter the local debugger by typing CTRL-X. You can then step through the commands being sent from the host, list all defined macros, list the contents of a specific macro, enable execution of that macro every video frame and exit the local debugger.

The command stream translator lets you set the Model One/10 to disassemble automatically the command stream from the host. As the command stream is received, therefore, it is translated into readable command mnemonics and parameters. The command stream translator allows you to see the actual commands as they are executed.

3.6 Coordinate System

The default coordinate system for all graphics commands has (0,0) in the center of the displayed image, with the positive x-axis to the right, and the positive y-axis above. The locations of the four corners of both image memory and the graphics display are given in Table 3.1.

Table 3.1 Corners of Image Memory and Graphics Display

	Lower-left	Upper-left	Lower-right	Upper-right
Image Memory	-320, -240	-320, 271	703, -240	703, 271
Display	-320, -240	-320, 239	319, -240	319, 239

Figure 3.1 illustrates the corners of image memory and the Model One/10 display. The figure shows that the display is a subset of total image memory.

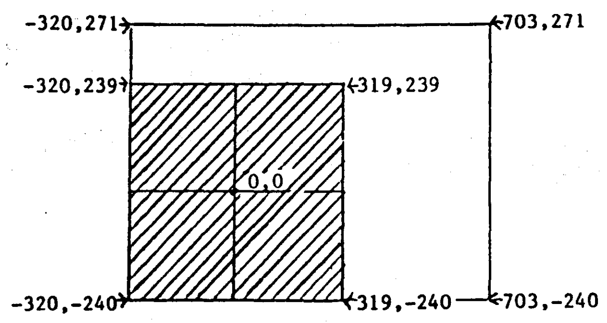


Figure 3.1 Corners of Image Memory and Graphics Display

You can use the coordinate origin register to change the default coordinate system to any desired orientation. To change the coordinate origin register, use the CORORG command. Coordinates are supplied as 16-bit integers ranging from -32,768 to 32,767. Relative moves and draws which would take a coordinate out of this range are clipped to these values.

Caution: The use of the CORORG command is not recommended because all coordinate registers are modified by the CORORG command, and you can obtain any desired translation with the XFORM2D command. If you do use the CORORG command, issue it only immediately following a COLDstart (since all coordinate registers are affected).

3.7 Coordinate Registers

One of the most powerful uses of the coordinate registers (CREGs) (see Table 3.2) is in passing coordinate parameters in macro calls. The Model One/10 has a set of 64 coordinate registers for manipulating coordinate data. Each coordinate register has a 16-bit x-component and a 16-bit y-component. Some of these registers are assigned to specific functions. Others are available to you for temporary coordinate storage.

You can load coordinate registers with the CLOAD command. In addition, you can add (CADD), subtract (CSUB), and move (CMOVE) between pairs of coordinate registers. Here is an example:

```
! CLOAD 25 100 150      ; Load CREG 25 with 100,150.
! CLOAD 26 10 20        ; Load CREG 26 with 10,20.
! CADD 25 26            ; Add contents of CREG 26 to CREG 25 and place
                        ; result in CREG 25.
! READCR 25             ; Read the contents of CREG 25.
 00110 00170            ; The Model One/10 responds that coordinate
                        ; values 110,170 are in CREG 25.
! CADD 25 26            ; Add contents of CREG 26 to CREG 25 and place
                        ; result in CREG 25.
! READCR 25             ; Read contents of CREG 25.
 00120 00190            ; The Model One/10 responds that coordinate
                        ; values 120,190 are in CREG 25.
```

Note: All operations (e.g., CORORG, CLOAD, CMOVE, etc.) affecting the coordinate origin (CREG 3) modify it only relative to its current value.

Table 3.2 Coordinate Register Assignments

Coordinate Register	Use
0	Current point: used as a reference point by graphics commands. The current point is modified by MOVE and DRAW commands.
1	Unscaled XY digitizer location, updated automatically by the Model One/10 with each new packet from the digitizing device.
2	Scaled XY digitizer location, updated automatically by the Model One/10 with each new packet from the digitizing device.
3	Coordinate origin: used to position physical image memory within the virtual address space. The coordinate origin can be modified by the CORORG command.
4	Screen origin: specifies the pixel which appears at the center of the screen. The screen origin is changed by the SCRORG command. CREG 4 is used for horizontal and vertical panning.
5	Crosshair current location: changes to this register move the crosshair. The crosshair is enabled/disabled using the XHAIR command.
6-8	Available for use by applications programmer.
9	Clipping window, lower-left corner.
10	Clipping window, upper-right corner.
11-13	Reserved.
14	Direction of pixel writing for PIXMOV destination windows.
15	Screen origin for overlay planes (bit planes 8, 9).
16-18	Reserved.
19	Center of pick aperture.
20-63	Available for use by applications programmer.

3.8 Clipping Windows

The Model One/10 supports a clipping window (see Figure 3.2a) for the clipping of graphics primitives to any prescribed window in image memory. Clipping windows keep vectors from "spilling over" into neighboring areas of the image.

For example, to clip a filled polygon:

1. Use the PRMFIL command with flag = 1.

2. Draw a polygon with the POLYGN command in a full screen window.

3. Define a smaller clipping window with the WINDOW command. Specify bottom-left and upper-right corners.

4. Draw the same polygon (see Figures 3.2b, and 3.2c).

You can define the clipping window corners either by directly modifying the appropriate coordinate registers, CREG 9 for lower-left corner and CREG 10 for upper-right corner, or with the WINDOW command.

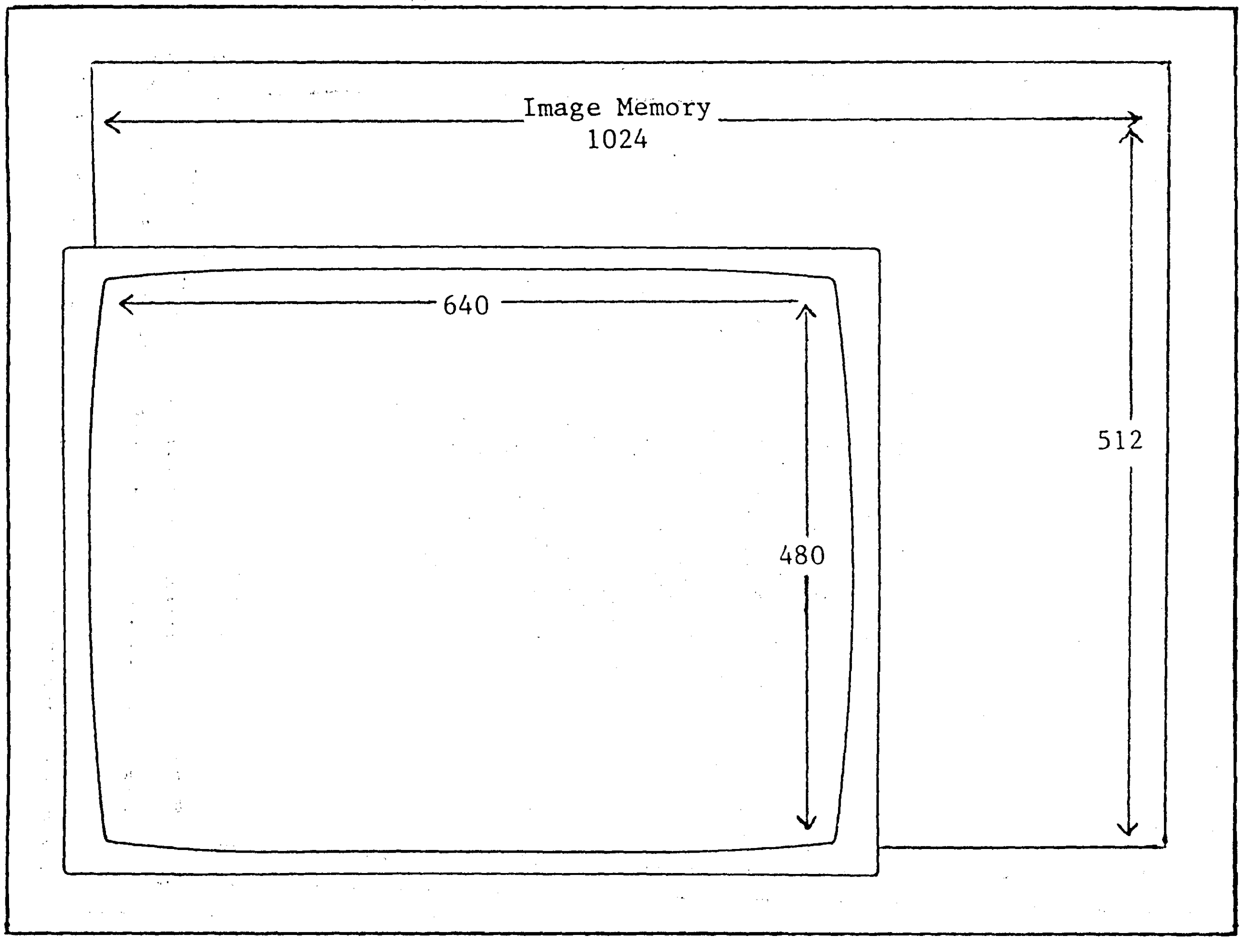


Figure 3.2a Default Clipping Window Location in Image Memory

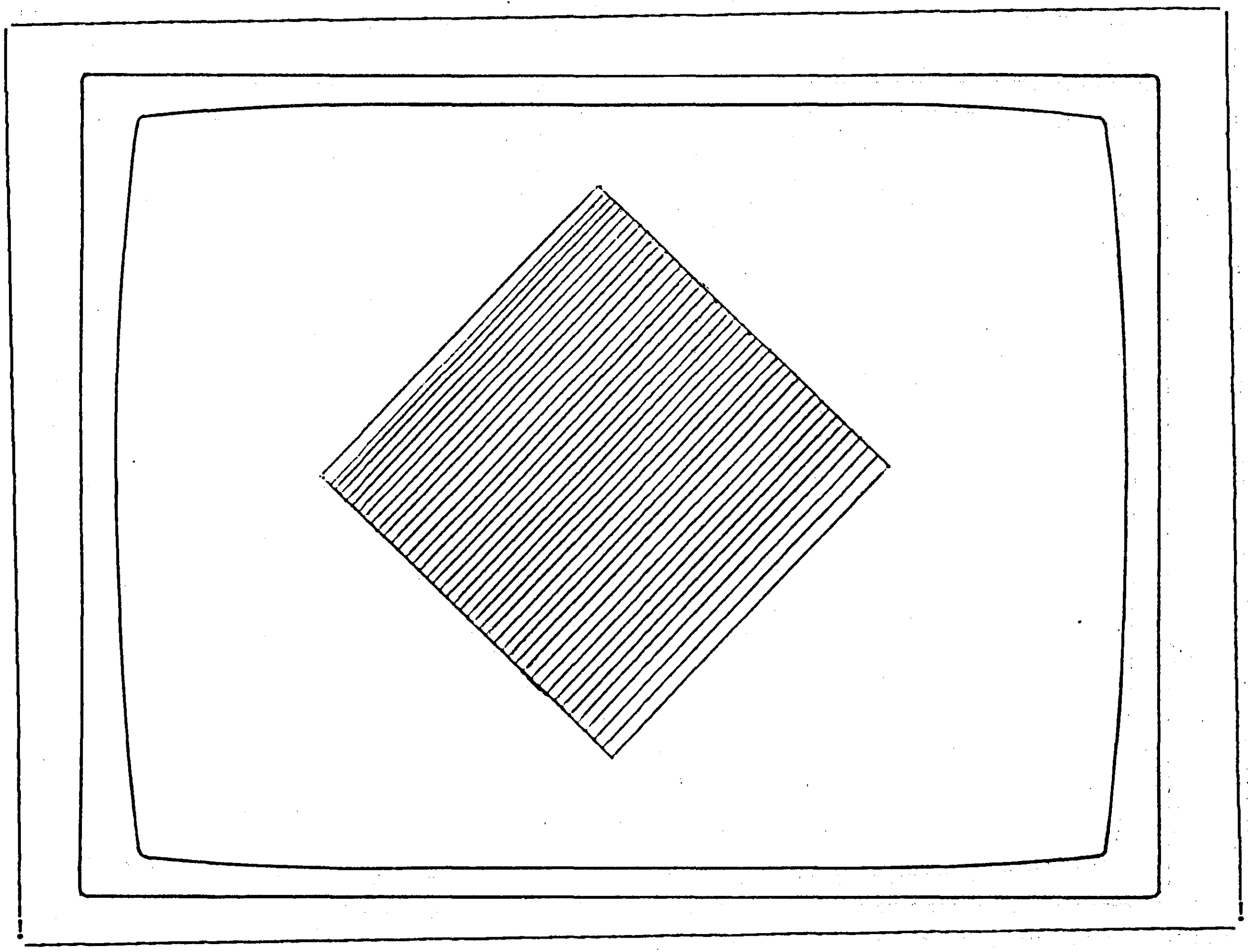


Figure 3.2b Polygon in Full-screen Clipping Window

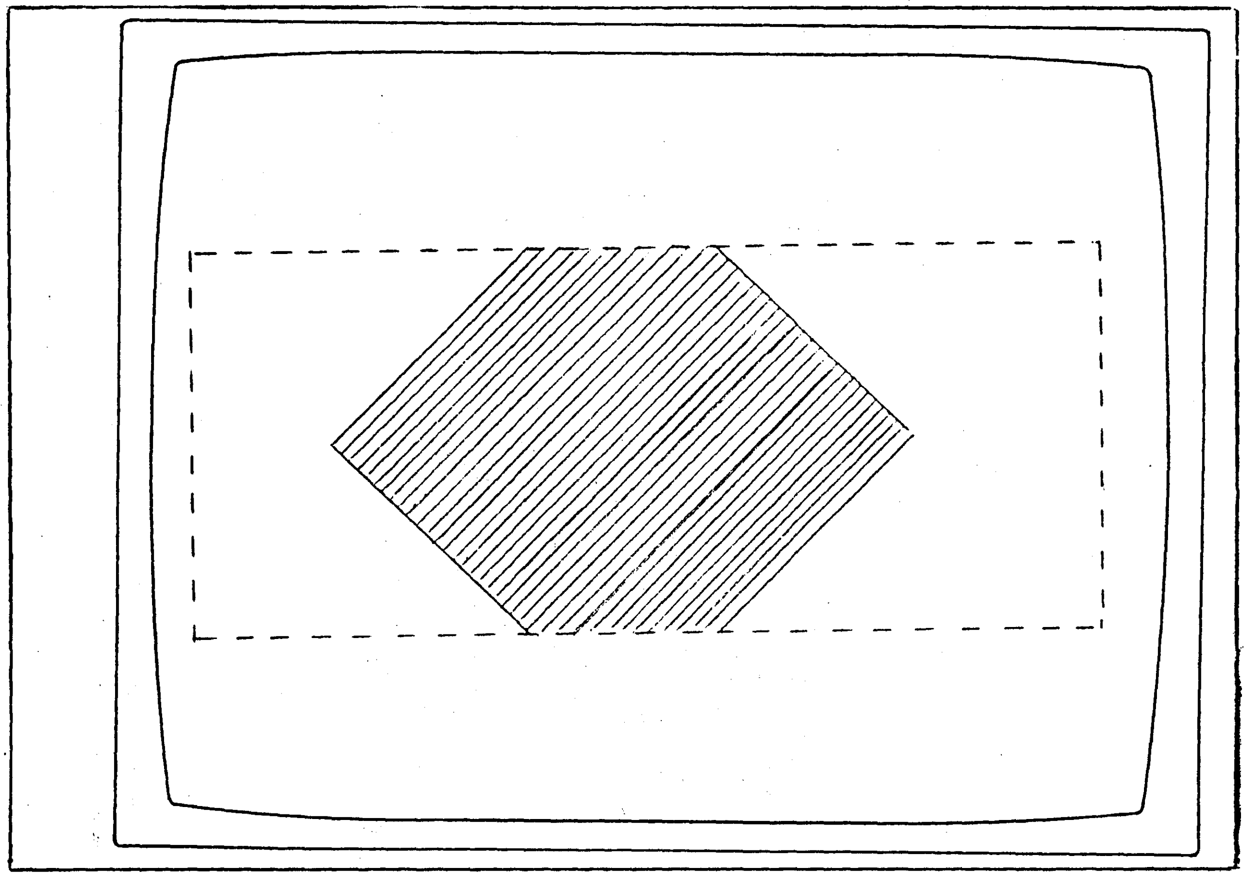


Figure 3.2c Polygon in Smaller Clipping Window

3.9 Graphics Primitives and Area Fills

The Model One/10 supports the filling of graphics primitives and two kinds of area fills. One command, PRMFIL (Primitive Fill), allows you to specify whether subsequent graphics primitives should be drawn filled or unfilled. For example:

```
!PRMFIL ON       Select filled primitives.
!VAL8 48         Set the active color to red.
!MOVABS 50 50    Move current point to 50,50.
!CIRCLE 20       Draw red-filled circle with radius of 20.
!MOVABS 100,100  Move current point to 100,100.
!PRMFIL OFF      Select unfilled primitives.
!CIRCLE 20       Draw circle (unfilled) with radius of 20.
!MOVABS 100 50   Move current point to 100,50.
!RECTAN 110 60   Draw rectangle (unfilled) from the current point to 110,60.
!PRMFIL ON       Select filled primitives.
!MOVABS 50 100   Move current point to 50,100.
!RECTAN 60 110   Draw filled rectangle from the current point to 60,110.
```

The vector pattern register, controlled by the VECPAT command, also determines the filling of graphics primitives (see Figure 3.3).

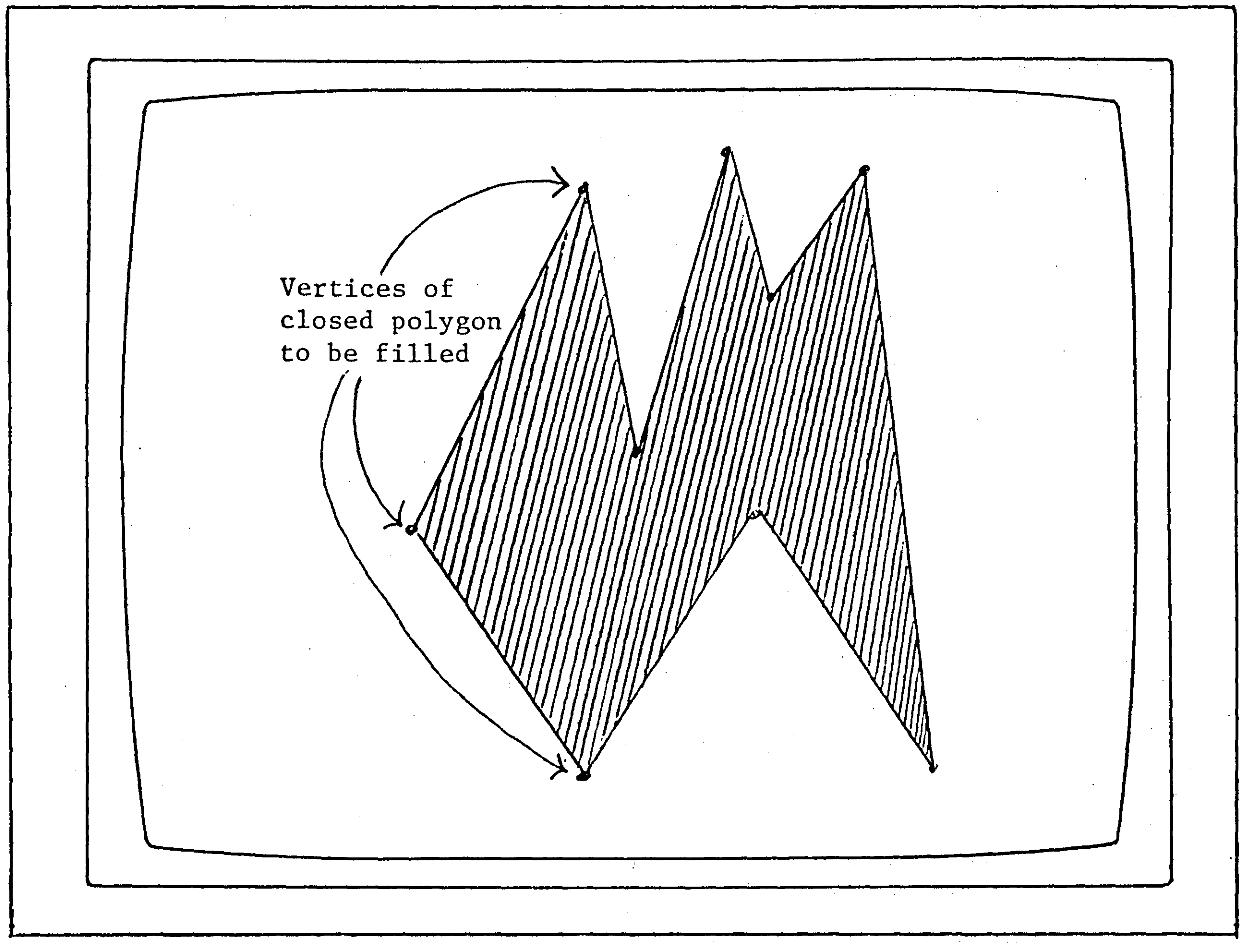


Figure 3.3 Polygon Fill

For an area fill, you must define a seed point and a boundary. Every pixel within the boundary is then set to the specified value (see Figure 3.4).

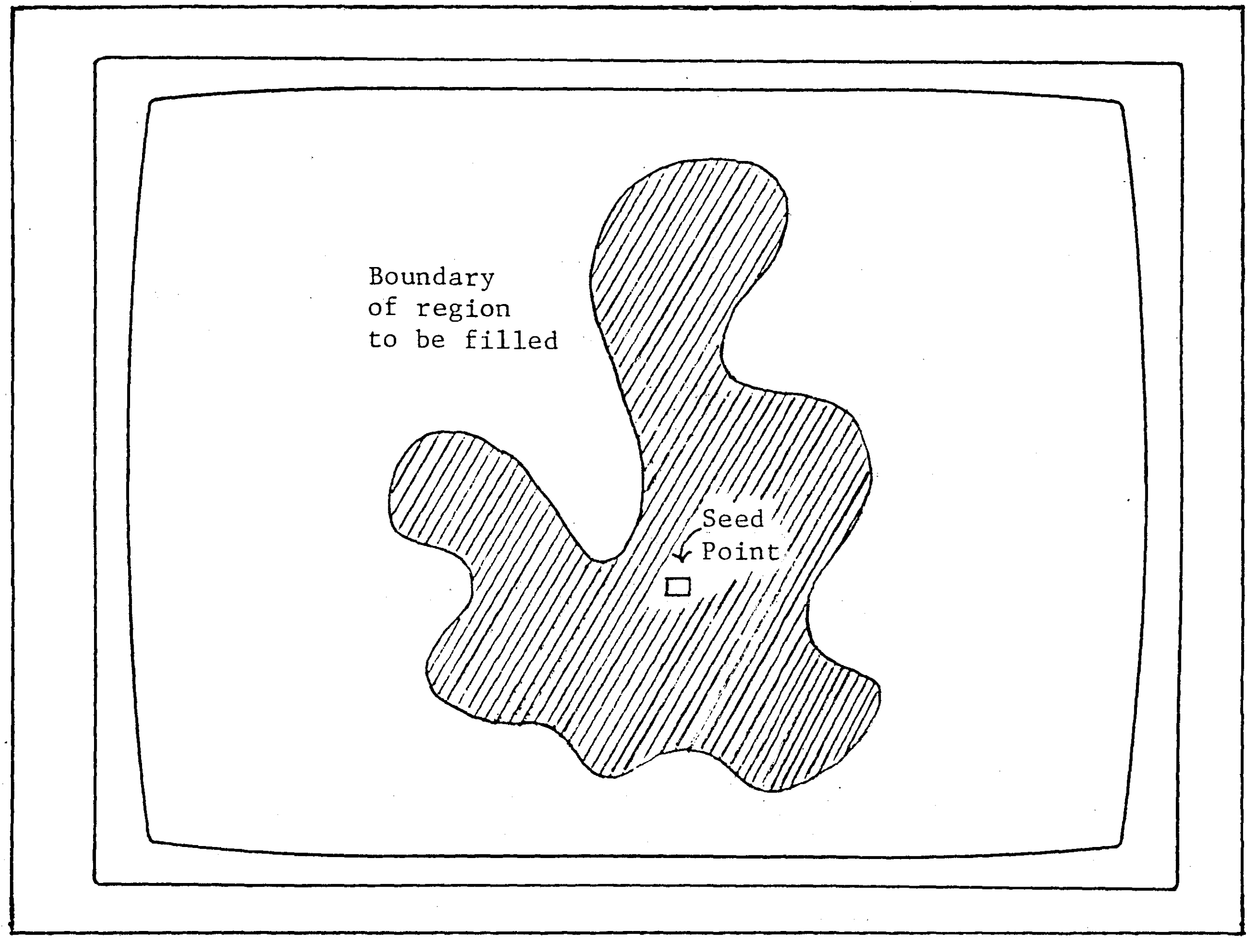


Figure 3.4 Area Fill

Both areas and primitives can be filled with a pattern. The PATFIL command enables and disables patterned area fill. The SETPAT command determines the pattern to be used in area fills.

3.10 Pixel Value Registers

As with the coordinate registers (CREGs), one of the most powerful uses of the pixel value registers (VREGS) is passing value parameters in macro calls.

The Model One/10 has 64 pixel value registers for the manipulation of image data values. Some of these registers are reserved for Model One/10 use; the rest are available for programmer use. To load the value registers, use the VLOAD command. Additionally, you can add (VADD), subtract (VSUB), and move (VMOVE) between pairs of value registers.

Table 3.3 summarizes the use of the value registers.

Table 3.3 Value Register Assignments

Value Register	Use
0	The current pixel value; this is the value used by all commands that write graphics data to the Model One/10.
1	Reserved.
2	Reserved.
3	Fill mask used for seeded area fills.
4	Color assignment for Overlay Plane 0.
5	Color assignment for Overlay Plane 1.
6	Reserved.
7-42	Available for temporary value storage.
43-44	Reserved.
45	Highlight color.
46-50	Reserved.
51-63	Available for temporary value storage.

3.11 Defining and Executing Macro Commands

The Model One/10 has macro programming capability. You can define and store a group of commands and then execute them whenever desired. The Model One/10 can store up to 256 macros simultaneously. To create a macro, use the MACDEF command, specifying a number for that macro, followed by a string of Model One/10 commands. Any Model One/10 command -- including defining and executing other macros -- can be included in a macro with only these exceptions: ASCII, QUIT, SYSCFG, DISCFG, DFTCFG, and SAVCFG. You can nest macros up to sixteen deep. A macro is ended with the command MACEND.

Furthermore, the local debugger allows macros to be sent from the host and then rewritten locally for testing. You do not have to recompile a host program. Parameters can be passed between macros by using the coordinate registers (CREGs) and pixel value registers (VREGs) for storage of coordinates and pixel values.

After defining macros, you can execute them in several ways. One method is to use explicit MACRO commands, sent either from the host, from the keyboard, or from another macro.

3.11.1 Using the Button Table to Execute Macros

An alternate method to execute macros is to use the button table. First, load the button table with the BUTTBL (Button Table) command, linking individual macro execution with specified button numbers. This table can hold 127 entries, numbered from -63 to 63. Each entry, or "button", corresponds to a particular macro. The positive button numbers represent button hits, and negative button numbers represent button releases.

Next, execute a particular macro in one of three ways. One way is to use the BUTTON command, specifying the button number that corresponds to the macro you want to execute. Note that if you first set Location 0 and then you do not specify a button number, the entry at button table Location 0 is executed; this macro executes every 1/60th second and is very useful for functions such as cursor tracking.

A second way is pressing a function key (Fkey). On the left side of the Model One/10 keyboard is a set of 12 function keys. Twelve buttons (16-27) are "mapped" to these Fkeys. Press the Fkey corresponding to the appropriate button entry. See Table 3.4 for function key/button mapping on the Model One/10.

Table 3.4 Function Key/Button Mapping

Fkey	Button	Fkey	Button
F1	16	F2	17
F3	18	F4	19
F5	20	F6	21
F7	22	F8	23
F9	24	F10	25
F11	26	F12	27

There is yet a third way to execute macros with the aid of the button table. If you have a digitizing tablet and multiple-button puck (tablet cursor), you can push or release a button on the puck to execute a macro. The button numbers on the puck correspond to the entry numbers in the button table, except that Button 0 on the puck maps to Entry 63 of the button table.

Macros executed by button pressing or releasing can even modify the button table, allowing interactive flexibility. For example, you can easily set up a tree of menus, since macros can also define other macros and execute them. When you combine it with the available macro command facilities, the button table supports many applications entirely within the Model One/10, thus off-loading the host computer.

3.12 Display List Firmware

Display List Firmware, standard with the Model One/10, offers a compatible two-dimensional display list management capability. Display List commands can perform transformations -- local scaling, rotation and translation of a two-dimensional graphics data base. These operations can be relative to a previous transformation or absolute. Transformation data, in addition to several other states such as flags and variables, can saved by placing them on a stack.

Several commands are available for efficient segment editing. Pickability and visibility of segments are easily controlled. Over five kinds of pick information can be stored and read back for quick editing, highlighting, and hierarchy verification. For more information, see the next section, called Display List Firmware.

4. Overview of the Model One/10 Display List Firmware

The Raster Technologies Display List Firmware is standard with the Model One/10. This firmware is a general purpose development tool, well suited to the following applications: electrical CAD, both VLSI chip design and PCB layout; mechanical CAD; cartography and seismic applications; and presentation graphics applications.

4.1 Display List Structures

The basic display list structure is the segment. Segments are lists of commands. The commands comprising segments can be categorized by the user in several ways. They can be grouped together into nested segments or labeled using pick IDs.

Both whole segments and segment parts have attributes. These attributes allow segments and segment subdivisions to be grouped and regrouped into many simultaneous, perhaps-overlapping categories. This process of easily manipulating geometry into groups in order to visually differentiate these groups is one of the basic purposes of display lists. The other is to facilitate rapid redisplay of images when the communications link would otherwise be a limiting factor.

4.1.1 Segments: Characteristics and Storage

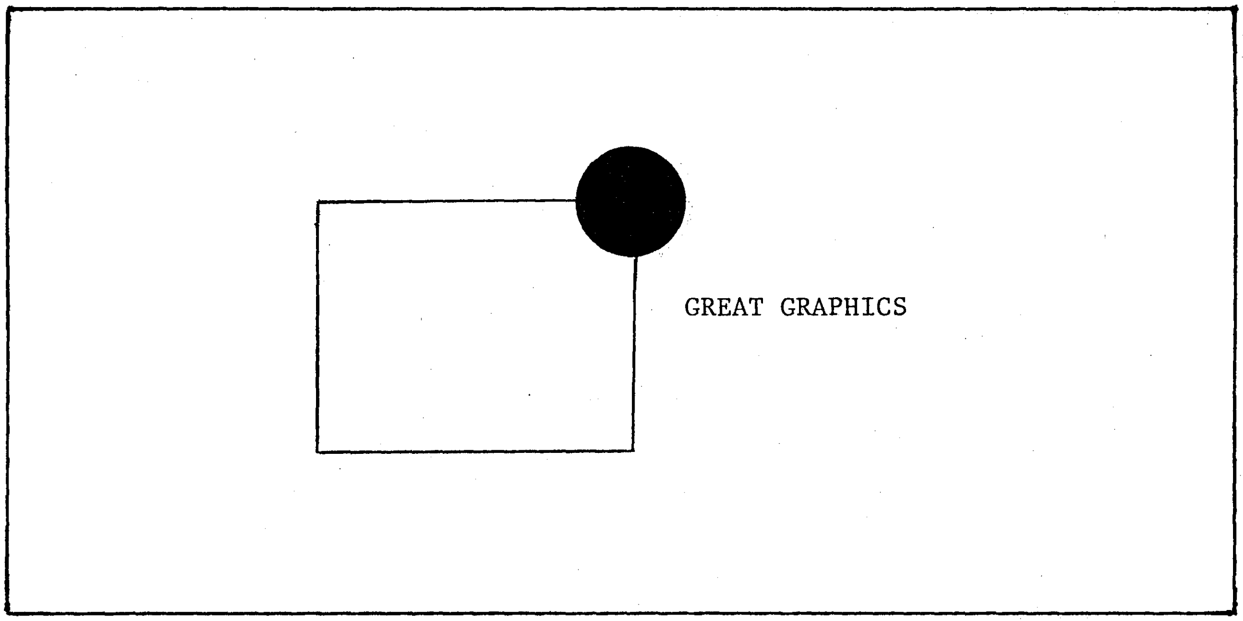
In a display list, graphics primitives are stored and referenced in structures called segments. A segment is a collection of Model One commands, manipulated as a single unit. The following example shows a typical segment definition.

Example 4.1 Typical Segment Definition

```
! SEGINI 0 128        ;Initialize display lists.
! SEGDEF 1            ;Begin definition of Segment 1.
$ VAL8 63             ; Definition begins with setting the current
                        value to white...
$ MOVABS 0 0          ; Move the current point to 0,0...
$ RECTAN 100 80       ; Draw a white rectangle...
$ PRMFIL ON           ; Set PRMFIL to ON; fill subsequent
                        primitives...
$ VAL8 3              ; Set the current value to blue...
$ MOVABS 100 80       ; Move the current point to 100,80...
$ CIRCLE 20           ; Draw a blue circle with radius 20...
$ VAL8 48             ; Set current value to red...
$ MOVABS 120 40       ; Move the current point to 120,40...
$ TEXT1 GREAT GRAPHICS
                      ; Draw text string, "GREAT GRAPHICS"...
$ SEGEND              ;End definition of Segment 1.
!
```

Note that defining Segment 1 does not draw it on your screen. To draw segments (see Section 4.4.1.), execute them with the SEGREF command. When Segment 1 above is drawn, it looks like this:

Example 4.1 (continued)



Each segment is identified by a unique 32-bit integer called the segment ID (Segment 1 in the above example). Segment IDs greater than or equal to FC000000 (hex) are reserved. Both the number of segments and the size of each one are limited only by the amount of display list memory in the system.

On the Model One/10, local lists are stored in the 128K of RAM accessed by the Z8000. To facilitate memory management, the firmware divides segments into smaller, fixed-length chunks, called blocks.

You must specify this block size with the SEGINI command. It is important to specify a size that is large enough to allow optimum performance (minimize memory management) but small enough to avoid wasting RAM. The minimum block size is 32 bytes and the maximum is 16,384 bytes. Note that no segment command can be executed until SEGINI is issued.

If you want to check on system storage, the SYSTAT command returns the Display List Firmware memory usage and availability.

4.1.2 Pick IDs

Segment-definition commands can be labeled (delimited) with pick identification numbers (pick IDs). A pick ID is a 32-bit-integer label used to associate a group of graphics primitives with each other and to separate one group from another group.

The PICKID command is a tag, or delimiter, which defines a new pickable entity. This "entity" consists of all subsequent Model One commands until the next PICKID command. Pickable entities can be deleted as a single unit. Pick IDs also facilitate local picking and highlighting (see Section 4.5), in addition to segment editing.

In the following example, the same segment definition used in Example 4.1 is subdivided by pick ID labels. Each group of commands that sets the color value, moves the current point, and draws primitives becomes one pickable entity.

Example 4.2 Typical Segment Definition Subdivided by Pick ID Labels

```
! SEGINI 0 128      ;Initialize display lists.
! SEGDEF 1          ;Begin definition of Segment 1.
$ PICKID 1          ; Definition begins with assigning pick
                      identification number 1 to the following
                      commands...
$ VAL8 63           ; Set the current value to white...
$ MOVABS 0 0        ; Move the current point to 0,0...
$ RECTAN 100 80     ; Draw a white rectangle...
$ PICKID 2          ; End Pick ID 1 and begin Pick ID 2; assign
                      Pick ID 2 to the following commands...
$ PRMFIL ON         ; Set PRMFIL to ON; fill subsequent
                      primitives...
$ VAL8 3            ; Set the current value to blue...
$ MOVABS 100 80     ; Move the current point to 100,80...
$ CIRCLE 20         ; Draw a blue circle with radius 20...
$ PICKID 3          ; End Pick ID 2 and begin Pick ID 3; assign
                      Pick ID 3 to the following commands...
$ VAL8 48           ; Set current value to red...
$ MOVABS 120 40     ; Move the current point to 120,40...
$ TEXT1 GREAT GRAPHICS
                    ; Draw text string, "GREAT GRAPHICS"...
$ SEGEND            ;End definition of Segment 1.
!
```

When Segment 1 is now drawn, it looks exactly the same as in Example 4.1. But editing this segment is now possible. For example, the graphics labeled PICKID 1 can be deleted as one entity. To do this, use the DELPID command.

See Section 4.2.2 for more information on segment editing and Section 4.5 for a detailed discussion of picking.

4.1.3 Segment Attributes: Visibility and Pickability

When you define a segment, two attributes, visibility and pickability, are initially set to ON. If a segment is visible, it is drawn when it is executed. On the other hand, if a segment is invisible and executed, the commands in its definition are interpreted (for example, DRAW commands move the current point) but image memory is not altered. Likewise, if a segment is pickable, its commands are subject to being picked during a pick operation.

You can change the visibility and pickability attributes of any segment with the SETATR command.

4.1.4 Segment Identity Information

The Model One stores and returns to you when requested several types of information on segment identity: segment ID numbers, pick ID numbers, and nesting levels. To save this information, the system uses registers, counters, and a pick buffer. To return this information, the Display List Firmware has several readback commands.

4.1.4.1 Registers

Picking operations (see Section 4.5) use two local registers: the segment ID register and the pick ID register. SEGREG contains the current segment ID and PIDREG holds the current pick ID.

The purpose of these registers is to determine what to highlight/unhighlight or what to delete with DELPID. Three commands can update these registers: SEGREF in pick mode (picking) if there is a pick hit, RDPICK, and SETGL.

The readback command RDREG returns the segment ID and the pick ID currently in SEGREG and PIDREG respectively. These registers are used primarily for highlighting but also for deleting pick IDs with the DELPID command.

4.1.4.2 Counters and Pick Buffer

The display list counters and the pick buffer are used, again, for picking operations. These two components of the Model One are discussed in detail in Section 4.5.

4.1.4.3 Reading Back Segment Information

The Display List Firmware readback commands are: RDPICK, RDREG, RDXFORM, SEGINQ, and SYSTAT. For all these commands, the format of the readback (binary or ASCII) is determined by the value you previously specified with the RDMODE command. You should use RDMODE 0 (ASCII), which is the default. RDMODE 1 is intended for DMA interfaces; the Model One/10 uses a serial interface.

4.2 Creating and Modifying Segments

After segments have been defined, they can be modified by adding commands to them, deleting commands from them, copying them, renaming them, and deleting them entirely.

4.2.1 Defining Segments

The SEGINI command initializes all display list structures and should, therefore, be used whenever an entirely new set of segments is to be defined. Only after you specify the block size and initialize display list structures are you ready to define segments and assign segment attributes.

The SEGDEF command begins the definition of a segment and sets the visibility and pickabiity attributes to ON for that segment. A segment definition can include a variety of commands, including

- Commands that draw 2-D graphics primitives.
- Commands that move the current point.
- Commands that change the current color.
- Commands that change primitive-generation attributes, such as VECPAT and PRMFIL.
- The SEGREF command to nest (reference) child segments.
- The PICKID command.

Note: Do not execute a BREAK during a segment definition. If you execute a BREAK during a segment definition, the system will hang when that segment is later referenced from Graphics mode.

You must end every segment definition with the SEGEND command to close the segment. SEGEND also exits the display controller from the sublevel "segment definition" mode, returning it to regular graphics mode.

Several readback commands give information on defined segments. A list of defined segments can be returned with execution of the SYSTAT command. In addition, the existence of a segment can be determined with the SEGINQ command.

4.2.2 Modifying Segments

Existing segment definitions can be modified in several ways. If you wish, you can add new commands. The SEGAPP command reopens a closed segment so that commands can be appended to it. You must use a SEGEND command to terminate each append.

Replacing portions of a segment definition can be done with a combination of deleting and appending. First, use the DELPID command, previously mentioned in Section 4.1.2, to delete the commands assigned a particular pick ID label. Second, use the SEGAPP command.

The INSPID command allows you to open a segment for insertion of graphics primitives. Any command which is legal in segment definition mode can be inserted until a SEGEND command is encountered.

You can also delete entire segments with the SEGDEL command, rename segments with the SEGREN command, and copy segments with the SEGCOP command.

4.3 Executing and Nesting Segments

While you are defining and modifying segments, no graphics are generated. To draw defined segments, you must execute, or "SEGREF," them.

4.3.1 Executing Segments

When you are ready to display your segment structures, execute them. The SEGREF command executes the commands within a defined segment. Unless you specify otherwise (see Section 4.4), the geometry is now drawn.

4.3.2 Nesting Segments

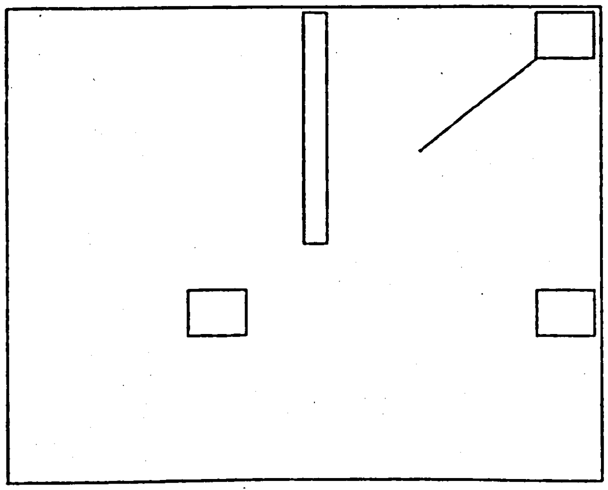
You can nest segment definitions to a depth of 16. With nesting, the outer, or primary, segment is called the parent and the segments it references are known as its children.

You can easily nest segments within other segments. Segment nesting is done by embedding SEGREF commands within a segment definition. These SEGREFs can reference both segments that have already been defined and segments not yet defined. If you draw a parent that references as-yet-undefined children, the displayed parent will not show those children segments and no message is generated to alert you.

Example 4.3 shows nesting by embedding multiple references to a child, Segment 3, within a parent, Segment 2. Note that child Segment 3 is referenced before it is defined. This is no problem because Segment 3 is defined before its parent is actually executed. The displayed parent will not show these children segments. The error message "SEGMENT NOT FOUND" is generated to alert you that the referenced segment is not defined.

Example 4.3 Nesting: Multiple References to a Second Segment From Within a Parent Segment

```
! SEGINI 0 128        ;Initialize display lists.
! SEGDEF 2            ;Begin definition of Segment 2.
$ RECREL 20 250       ; Definition begins with a rectangle...
$ MOVABS 100 100      ; followed by a MOVE...
$ DRWABS 200 200      ; followed by a line...
$ SEGREF 3            ; followed by as-yet-undefined Segment 3 (which
                        will be defined as four DRAWs making a
                        square)...
$ MOVABS 200 -100     ; followed by another MOVE...
$ SEGREF 3            ; followed again by Segment 3, a second square...
$ MOVABS -100 -100    ; followed by another MOVE...
$ SEGREF 3            ; followed again by Segment 3, a third square.
$ SEGEND              ;End the definition of Segment 2.

! SEGDEF 3            ;Begin definition of Segment 3.
$ DRWREL 0 50         ; Definition begins with a DRAW...
$ DRWREL 50 0         ; followed by a second DRAW...
$ DRWREL 0 -50        ; followed by a third DRAW...
$ DRWREL -50 0        ; followed by a fourth DRAW, making a square.
$ SEGEND              ;End the definition of Segment 3.

! SEGREF 2            ;Execute (draw) Segment 2.
```



There is no limit to the number of segment references at the same level. For example, Segment 3 (the square) can reference any number of other segments. If one of those segments, in turn, references another segment, the nesting level is three; it is the nesting level that is limited to 16.

Whenever a parent is invisible, so are its children. This is true even if the visibility attribute of any of those children is ON. However, if a parent is visible, any of its children can be made invisible by setting their visibility attribute to OFF with the SETATR command.

A child, likewise, always takes on the pick ID of its parent until a PICKID command within the child alters it. When a nested segment completes execution, the PICKID of the parent returns to its own value at the time of its invocation.

4.4 Execution Modes

With execution modes, you set the display controller for upcoming drawing, picking, highlighting, and unhighlighting operations. Execution modes are set using the EXMODE command. Use this command with the appropriate option before executing or referencing segments to perform the operation you want. EXMODE's options are:

- o Normal draw mode (the mode assumed until now during this discussion)
- o Pick mode
- o Hilite mode
- o Unhilite mode

4.4.1 Normal Draw Mode

Use this mode to draw graphics primitives, to execute other kinds of commands within segments (such as loading a register), and to execute macros. The default of the display controller is EXMODE NORMAL. A COLDstart or WARMstart restores the controller to this state.

4.4.2 Pick Mode

Execute EXMODE PICK before picking is to be performed (see Example 4.3). Picking is described in Section 4.5.

4.4.3 Highlight Mode

Use this mode to locally highlight graphics elements from within segments. Typically, local highlighting is done after picking to confirm the pick. First specify a highlight color by loading it into VREG 45. Then use EXMODE HILITE before drawing primitives in the highlight color (see Example 4 .4).

The contents of the segment ID register (SEGREG) and the pick ID register (PIDREG) determine which primitives are highlighted.

4.4.4 Unhighlight Mode

This mode is used to unhighlight objects that have just been highlighted (see Example 4.5). UNHILITE mode draws using the color stored in VREG 0 (the current color value). Execute EXMODE UNHILITE.

4.5 Picking

Picking is a display controller function which allows you to specify a location on the screen, and then request the Model One to return the identity of the graphics primitive or primitives located at that place.

Before picking, you must specify the size of the pick aperture and the center of the pick aperture. When you execute a pick operation, the Model One identifies the graphics primitives which either overlap or fall within the pick aperture. Such identification is called a pick hit.

Note: Do not pick area filled primitives. Picking an area filled primitive may cause the system to hang.

The Model One can identify either no pick hits, one pick hit, or multiple pick hits at one location. Each graphics primitive which is hit is labeled with a pick ID number and a segment ID number. These ID numbers are loaded into the registers and the pick buffer.

This is the picking process. The order of the first three steps can vary.

Step	Command
1) Specify the center of the pick aperture.	CLOAD 19 x,y
2) Specify the size of the pick aperture (or use default).	SETGL PICKAP
3) Specify the kind of information to store in the pick buffer (or use default).	SETGL PICKBUF
4) Enter pick mode.	EXMODE PICK
5) Reference the desired segment.	SEGREF n
6) Read the pick information.	RDPICK type
7) Enter normal draw or highlight mode.	EXMODE NORMAL or EXMODE HILITE

When you execute a pick operation, the Model One identifies the graphics primitives which either overlap or fall within a previously defined aperture. Such identification is called a pick hit. If no entities meet these conditions, there is no pick hit.

4.5.1 Specifying the Center of the Pick Aperture

Before picking, you must specify the pick aperture's center by loading its x,y values into Coordinate Register 19. Choose a center that is over or that crosses the graphics with information you want the Model One to read back and/or with the graphics you want to highlight.

Note that the aperture center can also be set with an interactive device by using the CMOVE command.

4.5.2 Specifying the Size of the Pick Aperture

The second step in the picking process is specifying the size of the pick aperture. Specify this square window with the SETGL PICKAP command. Specify the aperture in pixel units. Note that with the Model One/10, the pick aperture size must be either 8 x 8 (default) or 16 x 16.

4.5.3 Specifying the Kind of Information to Store in the Pick Buffer

Third, specify the kind of information you want the Model One to store in the pick buffer (assuming a pick "hit" occurs) when you subsequently pick primitives. You can specify one of four items:

- Segment IDs
- Pick IDs (as set by the PICKID command embedded within the segment definition)
- SEGID/PICKID pairs -- (default)
- Entire pick trees

  A pick tree is simply a list of pick IDs and segment IDs which reflects the hierarchy of each item picked. If an item is a child of some other segment, information can be derived about the parent segment as well. Pick trees are useful for applications using hierarchical structures and are described in Section 4.5.8.

To specify one of these options, use the SETGL PICKBUF command. In general, SEGIDs and PICKIDs require four bytes of storage. A tree requires eight bytes of SEGID/PICKID data for each level of segment nesting, plus a two-byte count of the nesting level.

After the pick operation, you can have the Model One read back the information you specified to store in the pick buffer with the RDPICK command (see Section 4.5.8).

4.5.4 Entering Pick Mode

Fourth, put the display controller into pick mode. To do this, use the EXMODE PICK command. At this point, the pick buffer is emptied and both SEGREG and PIDREG are set to -1.

4.5.5 Executing (Picking) Segments

The next step is executing, or picking, segments. Execute SEGREF and indicate the segment you want to pick. Note that nothing is drawn on the screen.

Every time the Model One/10 sees a PICKID command, it knows it has encountered a pickable entity with a unique identification. When it detects that geometry commands are overlapping the aperture, it has made the first "hit" and it loads SEGREG and PIDREG with the current segment ID and the current pick ID.

Only the first pick recorded loads the registers. These registers tell the application what pick ID has been found and can later be used for highlighting, unhighlighting, and deleting.

Simultaneously, pick information is being recorded in a RAM area called the pick buffer. This buffer area is 12K bytes.

In addition to being loaded into the appropriate registers, the first pick hit recorded is also stored in the pick buffer. The pick buffer contains this pick identity and the identities of all other pick hits recorded, unless the buffer overflows.

4.5.5.1 Pick Buffer Overflows

Any pick hits that occur after the buffer has filled are not recorded. Two counters are maintained to store two things -- the number of pick hits encountered and the number of pick hits recorded in the pick buffer. To determine if the pick buffer has overflowed, read these counters using the RDPICK PICKS command.

```
! SETGL IGNORE n
```

tells the display controller to ignore the first n pick hits. As a result, the application, after reading any pertinent information with RDPICK, can restart the pick operation and ignore all previously encountered pick hits if the pick buffer overflows. Use 0 to indicate "continue."

4.5.6 Reading Back Pick Information

Information stored in the pick buffer is read back with RDPICK. As discussed in Section 4.5.3, you have already specified the kind of information to store with the SETGL PICKBUF command. Depending on what you now request with parameters, RDPICK returns one of six types of information. Note that RDPICK parameters should correspond to SETGL PICKBUF parameters, although they need not always be exactly the same. For example:

SETGL PICKBUF parameter	stores	O.K. for
SETGL PICKBUF TREE	tree hierarchies	RDPICK PICKS, RDPICK TREEU RDPICK TREEP
SETGL PICKBUF SEGID	segment IDs	RDPICK PICKS, RDPICK SEGID
SETGL PICKBUF PICKID	pick IDs	RDPICK PICKS, RDPICK PICKID
SETGL PICKBUF SEGPID	segment IDs, pick IDs	RDPICK PICKS, RDPICK PICKID, RDPICK SEGID, RDPICK SEGPID

Note also that some readback with the RDPICK command is padded with minus ones.

Below is a list of all the RDPICK options along with a description of the information that the Model One returns with each. For each of the options, the <u>startent</u> parameter specifies the pick buffer entry to start reading from. The <u>totalent</u> parameter specifies the total number of entries to read back. If you specify 0 for <u>totalent</u>, the Model One updates PIDREG and/or SEGREG, but does not return any data.

o RDPICK PICKS 0 0

Returns two 16-bit numbers, the number of pick hits encounted and the number of pick hits recorded in the buffer (the total number of pick hits does not include ignored hits). This data is gotten from two display list counters, not the pick buffer. Therefore, no registers are ever updated. With this option, information is always returned. If there is no hit, zeros are returned; if there is no hit recorded because the pick buffer is full, zeros are returned.

o RDPICK PICKID startent totalent

  Returns the requested number of pick IDs. This option also loads the pick ID register (PIDREG) with the last pick ID read.

  - If the number of pick IDs you request exceeds the number of pick IDs stored, the remaining output is padded with minus ones
  - If pick IDs are not recorded, minus ones are returned.
  - If no pick hit occurs, minus ones are returned.

o RDPICK SEGID startent totalent

  Returns the requested number of segment IDs. This option also loads the segment ID register (SEGREG) with the last segment ID read.

  - If the number of segment IDs you request exceeds the number of segment IDs stored, the remaining output is padded with minus ones.
  - If segment IDs are not recorded, minus ones are returned.
  - If no pick hit occurs, minus ones are returned.

o RDPICK SEGPID startent totalent

  Returns the requested number of SEGID/PICKID pairs. This option also loads SEGREG with the last read segment ID and PIDREG with the last read pick ID.

  - If the number of SEGID/PICKID pairs you request exceeds the number of pairs stored, the remaining output is padded with minus ones.
  - If SEGID/PICKID pairs are not recorded, minus ones are returned.
  - If no pick hit occurs, minus ones are returned.

o RDPICK TREEU startent totalent

  Returns the requested number of pick trees in unpacked format.

  Trees in unpacked format are returned as a one-word count indicating the tree depth (segment-nesting depth) at the time of the pick, followed by exactly 16 SEGID/PICKID pairs, even if some "pairs" are returned as two minus ones. This option also loads the SEGREG and PIDREG with the segment ID and pick ID of the bottom level of the last tree read.

  - If the number of trees you request exceeds the number of trees stored, each remaining "tree" is returned as 00000 with 16 SEGID/PICKID pairs of minus ones.
  - If hierarchies are not recorded, a tree of depth one is returned.
  - If no pick hit occurs, minus ones are returned.

o RDPICK TREEP startent totalent

  Returns the requested number of trees in packed format.

  In packed format, trees are returned as a one-word count indicating the tree depth (segment-nesting depth) at the time of the pick, followed by x SEGID/PICKID pairs (where x is the nesting depth). This option also loads SEGREG and PIDREG with the segment ID and pick ID of the bottom level of the last tree read.

  - If the number of trees you request exceeds the number of trees stored, the remaining "trees" are returned as 00000 (no SEGID/PICKID pairs).
  - If hierarchies are not recorded, a tree of depth one is returned.
  - If no pick hit occurs, minus ones are returned.

4.5.6.1 Using the RDPICK Command Only for Updating Registers

You can use RDPICK to update SEGREG and PIDREG without also getting readback. To use RDPICK for updating registers, execute the command with a <u>type</u> parameter other than PICKS. Request that no data be returned by specifying "0" for the last parameter, <u>totalent</u>. For example:

```
o RDPICK PICKID 1 0  updates the pick ID register with the first entry.
o RDPICK SEGID 2 0   updates the segment ID register with the second entry.
o RDPICK SEGPID 12 0 updates both registers with the 12th entry.
```

The RDPID command reads back all the commands which have a pick ID equal to PIDREG and a segment ID equal to SEGREG. The RDPID command returns a list of commands which contains opcodes and parameters.

4.5.7 Entering Normal Draw or Highlight Mode

At this point in the picking process you should return to normal draw mode or highlight mode so that graphics can again be displayed or highlighted. Return to normal draw mode with the EXMODE NORMAL command or return to highlight mode with the EXMODE HILITE command.

4.5.8 Using Picking

An application that makes use of hierarchical local display lists generally needs to record entire pick trees so that the host can know where a pick occurred in a nested segment. Hence, a typical pick operation looks like this:

Example 4.4 Picking

```
! CLOAD 19 -210,-80       ;Load CREG 19 to specify the center of the
                           pick aperture.
! SETGL PICKAP 8 8        ;Set the pick aperture size to 16 x 16
! SETGL PICKBUF TREE      ;Specify to save entire trees in pick
                           buffer.
! EXMODE PICK             ;Enter pick mode.
! SEGREF 1                ;Execute (pick) Segment 1, the top level
                           segment.
! RDPICK TREEP 1 2        ;Read back tree hierarchy in packed format,
                           starting with the first entry and reading
                           back information on two entries.  Update
                           registers.
 00001
0000000001 0000000020
 00002
0000000001 0000000020 0000000002 0000000030
! EXMODE NORMAL           ;Restore execution mode to normal draw.
```

In the above example, the graphics commands to be searched are stored in Segment 1, the segment tree specified in the SEGREF command. Note that the SEGREF command always executes to completion; the full segment is traversed regardless of both the number of pick hits and whether or not the pick buffer is full. The first pick encountered loads the SEGREG and PIDREG to facilitate local highlighting.

4.6 Local Highlighting

Highlighting is the process of drawing graphics primitives from a given segment with a given pick ID in a specified highlight color. The highlight color can be used to make these entities stand out or even blink with use of the blink table.

The segment ID and pick ID of the primitives to highlight are contained in SEGREG and PIDREG. These registers are updated automatically during picking when there is a pick hit; or, you can update them interactively with either the RDPICK command and the SETGL command, both with the appropriate options. The highlight color is taken from VREG 45 which you must load before highlighting.

Unhighlighting is the inverse of highlighting. Unhighlighting is redrawing specified primitives based on the current color, taken from VREG 0.

Enter highlight and unhighlight modes with the EXMODE HILITE and EXMODE UNHILITE commands. Once you select either highlight or unhighlight mode, only primitives of the specified pick ID contained in the specified segment are drawn. All other commands are evaluated but they do not cause writes to image memory.

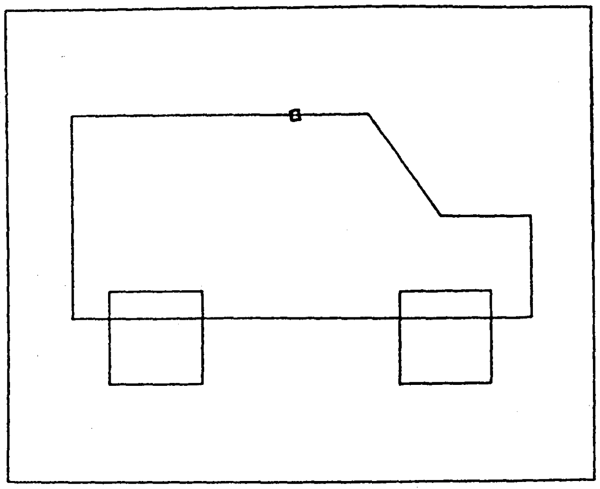
A local display list application generally highlights data that has just been picked. For a typical highlighting command stream, see Example 4.5.

The actual highlight operation begins when the segment ID and pick ID match those contained in the SEGREG and PIDREG and it continues until a PICKID command alters the PIDREG or until the segment ends. If there are nested segments within a highlighted segment, the nested segments are also highlighted if their visibility attribute is ON.

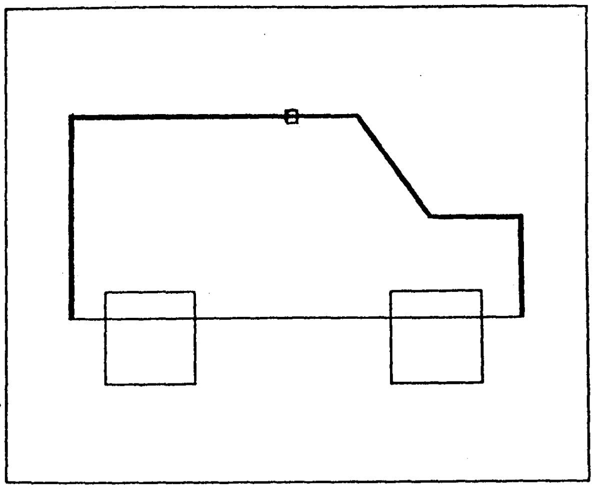
Example 4.5 Highlighting

```
! EXMODE NORMAL          ;Set execution mode to normal draw.
! SEGREF 1               ;Execute (draw) Segment 1 (See the figure
                          labeled NORMAL DRAW).
! CLOAD 19 0,140         ;Load CREG 19 to specify the center of the
                          pick aperture.
! SETGL PICKBUF SEGPID   ;Specify to store segment ID/pick ID pairs
                          (default).
! VLOAD 45 48,0,0        ;Load VREG 45 with highlight color, red.
! EXMODE PICK            ;Enter pick mode.
! SEGREF 1               ;Execute (pick) Segment 1.
! EXMODE HILITE          ;Enter highlight mode.
! SEGREF 1               ;Execute (draw) Segment 1, highlighting the
                          primitives of the first pick hit (See
                          figure labeled HILITE)
! EXMODE UNHILITE        ;Set execution mode to unhighlight.
! SEGREF 1               ;Execute Segment 1: draw in normal color the
                          geometry that has ben picked.
! EXMODE NORMAL          ;Resume normal draw mode.
```

NORMAL DRAW



HILITE



4.7 Transformations

The primitives within segments are defined in a 65,536 by 65,536 world coordinate system (WCS) ranging from (-32768,-32768) to (32767,32767). The device coordinate system (DCS) is the coordinate system of the display screen. DCS coordinates are calculated from WCS coordinates with a transformation applied.

Applications usually use a larger WCS area than can normally be seen at one time without applying transformations; they also often need to translate and rotate geometry. Raster's Display List Firmware allows scaling, translating, and rotating with a command that modifies, not the graphics primitives, but the display of those primitives.

How graphics are displayed is determined by a current transformation matrix which specifies the portion of the WCS which is visible and any scaling or rotation applied. The system keeps a default transformation matrix (3 x 2); you can modify this matrix with the XFORM2D command.

With one execution of XFORM2D, you can modify the current transformation matrix and achieve any combination of translation, rotation, and scaling. You define a transformation with XFORM2D by first specifying a transformation type.

Note: Transformations should not be used with area filled objects.

4.7.1 Types of Transformations

You can define the following four types of transformations with the XFORM2D command.

XFORM2D REL	Specifies a 2 x 2 matrix which performs relative scaling and/or rotation centered about the current point; matrix elements are concatenated with the previous transformation.
XFORM2D ABS	Specifies a 3 x 2 matrix which performs absolute scaling and/or rotation with translation.
XFORM2D XLATE	Specifies translation of the WCS origin to the current point.
XFORM2D RESET	Specifies a return to the default transformation (the identity).

4.7.1.1 What You Have to Specify

The parameters which you specify for the XFORM2D command vary according to the type of transformation.

The RESET and XLATE types require only one parameter, type.

The REL and ABS types require a reserved parameter, which must equal zero.

4.7.1.2 Default Transformation; The Identity

The default transformation is the identity, which specifies no scaling, no rotation, and no translation.

```
     x basis vector = (1, 0)
     y basis vector = (0, 1)
displacement vector = (0, 0)
```

To return to the default identity, use the RESET type of the XFORM2D command.

4.7.1.3 Translation With No Scaling or Rotation: XFORM2D XLATE

Use the XLATE type of the XFORM2D command to specify a translation with no scaling or rotation. Before executing the command, move the current point with the MOVABS or MOVREL command to reflect the translation you wish to effect (i.e., the displacement vector).

4.7.1.4 Absolute Transformation: XFORM2D ABS

For the absolute transformation type, ABS, you must specify a 3 x 2 matrix, consisting of the transformed x basis vector and y basis vectors, as well as the displacement vector.

Listed below are the sets of vectors (or 3 x 2 matrices) which describe scaling with translation, rotation with translation, and scaling followed by rotation with translation.

To specify no translation, use zeros in the last row.

To concatenate different transformations, apply the rules of matrix multiplication.

Scale by scale factors Sx and Sy
Translate by displacement vector D

```
     x basis vector =  (Sx, 0)
     y basis vector =  (0, Sy)
displacement vector =  (Dx, Dy)
```

```
Rotate by A degrees
Translate by displacement vector D
     x basis vector =  (cos(A), sin(A))
     y basis vector =  (-sin(A), cos(A))
displacement vector =  (Dx, Dy)

Scale by scale factors Sx and Sy
Rotate by A degrees
Translate by displacement vector D
     x basis vector =  (Sx  cos(A),  Sy  sin(A))
     y basis vector =  (-Sx  sin(A), Sy  cos(A))
displacement vector =  (Dx, Dy)
```

4.7.1.5 Relative Transformation: XFORM2D REL

For the relative transformation type, REL, you must specify a 2 x 2 matrix. This matrix includes the x and y basis vectors as listed above (4.7.1.4), but does not include the displacemant vector. To better illustrate the required input for the XFORM2D command, Tables 4.1 and 4.2 are provided. Table 4.1 gives the exact input for eight relative rotations.

Table 4.1 XFORM2D: Eight Relative Rotations

Rotate by	Command			COS	SIN	-SIN	COS
30 degrees	XFORM2D	REL	0	.866	.5	-.5	.866
45 degrees	XFORM2D	REL	0	.707	.707	-.707	.707
60 degrees	XFORM2D	REL	0	.5	.866	-.866	.5
90 degrees	XFORM2D	REL	0	.0	1	-1	.0
120 degrees	XFORM2D	REL	0	-.5	.866	-.866	-.5
135 degrees	XFORM2D	REL	0	-.707	.707	-.707	-.707
150 degrees	XFORM2D	REL	0	-.866	.5	-.5	-.866
180 degrees	XFORM2D	REL	0	-.1	.0	-.0	-.1

Table 4.2 shows the exact input for three absolute scalings.

Table 4.2 XFORM2D: Three Absolute Scalings

Scale by				Scale Factor Times x	0	0	Scale Factor Times y	Trans-lation Factor Times x	Trans-lation Factor Times y
					Command				
2	XFORM2D	ABS	0	2	0	0	2	0	0
4 1/2	XFORM2D	ABS	0	4.5	0	0	4.5	0	0
5	XFORM2D	ABS	0	5	0	0	5	0	0

After you execute the XFORM2D command, the current point remains unchanged. When the new matrix is properly set, the current point remains at the location you set it to and the internally stored DCS current point is recalculated using the new matrix.

4.7.2 Reading Back and Using the Current Transformation

The read back command RDXFORM returns the current transformation matrix in the same format as that used with the XFORM2D ABSOLUTE command.

You can use the current transformation matrix to perform calculations locally within the Model One. Do this with the XMOVE command which maps a specified x,y coordinate from one coordinate space into another coordinate space. With the appropriate option, the command maps either from WCS to DCS or from DCS to WCS.

4.7.3 More About the Current Point and Transformations

The current point, in CREG 0, is updated by 2-D geometry commands such as MOVABS, MOVREL, and DRWABS. Whenever a 2-D transformation is active, the current point should be thought of as a WCS current point; it resides in the same space as the geometry commands which update it. For example:

Example 4.6a The Current Point and Transformations

```
! XFORM2D REL 0 2 0 0 2
                              ;Define a relative transformation which
                               scales by a factor of two.
! MOVABS 124,45               ;Update the current point (CREG 0) to
                               124,45 in the WCS.
! DRWABS 30,30                ;Draw a line to 30,30 in the WCS
                               (from DCS 248,90 to DCS 60,60).
```

A transformation affects the manner in which the current point is interpreted but it is important to remember that this point is not changed by the XFORM2D command, nor by PUSH XFORM and POP XFORM (see Section 4.8 for a discussion of the PUSH and POP commands). This ensures that the value loaded into a CREG is not "magically" changed. For example:

Example 4.6b The Current Point and Transformations

```
! XFORM2D RESET               ;Reset to the identity transformation.
! MOVABS 20,30                ;Move the current point to 20,30.
! POINT                       ;Draw a point at 20,30.
! XFORM2D ABS 0 2 0 0 2 0 0
                              ;Define a new absolute transformation which
                               scales by a factor of two.
! READCR                      ;Read the current point; it has not changed
 0020  0030                    but it now corresponds to a different point
                               on the screen.
! POINT                       ;Draw a point; this point is now at 40,60 in
                               DCS coordinates.
```

The current point should be explicitly reset (MOVABS 0,0) each time the current transformation is redefined. This practice also facilitates legibility of code by ensuring that the current point always be at the origin of the a new transformation.

4.8 Stacking

For sophisticated applications, the Model One/10 display list command set allows application programs to store certain items in internal RAM. Two commands, PUSH and POP, fall into the category of context-saving commands.

With the PUSH command you can save registers, flags, variables, and transformation matrices. These items are placed on a stack until you want to restore them, at which time you execute the POP command to restore that context. Note that no type checking is enforced by the PUSH/POP operation. For example, this may produce unpredictable results:

```
! PUSH XFORM 0
! POP CREG 0
```

The Model One/10 has 2,000 stack locations. PUSHing a 2-D transformation matrix uses 11 locations, PUSHing a CREG uses two locations, PUSHing a VREG uses three locations, and PUSHing a variable uses one location.

4.8.1 Pushing and Popping Transformations

When you save, or PUSH, a transformation, a duplicate of that transformation is made, the six transformation elements are placed on top of the stack, and the stack pointer is updated.

The POP XFORM 0 command replaces the current transformation with the six elements of the last PUSHed transformation. The DCS current point is recalculated by transforming the WCS current point (in CREG 0) through the new, popped transformation. The command also adjusts the stack pointer to point to the new top item on the stack.

Example 4.7 Pushing and Popping a Transformation

```
! SEGDEF 10           ;Begin definition of a segment.
$ MOVABS 0 0          ;Move to the origin.
$ DRWABS 3 5          ;Draw the letter 'A'...
$ DRWABS 5 0
$ MOVABS 2 3
$ DRWABS 4 3
$ MOVABS 5 0          ;End at the bottom of the right leg of the 'A'.
$ SEGEND              ;End the definition of Segment 10.
! MOVABS 2 1          ;In the current space, move to (2,1).
! PUSH XFORM 0        ;Push the current transformation.
! XFORM2D REL 0 .5 0 0 .5
                      ;Define a new relative transformation, one that
                       scales down by 1/2.
! SEGREF 10           ;Draw the letter 'A'.
! POP XFORM 0         ;Restore  the PUSHed transformation.
! READCR 0            ;The current point is now (5,0) - which is NOT near
                       the bottom of the right leg of the 'A'. The contents
                       of CREG 0 have not changed at all but their context
                       has.
```

4.8.2 Pushing and Popping the Current Point

PUSH and POP also have an option to save/restore the current point (the WCS and DCS current point). The commands selecting this option are PUSH/POP CREG CURPNT.

The following example is similar to Example 4.7. However, in this example we save the current point before defining a new transformation and executing Segment 10 (which changes the current point). At the end of the example, we restore the current point with the POP command.

Example 4.8 Pushing and Popping the Current Point

```
! MOVABS 2 1          ;In the current space, move to (2,1).
! PUSH XFORM 0        ;Push the current transformation.
! PUSH CREG CURPNT    ;Push the current point.
! XFORM2D REL 0 .5 0 0 .5
                      ;Define a new relative transformation -
                       a half-scaling matrix.
! SEGREF 10           ;Draw 'A'.
! POP CREG CURPNT     ;Restore the previous current point.
! POP XFORM 0         ;Restore the previous transformation.
! READCR 0            ;Read CREG 0; the current point is now (2,1), where
                       you started from.
```

4.9 Display List Firmware Commands By Functional Category

The Display List commands can be divided into the following functional categories:

- o Segment creation and modification commands
- o Attribute setting commands
- o Segment execution commands
- o Readback commands

Other related groups of commands include:

- o Transformation commands
- o Stacking commands
- o Picking commands

Segment creation and modification commands create and edit segment structures. Attribute setting commands are used to set and modify segments attributes (visibility and pickability) and to assign commands to pick IDs.

The segment execution commands set the display controller execution mode and execute segments. Readback commands return information about the list structure. Transformation commands perform translations, scaling, and rotations. Finally, the stacking commands save and restore registers, flags, variables, and transformations.

A complete list of Model One/10 Display List, functional category, is given in Table 4.3. Each of these commands is documented fully in the Model One/10 Command Reference (Revision 1.0).

Table 4.3 Display List Firmware Commands

Segment-creation and -modification Commands

SEGINI (Initialize All Segments)
SEGDEF (Begin A Segment Definition)
SEGAPP (Segment Append)
SEGEND (Segment End)
SEGDEL (Segment Delete)
SEGCOP (Segment Copy)
SEGREN (Segment Rename)
DELPID (Delete Pick Identification Number)

Attribute-setting Commands

PICKID (Assign Pick Identification Number)
SETATR (Set Attribute)
SETGL (Set Global Parameter)

Segment-execution and Picking Commands

EXMODE (Set Execution Mode of Display Controller)
SEGREF (Execute a Segment)

Readback Commands

RDPICK (Read Back Pick Information)
RDREG (Read Back Segment ID and Pick ID Registers)
RDXFORM (Read Back Current Transformation)
SEGINQ (Segment-attribute Inquiry)
SYSTAT (Read Back Defined-segment or Existing-RAM Information)

Transformation Commands

XFORM2D (Transform All 2-D Coordinates)
XMOVE (Map One (x,y) Location)

Stacking Commands

PUSH (Save Specified Current State)
POP (Restore "PUSH" State)

5. FORTRAN Library

5.1 The Model One/10 FORTRAN Library

The Model One host FORTRAN library, called ONELIB, gives you access to all of the Model One/10 commands through subroutine CALLs from the host application program.

To send any command from the host to the Model One/10, issue a CALL to a ONELIB subroutine. All ONELIB command subroutines have the same name as the local command mnemonic. For example, these FORTRAN lines:

```
    CALL MOVABS (0,0)
    CALL CIRCLE (100)
    CALL DRWABS (20,50)
```

are identical to typing these commands at the local terminal:

```
    ! MOVABS 0,0
    ! CIRCLE 100
    ! DRWABS 20,50
```

The FORTRAN library contains several levels of subroutines. The subroutines which generate Model One/10 commands are called by the applications program; those subroutines in turn call low level subroutines to perform I/O between the host and the Model One/10.

Each Model One/10 command has an equivalent FORTRAN subroutine. For example, the FORTRAN call for the MOVABS command is:

```
     CALL MOVABS (IX,IY)
```

IX and IY are INTEGER*2 variables; the order for parameters is the same as for locally typed commands.

5.1.1 Conventions

For all FORTRAN subroutines, the following conventions are used. Parameters are

- o given in the same order as for locally-typed commands.
- o generally INTEGER*2 (ranging from -32,768 to 32,767).
- o never changed by the FORTRAN call (except in the case of readback commands).

5.2 Installing the FORTRAN library on the VAX

5.2.1 Documentation Included on Library

The Model One/10 FORTRAN library for the VAX includes several documentation files, describing the FORTRAN library.

File	Description
README.DOC	Describes how to install the ONELIB software.
RELNTSV*.TXT	Describes release-specific information on the software. The * is replaced by the version number; for example, RELNTSV81.TXT describes release 8.1 of the library.

5.2.1.1 Suggestion

The INSTALL command (see below) prints a lot of information on the structure of the library.

You may want to use a hardcopy terminal.

5.2.2 How to Install the FORTRAN Library

Step	Action
1	Create a directory for the top level of the Raster Technologies software directory tree.

This directory can be a root-level directory or a sub-directory of an existing tree.

```
$ CREATE /DIRECTORY [RASTER]
```

2	Make the new directory the default directory.

```
$ SET DEFAULT [RASTER]
```

3	Mount the distribution tape containing the software.

```
$ MOUNT MSA0:  RASTER/OVER=OWNER
```

4	Copy all the files on the tape into the new directory.

Note: The directory must be empty before this command is executed (if you just created it, as in Step 1), it will be empty.

```
$ COPY MSA0:*.*;* *.*;*
```

5	Execute the command procedure, INSTALL.COM, (which was copied from the tape in Step 4).

```
$ @INSTALL
```

Result: INSTALL.COM asks the following three questions. The normal responses are listed in parentheses.

- Have files been copied from tape? (Yes)
- Do you want to change the default device type? (Yes)
- Do you want to install a demonstration program for the Model One/380 (No).

These questions are discussed below.

5.2.2.1 Question 1: Copying Files from Tape

The INSTALL command first asks

"Are the distribution tape files copied to this directory?"

"Y" is the normal response. If you have forgotten to copy the files, you should respond "N". Otherwise, the command generates a number of error messages.

5.2.2.2 Question 2: Default Device Type

The second question in the installation procedure is

> "Do you want ONELIB to provide default support for the devices listed in category B above?"

The two categories of devices listed on the screen are:

Model One/25 Class:

A: Model One/20, Model One/25, Model One/25-S, Model One/40, and Model One/60.

Model One/80 Class:

B: Model One/10, Model One/75, Model One/80, Model One/80-SL, and Model One/380.

The usual response to the question is "Y", since the Model One/10 is included in category B. The result of responding "Y" is that the library is built to provide default support for the Model One/80 class of devices rather than the Model One/25 class.

An application program can override the default which is set at installation by calling the RTSET routine before calling RTINIT.

5.2.2.3 Question 3: Optional Model One/380 Demonstration Program

The last question in the installation procedure is

> "Do you want the ONELIB Model One demo programs?"

The normal response is "N". If you respond "Y", a Model One/380 demonstration program is included in the RASTER_TEST directory. This program only works on a 40-bit Model One/380. This program and its binary files consume up to 1500 disk blocks.

5.2.3 Results of Using the INSTALL Command

The INSTALL command creates a number of sub-directories and command procedures.

5.2.3.1 Sub-directories

When the INSTALL command has been executed, you will have the following sub-directories.

Sub-directory	Description
[.DRIVER]	Contains the DMA driver (not applicable to the Model One/10)
[.ONELIB]	Contains the ONELIB object module library.
[.SOURCE]	The root directory for all source files.
[.SOURCE.COMMANDS]	Holds Model One output-only command source files.
[.SOURCE.IO]	Contains the Model One I/O-related command and driver source files.
[.SOURCE.UTILS]	Contains the ONELIB utility source files.
[.TEST]	Holds the test and example programs (and the demo program if requested during installation).

5.2.3.2 Command Procedures

The INSTALL procedure also creates several command procedures in the root directory. In this list, directory [RASTER] is assumed.

Command Procedure	Description
SYSLOGIC.COM	Defines system-wide logical names. It can be executed as part of the SYS$MANAGER:SYSTARTUP.COM command procedure. The logicals created are:

```
RASTER_DIRECTORY  (points to [RASTER])
RASTER_ONELIB     (points to [RASTER.ONELIB])
RASTER_LIBRARY    (points to [RASTER.ONELIB])
RASTER_DRIVER     (points to  [RASTER.DRIVER])
RASTER_SOURCE     (points to [RASTER.SOURCE])
RASTER_COMMANDS   (points to [RASTER.SOURCE.COMMANDS])
RASTER_UTILS      (points to [RASTER.SOURCE.UTILS])
RASTER_IO         (points to [RASTER.SOURCE.IO])
RASTER_TEST       (points to [RASTER.TEST])
```

LOGICALS.COM	Defines logical names in the process logical name table if desired. The logicals are the same as in SYSLOGIC.COM.

Command Procedure	Description
SYMBOLS.COM	Defines two global symbols used for linking to the library. LINKONE links an object module to the Model One library. LINKONEDB links an object module, using the /DEBUG specifier on the LINK command. These symbols can be used as follows: $ FORTRAN myprogram $ LINKONE myprogram $ RUN myprogram Note: The library modules are compiled with the /NODEBUG/OPTIMIZE compiler switches. If you wish to examine any module using the debugger, you will need to recompile the module with the /NOOPTIMZE/DEBUG switches. This object module will then need to be explicitly specified when linking with the library.

5.3 Using the VAX FORTRAN Library

5.3.1 Compiling User Programs

The library source modules are compiled with the /NODEBUG/OPTIMIZE compiler switches to increase their performance. If you wish to examine any ONELIB module using the symbolic debugger, you will need to recompile the module separately with the /DEBUG/NOOPTIMZE switches. This object module will then need to be explicitly specified when linking with the library.

For example, the following commands allow the synbolic debugger to use the ONELIB routine CIRCLE.

```
$ COPY RASTER_COMMANDS:CIRCLE.FOR *.*
$ FORTRAN/DEBUG/NOOPTIMIZE MAINPROG,CIRCLE
$ @LINKDEB MAINPROC,CIRCLE
$ RUN MAINPROG
```

5.3.2 Linking User Programs

The SYMBOLS.COM file includes the following defintions, relating to linking user programs:

```
   LINKONE:==@RASTER_DIRECTORY:LINK          (for normal linking)

   LINKONEDB:==@RASTER_DIRECTORY:LINKDEB     (for linking and debugging)
```

The normal program sequence is:

```
$ EDIT programname
$ FORTRAN programname
$ LINKONE programname
$ RUN programname
```

5.3.3 Specifying the Model One/10 for Serial I/O

To specify the Model One/10 for serial I/O, you call RTINIT with the logical device name of the terminal to which the Model One/10 is connected. The arguments to RTINIT are the logical device name and the number of characters in the name. Three examples are given below.

```
    call RTINIT ('DEV', 3)
    call RTINIT ('TT', 2)
    call RTINIT ('TTA2:', 5)
```

If you use a logical device name, it must be assigned to the terminal driver. For example,

```
    $ ASSIGN TTxx DEV
```

You must call RTSTOP before attempting to talk to a different Model One with a second call to RTINIT.

5.4 Verifying the VAX Library

To verify the library, you should run the two serial tests which are included in the RASTER_TEST directory. Execute the following commands.

```
$ SET DEF RASTER_TEST
$ ASSIGN TT DEV
$ RUN CPRIMS1           (tests output)
$ RUN BREADCR           (tests readback functions)
```

6. Central Processor

The Model One/10 central processor is a Z8001 16-bit microprocessor. The Z8001 processor has 16-bit and 32-bit logic and arithmetic capabilities, including multiply and divide instructions.

The central processor runs a ROM firmware program which:

1. Decodes and processes incoming graphics and alphanumeric commands.
2. Provides interactive device support.
3. Maintains input/output buffers.
4. Controls display attributes.
5. Processes locally stored macros and Display List segments.

The central processor reads from and writes to image memory through I/O-mapped Z8001 addresses. Two serial ports are provided for external communications. Figure 6.1 diagrams the central processor.

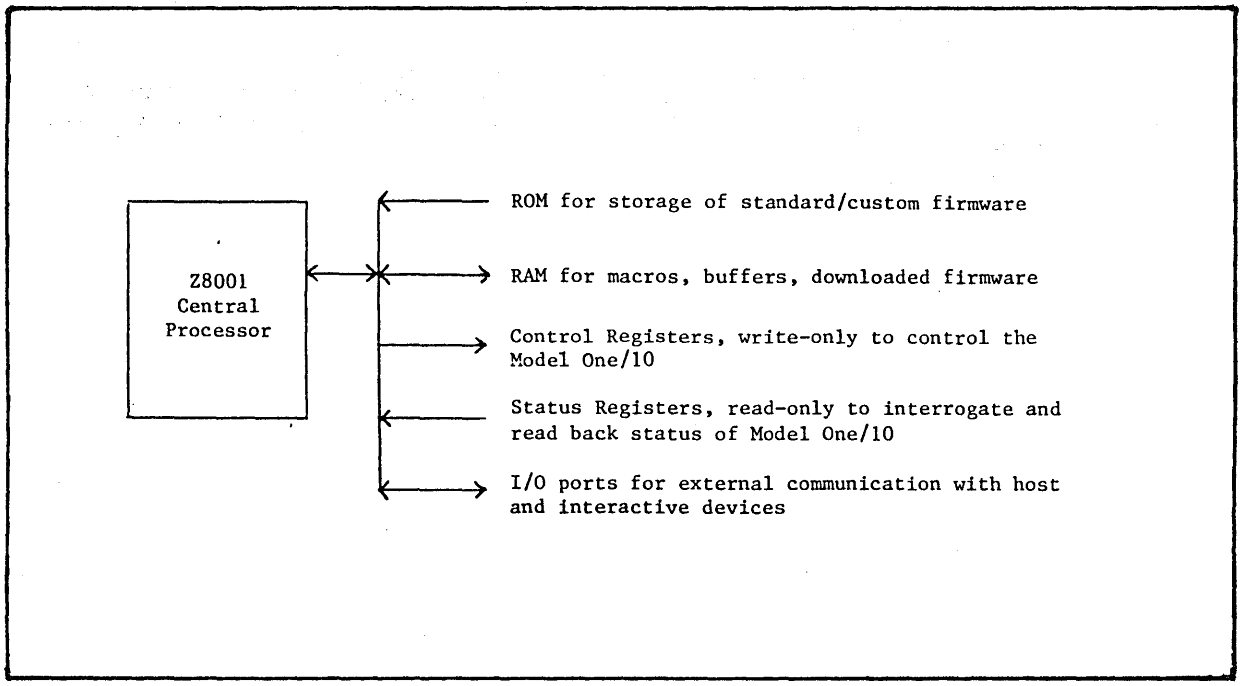

Z8001
Central
Processor
ROM for storage of standard/custom firmware
RAM for macros, buffers, downloaded firmware
Control Registers, write-only to control the Model One/10
Status Registers, read-only to interrogate and read back status of Model One/10
I/O ports for external communication with host and interactive devices

Figure 6.1 Central Processor Addressing and I/O

6.1 Processor Memory

The Z8001 has a direct addressing range of eight megabytes. The Model One/10 provides 256 Kbytes of physical memory. This memory space is available for macro and segment storage, configurable buffers, and user-written software. The Z8001's memory map is divided into three parts: ROM, RAM, and I/O-mapped control/status registers.

6.2 Input/Output Ports

The Z8001 central processor communicates with external devices through five ports, two of which are RS-232C. The ports are marked "keyboard," "tablet," "trackball," "printer," and RS-232. The RS-232 port is for host interfacing; the "printer" port provides parallel output. Using either Model One/10 Setup mode or the SYSCFG command, you can configure the host port for: parity, baud rate (from 75 baud to 19.2K baud), XON/XOFF protocol, and others. In the standard firmware, serial input and output are interrupt-driven, with firmware-configurable buffers to maximize processor utilization.

6.3 Control and Status Registers

The Z8001 central processor uses I/O-mapped control/status registers for internal communications. Control registers are write-only locations that cannot be read back. Status registers are read-only locations and cannot be changed by a write operation.

The control registers fall into two major subsystems: vector generator and display control. Two groups of status registers interrogate each subsystem and monitor its operation.

7. Vector Generator

A hardware vector generator is used to off-load digital line generation from the central processor. The Model One/10's vector generator is a custom VLSI processor which performs vector-to-raster conversion. Using control registers, the central processor sends the vector generator the start-point, end-point, and pixel value information. The start-point and end-point give the address of the pixels in image memory which are to be connected with a line.

The vector generator has the image memory available to it approximately 25% of the time. The vector generator has an average pixel-writing rate of 1,200,000 pixels per second.

7.1 Pattern Register

The vector generator provides patterned lines, such as dashed or dotted lines. The firmware command VECPAT accesses a 16-bit control register which is used to provide the line patterns. The same line pattern can also be used for both primitive and area fills.

At every pixel along the register, the pattern register is left-shifted by one bit. If the shift causes a carry to occur, the pixel currently being processed is written into image memory normally. If no carry occurs, the pixel is suppressed. The carry bit is shifted into the right-most bit of the pattern register so that the bit pattern recycles every 16 pixels. Because the pattern register is not reinitialized after each vector, the pattern continues properly from one vector to the next.

8. System and Image Memory

The system memory of the Model One/10 consists of 256 Kbytes of dynamic RAM. The Model One/10's image memory has 640 Kbytes.

The Model One/10 uses 64K RAM chips contain all of image memory on a single printed circuit board. Image memory is configured as 1024 x 512 x10.

8.1 Write-enable Masks

Each bit plane of image memory can be selectively write-protected or write-enabled by using write-enable masks. A set of control registers gives the central processor access to the write-enable masks. You can modify the write masks with the WRMASK or WMSK16 commands.

8.2 Read Masks

The output from image memory passes through a set of read masks which can force the output of any bit plane to zero. These masks, which you can modify with the RDMASK or RMSK16 commands, are useful in double-buffering and other real-time applications.

8.3 Image Memory Output

In the Model One/10, the image memory output is routed into three color look-up tables (LUTs). These LUTs are used to drive 8-bit digital-to-analog converters (DACs) for each primary color (see Figure 8.1).

Several commands control the output from the LUTs: LUTR, LUTG, and LUTB control respectively the red, green and blue LUTs; LUTA and LUT8 modify all three LUTs; and LUTRMP can be used to modify the default linear ramping of the look-up tables. There is also the LUT16 command which corresponds to the LUT8 command but allows access to all 1024 LUT entries.

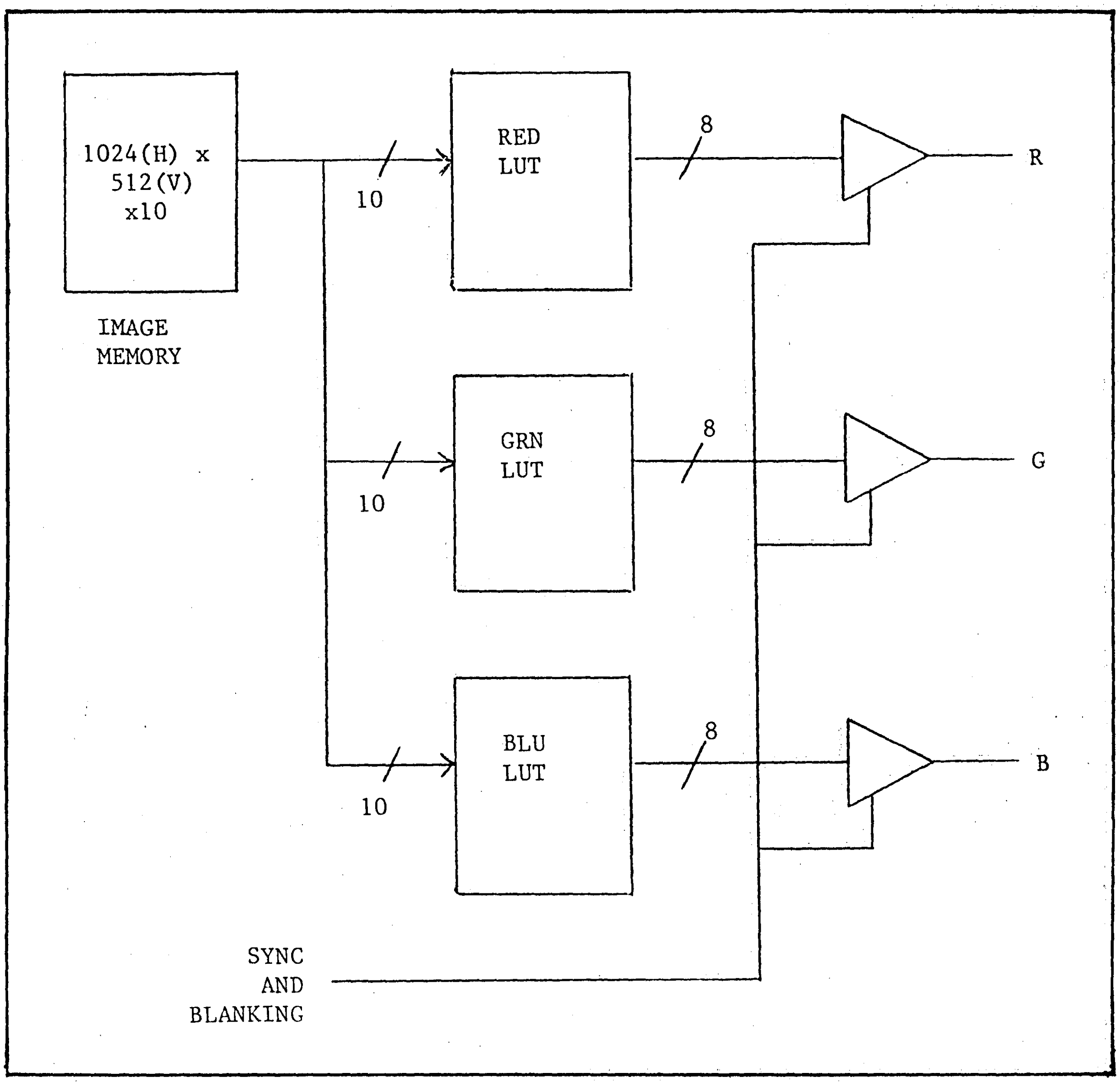


Figure 8.1 Image Memory Output Diagram

8.3.1 Look-up Tables

In the Model One/10, three 10-bit-in, 8-bit-out LUTs (1024 x 8) are used to drive the video DACs.

The default look-up table values for the Model One/10 from index 128 to 255 are white, for use with the VT100.

8.3.2 Video Amplifiers

A separate video amplifier stage is used for each of the three primary colors: red, green, and blue. A summing amplifier for each channel adds blanking and sync pulses to the video signal and drives a 75 ohm output buffer. The output of this buffer is connected to external BNC connectors for output to a standard color video. The Model One/10 does not provide composite sync on the R, G, or B video outputs. Separate horizontal and vertical sync outputs are provided, however.

8.4 Using the LUT16, RMSK16, VAL16, and WRMSK16 Commands

The RMSK16, VAL16, and WMSK16 commands swap the high (left) and low (right) bytes that make up the 16-bit word used for the parameters for each command. In other words, 52 in hexadecimal must be entered as #5200, not #0052.

However, the LUT16 command does not swap bytes. In other words, 52 in hexadecimal must be entered as #0052.

The host library supplied by Raster Technologies eliminates this issue of inconsistency; you can specify the parameters without swapping the bytes when you use the RMK16S, VAL16S, or WMK16S calls.

The following examples specify parameter values using hexadecimal values. You should use hexadecimal values for RMSK16, VAL16, and WMSK16 parameters.

Example 1: The following example indicates how to use the LUT16 command to load look-up table index 128 (decimal) with all red, and then select index 128 to draw a vector.

```
! WMSK16 #FF03              ; Set the write and read masks to write into and
! RMSK16 #FF03              ; read from all 10 bit planes:
                            ; FF is 8 bits for graphics
                            ; 03 is for the overlay planes.

! LUT16 #0080 255 0 0       ; Load look-up table index 128 (decimal) with all
                            ; red.  Note that 128 decimal = 80 hexadecimal,
                            ; which for the LUT16 command is not swapped.

! VAL16 #8000               ; Select look-up table index 128 (decimal).  Note
                            ; that 128 decimal = 80 hexadecimal, which for
                            ; the VAL16 command is swapped, so the "80" is in
                            ; the high (left) byte.

! DRWREL 100 0              ; Draw a red vector.
```

Example 2: This example builds on Example 1, drawing a green vector using look-up table index 350 (decimal).

```
! LUT16 #0154 0 255 0       ; Load look-up table index 350 (decimal) with all
                            ; green.  Note that 350 decimal = 154 hexadeci-
                            ; mal, which for the LUT16 command is not swap-
                            ; ped.
```

```
! VAL16 #5401            ; Set current pixel value to value of look-up
                         ; table index 350 (decimal).  Note that 350 deci-
                         ; mal = 154 hexadecimal, which for VAL16 is
                         ; swapped, so that the "54" is in the high (left)
                         ; byte, and the "01" in the low byte.

! DRWREL 100 0           ; Draw a green vector.
```

9. Display Control

The Model One/10 provides control of several graphics display attributes in hardware. The firmware then supports these features through control registers which are accessed by the central processor.

9.1 Screen Origin

A pair of control registers is used to preset the screen refresh address counters. These counters control the placement of the screen origin in image memory and are used for panning. You can use the SCRORG command to modify these control registers.

The screen origin can be placed on 16 pixel boundaries horizontally and single pixel boundaries vertically. Changes to the screen origin are performed only during vertical blanking to eliminate tearing.

9.2 Display Scaling

Display List firmware supports display scaling of graphics. Scaling varies, depending on the commands and parameters used.

9.3 Flooding the Screen

Using the CLEAR command, you can clear the currently defined window with a desired pixel value. In addition, the FLOOD command does the same thing as CLEAR, but to the full image memory.

Note that the Model One/10 also supports extensive display control of text (see Section 13).

10. Interactive Device Support

The Model One/10 supports interactive input devices, such as digitizing tablets and mice. A set of firmware commands is available to configure the interactive device support for any application.

10.1 Digitizing Tablet

The Model One/10's digitizing tablet has an 11" x 11" active surface area and can be used to enter coordinate information and position crosshairs. Tablets are available with a simple stylus or a puck. Macros can be easily executed with these buttons (see Section 3.11).

You can use the XYDIG command to change the scaling of the digitizing tablet.

Note: The tablet scales at a ratio of 4/3 in the y direction. This results in objects drawn on the tablet being distorted on the screen.

Part II

INSTALLATION OF THE MODEL ONE/10

11. Unpacking and Cabling the Ports

This section describes installing the Model One/10: unpacking it and connecting it to the host computer, to I/O devices, and to optional peripherals.

11.1 Unpacking

The Model One/10 terminal (without options) comes in two packing cartons. One contains the monitor and power cord and the other has the keyboard with attached cable. (The controller is contained within the monitor.) Unpack the Model One/10 carefully, saving the boxes. If there seems to be any damage, contact Raster Technologies or your local representative immediately.

After unpacking, you should have:

- o Monitor
- o Power cord
- o Keyboard
- o RS-232C digitizer cable, if you ordered a digitizing tablet
- o RS-232C alphanumeric terminal cable
- o FORTRAN library tape, if ordered
- o Model One/10 Manual and Release Notes
- o Other cables, if ordered

Next, check the serial number of each item. These numbers are labeled on the back of each piece and are also on the outside of each packing carton. Record these serial numbers in the place provided below; you will need to know these numbers if you call Raster Technologies regarding repair of the equipment.

Record of Model One/10 Serial Numbers

Monitor______________________________

Keyboard ______________________________

Tablet ______________________________

11.2 Environmental and Technical Specifications

System Processor:	16-bit, cycle time 167 nanoseconds
Graphics Processor:	Full custom VSLI, one million pixels per second
Image Memory:	1024H x 512V, 10 bit planes
Display Resolution:	640 x 480 window, 60 Hz non-interlaced
Pixel Update Time:	800 nanoseconds per pixel, 1 to 10 bits per pixel
System Memory:	256K bytes
Display List Capacity:	Maximum capacity approximately 80,000 primitives including filled polygons, circles, and text characters
Host Interface:	RS-232C, maximum 19.2K baud
Local Device Interfaces:	Mouse, data tablet, local hardcopy, RGB video
Operating Temperatures:	0 degrees C to 35 degrees C
Power Requirements:	120 VAC 60 Hz, 240 W
Physical Dimensions --	
Terminal:	17-1/4 x 14-1/2 x 13-1/4
Keyboard:	19 x 7 x 1-3/4
Weight:	52 pounds

The top, back, and side vents must not be blocked. The Model One/10 terminal requires at least one-inch clearance on all sides. It also needs a three-prong, grounded outlet.

11.3 Overview of the Model One/10's Ports

The Model One/10's central processor communicates with external devices through five ports, marked "keyboard," "tablet," "trackball," "printer," and "RS-232." Two of these, the host port and the tablet port, are serial RS-232C ports. The RS-232 port is for host interfacing; the "printer" port provides local hardcopy.

The Model One/10 terminal is supplied from the factory with default configuration options chosen to match commonly-used settings. Using Model One/10 Setup mode (see Section 13) or the SYSCFG command, you can configure the host port as you wish for: parity, baud rate (from 75 baud to 19.2K baud), XON/XOFF protocol, and control character acceptance. Serial input and output are interrupt-driven, with firmware-configurable buffers to maximize processor utilization.

11.4 Connecting the Monitor, Keyboard, and Tablet

After you have unpacked the monitor, keyboard, and tablet and have recorded their serial numbers, look at the monitor's back panel (see Figure 11.1). Notice that there are five ports, or connections, for cables. These ports are labeled:

- o Keyboard
- o Tablet
- o Trackball
- o Printer
- o RS-232 (host port)

Before you attach any cables, position the monitor where you will be using it. Attach the keyboard first. Put the keyboard's cable connector into the port labeled "keyboard." Note that the key in the cable's connector aligns with the top of the Model One/10's connector.

Next, connect the Model One/10 to the host. Connect the mating connector on the host cable into the RS-232 port. You will find that the pattern of the pins inside the mating connector matches the pattern of this port's connector.

In addition, notice that the pins are numbered. For the host port, Pin 2 = TRANSMIT data from the Model One/10 and Pin 3 = RECEIVE data to the Model One/10.

Next, connect the data tablet to the "tablet" port. For this port, Pin 2 = RECEIVE and Pin 3 = TRANSMIT. The tablet/mouse port has its baud rate fixed at 9600 baud and is set for 8 data bits, no parity, and one stop bit.

Last, attach any optional devices into the appropriate ports. When you are ready to type at your fully cabled Model One/10 terminal, adjust the leg under the front of the monitor to a tilted position, if desired.

You are now ready to test the Model One/10 terminal (see Section 11.6).

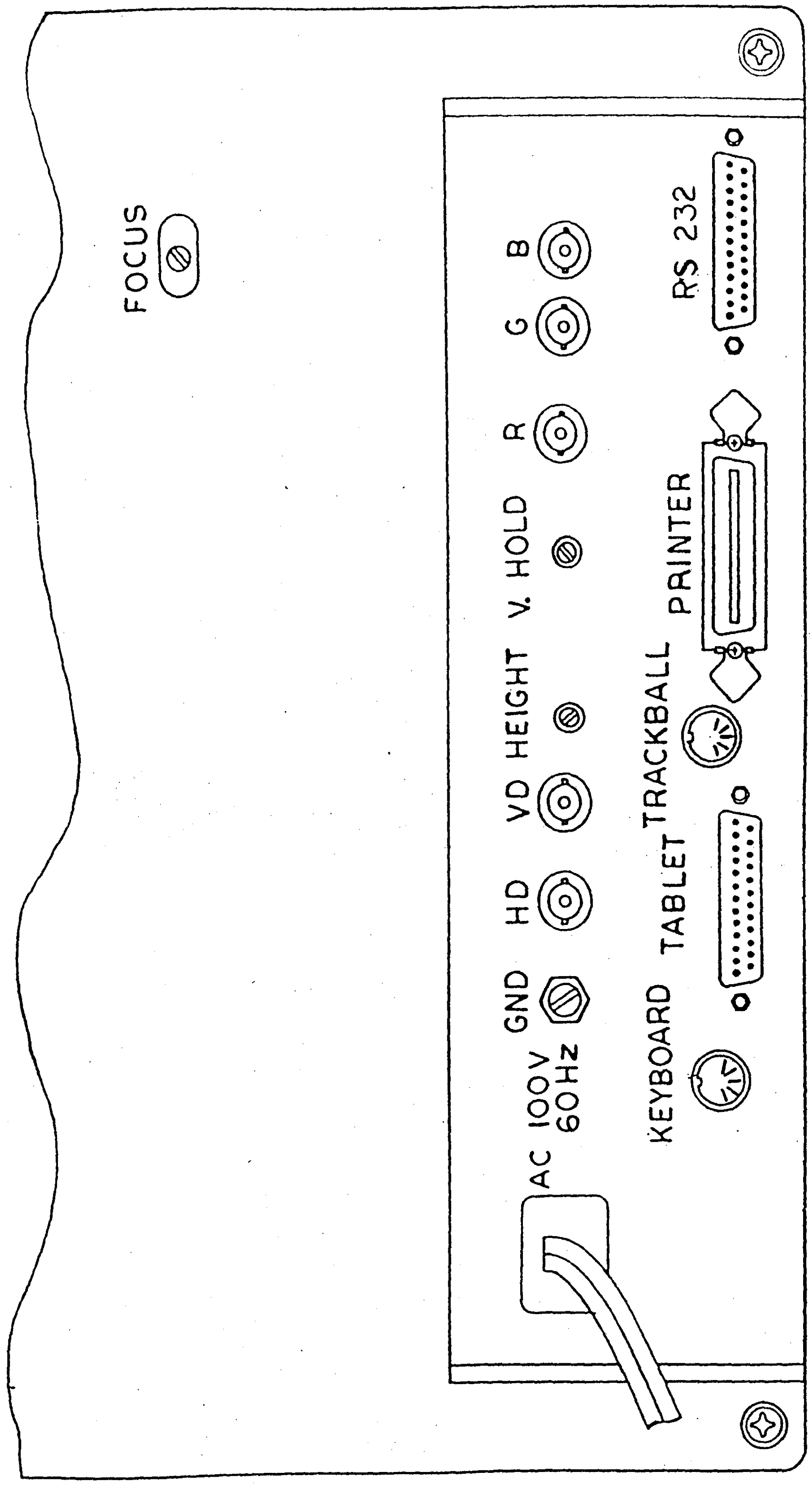


Figure 11.1 Model One/10 Back Panel

11.5 Printer Interface

Use the following procedure to connect the printer to the Model One/10:

- Attach the Centronix cable to the Centronix port.
- Press the ENABLE button on the printer to enable the printer.

Important: If the printer is attached to the Model One/10, you cannot use the Model One/10 without enabling the printer. If the attached printer is not enabled (READY), the system will hang whenever you power up, COLDstart, WARMstart, or try to print something.

11.5.1 Using the Printer

Use the HDCOPY command to print out the image memory. With HDCOPY you can indicate the printing method you want to use. Since there are only seven colors available for the printer, each printing method has its advantages and disadvantages.

11.5.2 Printing Text

You can print text in two different sizes. Use DITHER1 and PATTERN1 for smaller size text, and DITHER2 and PATTERN2 for text which is twice as large. The double size text prints out sideways.

The command ALPHEM PRNTON causes any text on the screen to be output to the printer. To turn off this mode, use the command ALPHEM PRNTOFF.

11.6 Using the DIAG Command

The DIAG command can be used to run diagnostic tests. There are four diagnostic tests available.

Test	Description
DIAG 0	Displays a 16 x 16 monitor adjustment grid.
DIAG 1	Displays a 256-color grid, going from 0 in the lower left corner to 255 in the upper right corner.
DIAG 2	Displays a 1024-color grid, going from 0 in the lower left corner to 1023 in the upper right corner.
DIAG 3	Increments pixel values starting at the bottom of the screen and moving up. This test runs until a key is pressed.

Failure of any of the DIAG tests indicates a firmware or hardware problem. Contact Raster Technologies main office or your local representative.

The address and phone number of the Raster Technologies main office is

Raster Technologies, Inc.
9 Executive Park Drive
North Billerica, MA 01862

(617) 667-8900

Raster Technologies has local offices in

- Houston, Texas	(713) 460-4741
- Orange, Califiornia	(714) 835-3791
- Sunnyvale, California	(408) 720-0440

In addition to using the DIAG command for testing, you may want to use the DIAG command to align the monitor or to check on the current look-up table.

12 Installing PROMs

These pages describe how to install new Model One/10 firmware in an existing Model One/10. This procedure involves replacing the PROMS on one of the printed circuit boards.

Note: New Model One/10s are shipped with the PROMs already in place.

IMPORTANT: Upgrading the Model One/10 firmware should only be done under the supervision of Raster Technologies or your local representative. If PROMs are installed incorrrectly, they will be destroyed when the unit is powered on, and the Model One/10 itself may be damaged.

Be sure to read suggestions and notes associated with each step of the procedures before performing that step!

12.1 Variations in Model One/10s

There is one variation in different versions of the Model One/10 that affects how you replace the PROMs: earlier Model One/10s do not have a metal plate to hold the circuit boards in place, while more recent Model One/10s do. This document assumes such a metal plate is in place. If you have an earlier Model One/10 without this plate, the procedures described in this document can still be followed, ignoring references to the metal plate.

12.2 Warnings

Be sure that you take proper static precautions. Static electricity can cause great damage to the boards.

Replacing PROMs on the Model One/10 involves disconnecting and reconnecting several connectors. You must be careful throughout the procedure to make sure that pins are not accidently bent or damaged.

When disconnecting cables, you should generally pull firmly on the connectors, but pull straight (don't wiggle the connector back and forth very much). Be careful not to push against exposed pins.

12.3 Basic Tasks

The basic tasks involved in installing new Model One/10 firmware are

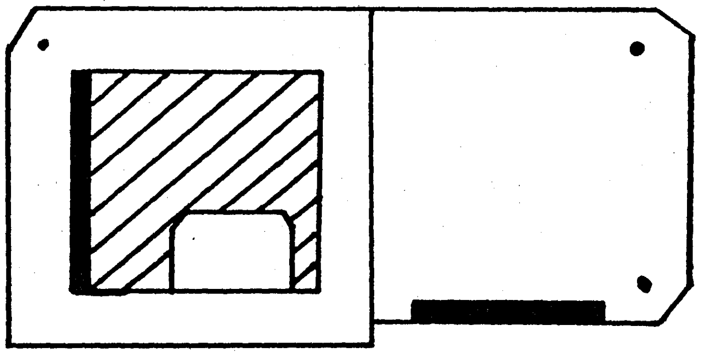
- removing the printed circuit board containing the PROMs
- replacing the PROMs, and
- inserting the board.

12.3.1 Removing the Board

Removing the printed circuit board with the PROMs involves disconnecting cables and pulling the board with the PROMs out of the Model One/10.

Before you begin: if your Model One/10 has a metal stand, make sure that the metal stand is collapsed, so that the unit lays flat. This makes it easier to remove the boards.

Step	Action
1	Make sure that the Model One/10's power is turned off. Unplug the black power cable on the back of the Model One/10.
	Note: You do not have to unplug the keyboard. If you do, be very careful that when you plug it back in, you plug it into the port labeled "KEYBOARD" (the port on the left); the trackball port looks very similar.
2	Remove the back plastic panel of the Model One/10. See Figure 11.1.
	This involves unscrewing the 6 Phillips head screws (4 are inset at each corner, and the other two screws are near the bottom of the back panel between the left corner and the middle). Then pull the back panel off, toward you.
	Suggestion: A magnitized Phillips head screwdriver makes the process easier, by removing the screws from the inset; otherwise the screws may remain in the inset and may fall out when you pull the back panel off.
	Result: You now can see the printed circuit boards (on the left, in a vertical position), the metal plate that holds the boards in place, and several cables. (See diagram below.)
	Diagram: The following is a diagram of the metal plate (the cross-hatched portion represents the opening in the plate).



Step Action

3 Disconnect the 2 cables that go through the cut-out in the metal plate (see above diagram):

- Disconnect the cable on the right (as you face the back of the Model One/10), which is black and has 5 wires. Pull the plastic flap that is part of the pin unit away from the connector, to allow you to disconnect the cable. Pull the connector firmly.

- Disconnect the next cable to the left, which is a black cable with 3 wires. Disconnect it the same way you did the first cable.

4 Remove the two green rectangular connectors that join the two boards, by pulling each connector toward you.

Note: On older Model One/10s these connectors are flat cables; on more recent Model One/10s these connectors are printed circuit boards.

5 Remove the metal plate that holds the boards in place. (See above diagram).

Unscrew the 3 Phillips head screws, pull the metal plate toward you, and slide the plate down over the attached black cable.

Caution: You may have to maneuver the plate around the black cable that attaches in the top right-hand corner of the Model One/10; be careful not to push against the connector too hard.

6 Disconnect the cable that attaches in the upper left-hand corner of the Model One/10. This cable has a 1 1/4" connector, with many wires feeding into the connector. Push the connector toward the left wall of the Model One/10.

7 Slide the metal plate over the cable that you just disconnected (Step 6), so that the plate is completely removed.

8 Slide the board on the left out toward you about 1 1/2", to access the remaining cables that you need to disconnect.

9 Disconnect the remaining 6 cables that are attached to the left-hand board:

- Disconnect the black cable with 4 wires (in the bottom left-hand corner). Pull the plastic flap that is part of the pin unit away from the connector, to allow you to disconnect the cable. Pull firmly, holding the board to provide resistance.

- Disconnect the flat cable that goes in back of the other cables in the left-hand corner. Pull the connector toward the left wall of the Model One/10 (there is no plastic flap to release).

Step	Action
9 (cont)	- Disconnect the white connector with 3 wires (above the 3 brass connectors). You need to pull quite firmly. - Disconnect the 3 (R, G, B) brass connectors, one at a time. <u>IMPORTANT</u>: Older Model One/10s may not have labels on each brass connector. If not, you should label them as you disconnect them; label the top connector "R" (for red), the middle connector "G" (for green), and the bottom one "B" (for blue).
10	Remove the left board by pulling it out toward you. <u>Notes</u> - Make sure all the cables are clear of the board, including the black cable that is attached in the top left-hand corner. - You may want to grasp the board by the shiny metal component on the left side of the board, near the bottom, and by the right side of the board, which is smooth. - Be careful not to bend any of the exposed pins.

12.3.2 Replacing the PROMs

Once you have disconnected the cables attached to the left board and removed the board, you are ready to replace the old PROMs with the new PROMs.

Once the board is removed, you may want to place it down on a clean, flat, non-static area. Make sure there is nothing in the work area that might bend any of the exposed pins.

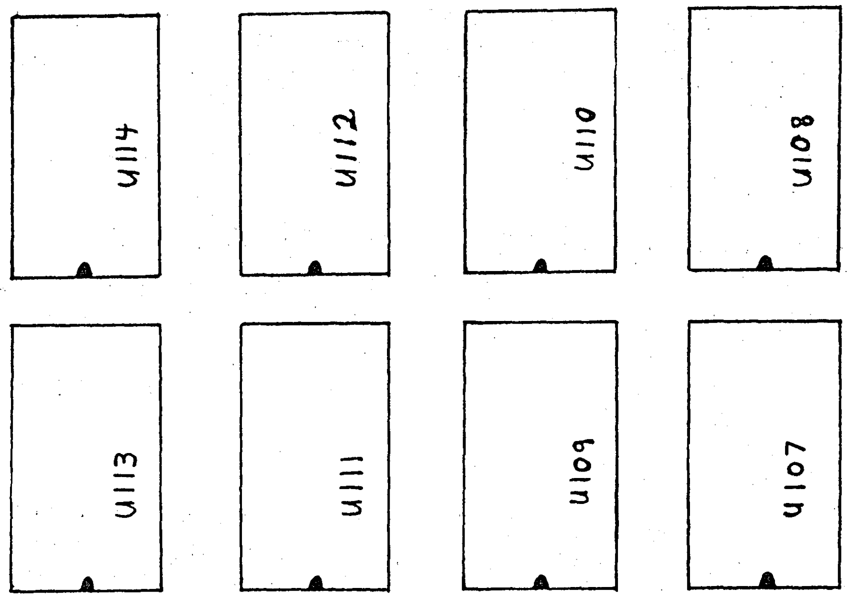
Step	Action
1	Place the board so that the PROMs are at the top of the board as you face it.

Step	Action
2	Remove the PROMs from their sockets, using a chip puller or a small screwdriver. If you use a screwdriver, insert the screw-driver between each PROM and the blue socket, and pry each PROM up carefully.

The diagram below shows the location of the 8 PROMs on the board.

IMPORTANT: Place the PROMs where they will not be damaged (their legs bent or stressed), or subject to static electricity.

Diagram: This diagram assumes you have the board aligned as indicated in Step 1. The first number on each PROM (e.g., U107) will be the same on the old and new PROMs; the second number on each PROM indicates the version of the firmware.



Step	Action
3	Install the new PROMs carefully into the correct sockets. Line up the pins with the sockets carefully before pushing in each PROM. Match the location numbers on the PROMs to the location numbers shown in the diagram above. Make sure all 8 PROMs you are installing are of the same version and revision level. IMPORTANT: - Be extremely careful to orient the PROMs with the notch facing in the direction indicated in the diagram. All notches on the PROMs must align with the notches on the sockets. - Be careful not to bend the PROM itself, its legs, or the board when you insert the PROMs.
4	Verify that the PROMs are inserted correctly: - Are the notches oriented corrrectly? - Do the location numbers correspond correctly? - Are all 8 PROMs of the same version and revision level? - Are all pins in the sockets?

You are now ready to insert the board and reconnect the power.

12.3.3 Inserting the Board and Reconnecting the Power

The procedure for inserting the board is essentially the reverse of the procedure you used to remove the board.

When reconnecting cables, make sure the connectors are firmly in place.

Step	Action
1	Insert the board in the 1/8" tracks at the top and bottom of the board housing, with the shiny metal component on the board toward the left wall of the Model One/10. Leave the board out by about 1 1/2" to allow access for reconnecting cables. Suggestions: - Line up the board with the top track first, then the bottom track. - Push the board in by gripping just below the white pin unit near the top of the board.

Step	Action
2	Reconnect the 6 cables that attach to the board you inserted:

- Reconnect the R, G, B brass connectors:
  - the connector marked "R" goes on the top pin, which is also labelled "R"
  - the connector marked "G" goes on the middle pin, which is also labelled "G"
  - the connector marked "B" goes on the bottom pin, which is also labelled "B".
- Reconnect the flat cable that goes in back of the brass R, G, B connectors, fitting the connector in the plastic guides at the top and bottom of the pin unit.
- Reconnect the cable that attaches above the R, G, B connectors. The connector snaps into place.
- Reconnect the black cable (with 4 wires) on the pins at the bottom of the board. Pull the plastic flap on the pin unit open, and insert the connector with the plastic ridge facing the plastic flap on the pin unit.

  Notes:

  - The connector has 4 holes; be sure the single hole is at the top, and the 3 other holes at the bottom.
  - You may need to hold the board to provide the necessary resistance for inserting the connector.

3 Push the left board in all the way.

4 Secure the metal plate in place with the 3 screws. Make sure that the black cable that attaches to the top left corner of the Model One/10 is clear of the metal plate.

5 Reconnect the flat cable to the top left pins facing the left wall of the Model One/10. The connector should be oriented so that the wires are entering from the back side of the Model One/10.

Step	Action
6	Reconnect the 2 cables that fit through the cut-out of the metal plate: - The cable with the 5 wires is inserted on the right-hand side. Pull the plastic flap on the pin unit open, and insert the connector with the plastic ridge facing the plastic flap on the pin unit. The connector has 3 holes; be sure the single hole is at the top, and the 2 other holes at the bottom. - The cable with 3 wires is inserted on the left-hand side. Use the same procedure as for the other cable (see above).
7	Reconnect the 2 green rectangular connectors. The slightly longer one goes at the top, connecting the 2 boards. The shorter one goes below the other green connector. You may want to push them in by pressing on the middle part of the connectors. You may need to hold the boards slightly apart to insert the connectors. IMPORTANT: There are arrows on each of these connectors indicating which side should face up.
8	Plug in the power cable. Turn on the power switch on the front of the Model One/10. If you do not hear a "beep" or see the start-up message ("Model One/10, Revision n.n"), check that all the cables are connected. If the all the cable connections are correct, check the location and seating of the PROMs. If either is incorrect, the PROMs have been destroyed; you will have to obtain a new set of PROMs. IMPORTANT: If you disconnected the keyboard plug, be sure that you plug it in the port marked "KEYBOARD" (the port on the left); the trackball port looks very similar.
9	Replace the back panel of the Model One/10, screwing in the 6 Phillips head screws. Suggestion: Line up the top of the back panel first, then fit the rest of the panel onto the back of the Model One/10 (the back panel fits quite snuggly; you may have to try lining it up a few times to get the correct alignment). You may have to tap it into place.

13. Selecting Terminal Features Using Setup Mode

The Model One/10 has a wide variety of terminal features. A few of these features are:

- o Display format (number of rows across and number of lines down displayed)
- o Tab stops
- o Autowrap
- o Host data bits
- o Host baud
- o Text color (r,g,b values)

When you first install it, the Model One/10 is programmed so that all terminal features have defaults which are stored in NVRAM. However, each feature is changeable, that is, user-selectable, at any time.

You select and store these terminal features in a dedicated mode called Setup mode. Setup mode is divided into two parts: Setup A and Setup B. Many of these features can also be selected with control sequences (see Section 15) or with the SYSCFG command.

13.1 Entering Setup Mode

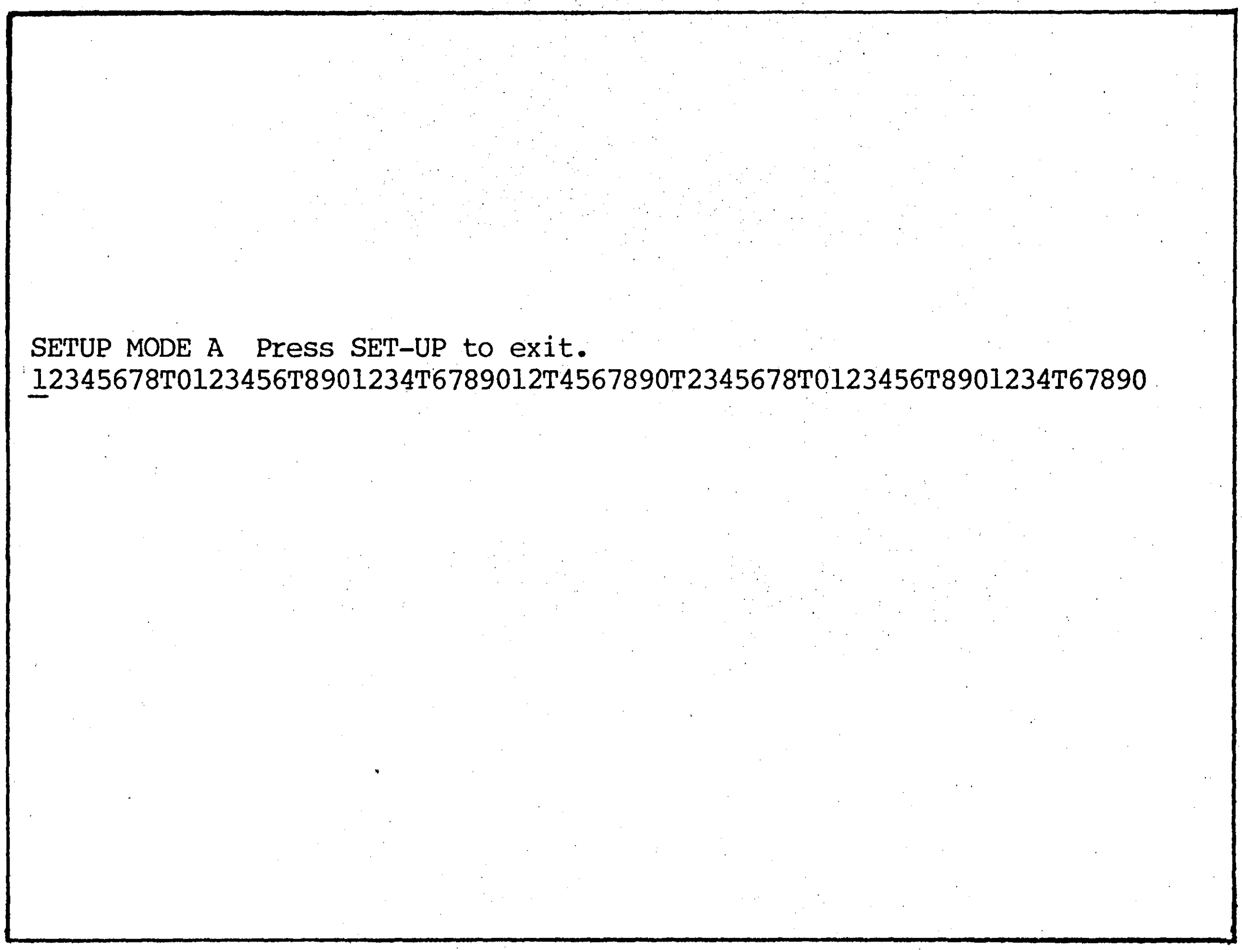
To enter Setup mode, press $\boxed{\text{SET-UP}}$ at the top corner of the Model One's keyboard. Pressing this key puts you in Setup A, the default. Once you are in Setup A, pressing $\boxed{\overset{\%}{5}}$ moves you into Setup B. Likewise, toggling $\boxed{\overset{\%}{5}}$ moves you back and forth between the two setup modes.

Note: You should not enter Setup mode during a graphics application.

13.2 Viewing the Status of Setup Features

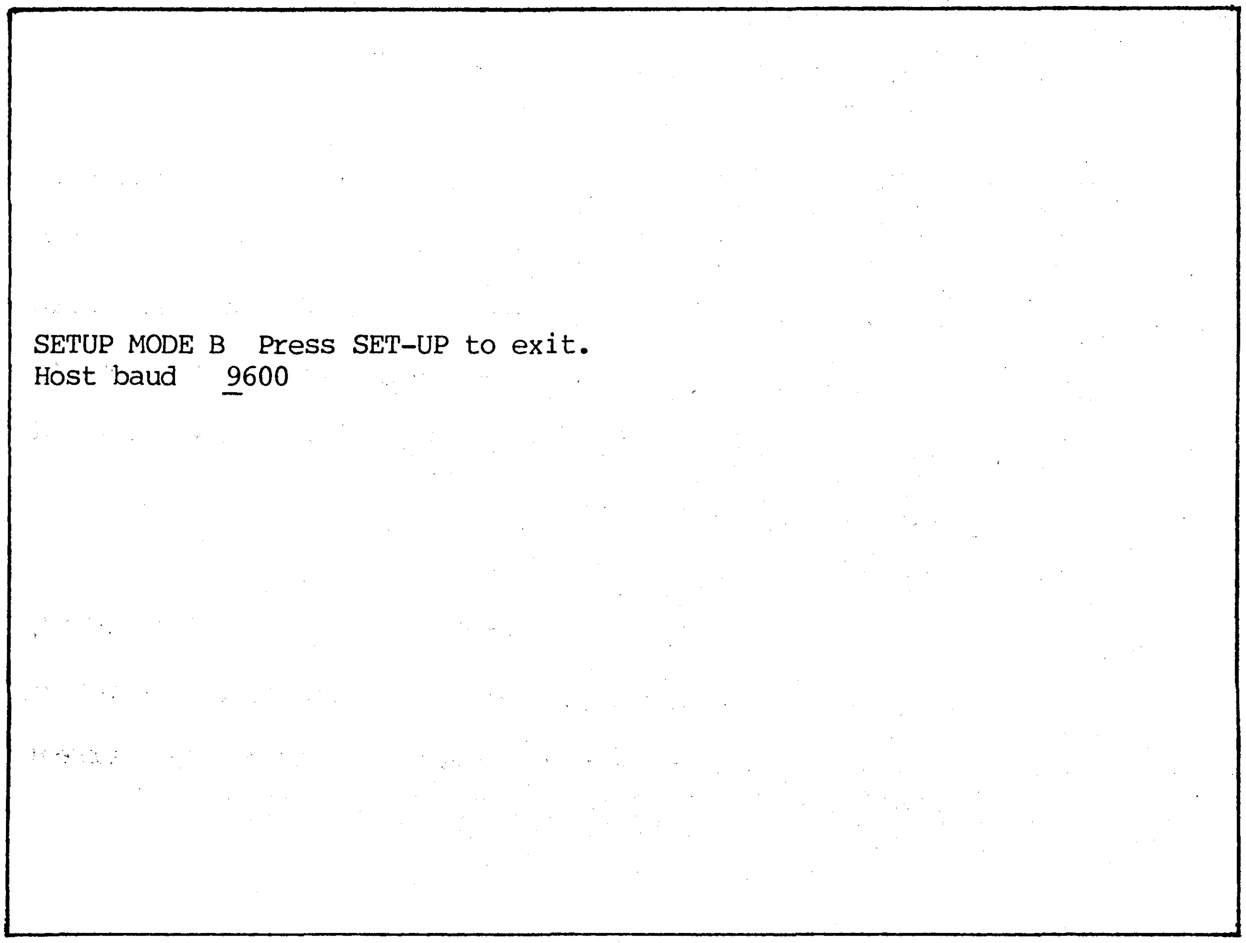
As soon as you enter either of the setup modes, you see a two-line setup display starting at the current line. The cursor is repositioned on the second display line. The setup display overlays the current screen of text and/or graphics which remains visible. An exception to this is a filled-up screen; in this case, the top two lines of text scroll up to allow room for the setup display. When you leave Setup mode, the cursor automatically returns to its previous location.

The Setup A display shows a long line of numbers (see Figure 13.1). These numbers represent columns for setting tab stops. The Setup B display shows one selectable feature at a time, along with the current value (see Figure 13.2). Both setup displays initially show the current values, either the default for each feature or the last value you selected.



```
SETUP MODE A  Press SET-UP to exit.
12345678T0123456T8901234T6789012T4567890T2345678T0123456T8901234T67890
```

Figure 13.1 Setup A Display



```
SETUP MODE B  Press SET-UP to exit.
Host baud   9600
```

Figure 13.2 Initial Setup B Display

13.3 Selecting Setup Features

Select the setup features you want by entering the correct setup mode. Use Setup A to set tabs and adjust screen brightness. Press $\boxed{\uparrow}$ to increase brightness and $\boxed{\downarrow}$ to decrease it. To set and change tabs, use $\boxed{\rightarrow}$ and $\boxed{\leftarrow}$, along with $\boxed{\overset{@}{2}}$ or $\boxed{\overset{\#}{3}}$ (see Section 13.5.1 for further directions on setting tab stops).

Select Setup Mode B features, again, with $\boxed{\uparrow}$, $\boxed{\downarrow}$, $\boxed{\rightarrow}$, and $\boxed{\leftarrow}$. Since Setup B displays only one feature at a time, use $\boxed{\uparrow}$ to display the next feature and $\boxed{\downarrow}$ to display the previous one.

When the feature you want to select is displayed, specify a value by successively pressing $\boxed{\rightarrow}$ and $\boxed{\leftarrow}$ to step through and display all the available choices even if there are only two, such as disabled/enabled. When the value you want is displayed, stop pressing the key and either move on to another feature or exit Setup Mode (see Section 13.6 for a complete list of terminal features you can select using Setup B).

If you want to save your current configuration in the Model One/10's NVRAM, press $\boxed{\text{Shift}}$ - $\boxed{\text{S}}$ after making any changes to the configuration in Setup modes A or B. You can also use the SAVCFG command to save the current configuration in NVRAM.

Note: The Display Control Enabled feature is not saved with a $\boxed{\text{Shift}}$ - $\boxed{\text{S}}$. You should use SAVCFG.

If you want to reset your terminal completely, press $\boxed{\text{SHIFT}}$ - $\boxed{\text{0}}$ ("zero"). This is equivalent to a Model One/10 COLDSTART. Additionally, $\boxed{\text{CTRL}}$ - $\boxed{\text{SHIFT}}$ - $\boxed{\text{PF4}}$ resets all setup values to defaults (equivalent to pressing a hardware button on other Model Ones).

13.4 Exiting Setup Mode

Any feature you select remains in effect until you again enter Setup mode and change it. When you are finished selecting terminal features, exit Setup mode by pressing $\boxed{\text{SET-UP}}$.

It is important to remember that selections are not permanently saved in NVRAM until you press $\boxed{\text{Shift}}$ - $\boxed{\text{S}}$ after making any changes to the configuration in Setup modes A or B, or use SAVCFG.

13.5 Setup Mode A Features Defined

This section describes each Setup Mode A feature.

13.5.1 Tabs

The Model One/10 terminal supports regular tab stops. The default tabs are at every eight columns, starting with Column 9. To set and clear tab stops, enter Setup A. Then move the cursor with $\boxed{\leftarrow}$, $\boxed{\rightarrow}$, $\boxed{\text{TAB}}$, or $\boxed{\text{RETURN}}$ until it is under a column at which you want to set or clear a tab stop. Next, press $\boxed{\overset{@}{2}}$. The location of each tab stop is indicated by a "T" in each column at which you set a tab (see Figure 13.1).

If you want to clear all tabs, press $\boxed{\overset{\#}{3}}$.

You can also set and clear tabs with the host computer (see Section 15.9).

13.5.2 Screen Brightness

Screen brightness can be adjusted higher or lower. To increase the brightness, press $\boxed{\uparrow}$. To decrease the brightness, press $\boxed{\downarrow}$. All color components of normal text are changed by the same increment even if the current RGB values are different.

13.6 Setup B Features Defined

This section describes each Setup Mode B feature. These features are listed below with the available values you can select. The first value shown for each feature is the default value.

o Host baud	9600/19200/300/600/1200/2400/4800
o IEEE mode	Normal/Listen/Talk
o Host parity type	Even/Odd
o Host parity	Disabled/Enabled
o Host data bits	8/7
o Overstrike	Disabled/Enabled
o Tablet type	GTCO/old GTCO/new Bitpad
o Host graphics mode	Binary/ASCII-hex
o XON/XOFF	Disabled/Enabled
o Newline	Disabled/Enabled
o Autowrap	Disabled/Enabled
o Cursor type	Underscore/Block
o Reverse screen	Disabled/Enabled
o Display control codes	Disabled/Enabled
o Display format	24x80 24x128 30x40 30x80 30x128 60x80 60x128 15x40 15x80 24x40
o Solid background	Disabled/Enabled
o Text blinking	Enabled/Disabled
o Background color	R=000 G=000 B=000 +
o Blinking text color	R=000 G=000 B=000 +
o Normal text color	R=000 G=000 B=000 +

13.6.1 Host Baud

Use this feature to select one of 7 available baud rates:

- 300
- 600
- 1200
- 2400
- 4800
- 9600 (default)
- 19200

13.6.2 IEEE Mode

This mode appears on the Model One/10 list of Setup B options, but is not supported.

13.6.3 Host Parity Type

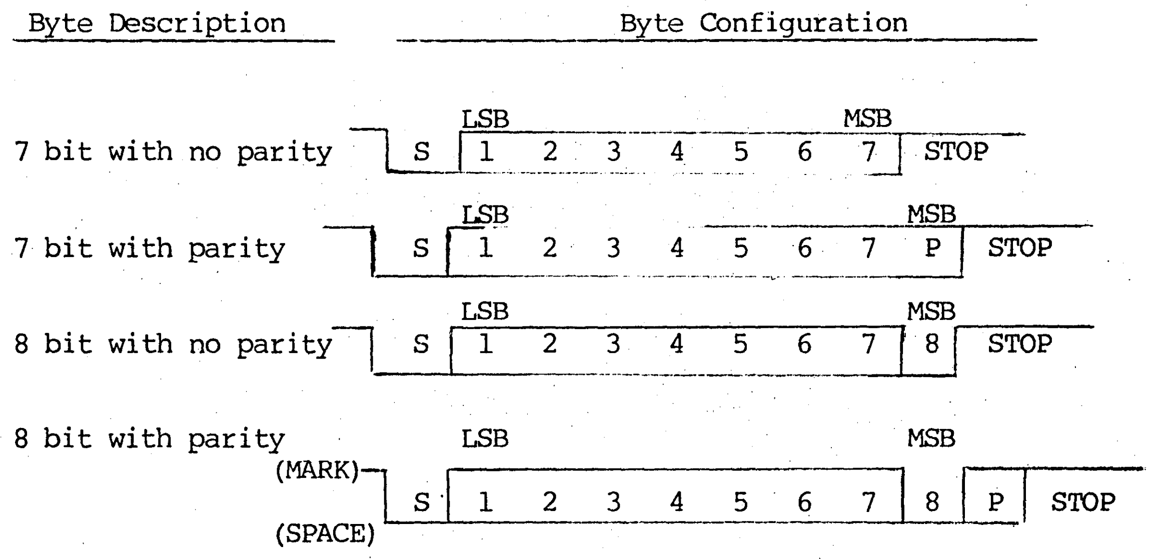
This feature selects the type of parity. You can specify even (default) or odd parity. If you have disabled the Host Parity option, Host Parity Type is disregarded.

13.6.4 Host Parity

The Host Parity feature checks for errors in host transmitted data. When you activate this feature, the Model One/10 checks received data for current parity, odd or even, and generates parity on transmitted data. If the Model One/10 receives a character with incorrect parity, a parity error message is displayed on the screen in place of the erroneous character.

13.6.5 Host Data Bits

This feature determines whether each transmitted character contains eight (default) or seven data bits. When you select 8-bit characters, Bit #8 (L.S.B.) is always set to space (0) for normal ASCII text (see Figure 13.3 for possible byte configurations).



Key

S = Start bit, always = 0
LSB = Least significant data bit
MSB = Most significant data bit
Stop = Stop bit(s), 2 at 110 baud or less, 1 at more than 110 baud
P = Parity bit, odd or even

Figure 13.3 Model One/10 Byte Configurations

Note that with graphics commands, if you are sending 7-bit characters, you must send this data in either ASCII or ASCII-"hex." However, the Model One/10 does accept data in three possible forms: 8-bit binary; ASCII-hex in which 8-bit code is represented by two ASCII characters, one character for each nibble of the 8-bit value; and ASCII data, in ASCII mode. ASCII-hex data has twice as many bytes as an 8-bit code normally has. For example, send:

E7H as:

'E' (45H) 8 bits

'7' (37H) 8 bits

You can send graphics data in the form of Model One/10 command mnemonics, MOVABS, for instance, by enabling ASCII mode. If you are in ASCII-hex mode, enable ASCII mode with this sequence:

CTRL-D (04H)

'9' (39H)

'B' (42H)

'0' (30H)

1' (31H)

When you are in binary mode and you want to switch to ASCII mode, send this sequence:

CTRL-D (04H)

9BH

01H

13.6.6 Overstrike

With Overstrike disabled (default), each character cell is erased before a character is drawn. With Overstrike on, the contents of the character cell is not erased prior to drawing a new character. Overstrike allows characters to be drawn combined.

13.6.7 Tablet Type

Select the tablet type you are using: GTCO/old (default), GTCO/new, or Bitpad.

13.6.8 Host Graphics Mode

Select the way you want to talk to the host when you are in Graphics mode: binary (default) or ASCII-hex.

13.6.9 XON/XOFF

This feature controls the generation of the XOFF (default, DC3) and XON (default, DC1) synchronization codes. These codes ensure that data from the host computer is not lost. With Auto XON/XOFF enabled, the XOFF code is transmitted under any of these conditions:

- o The receive FIFO buffer is almost full.
- o You press CTRL-S.

The XON code is automatically generated to resume host transmission under any of these conditions:

- o The receive FIFO buffer is almost empty and an XOFF code was sent.
- o You press CTRL-Q.

The XON and XOFF codes are Model One/10 special characters and can be changed with the SPCHAR command.

13.6.10 Newline

The Newline feature controls line feeds. With Newline disabled (default) pressing RETURN generates only the CR (Carriage Return) code. With this feature enabled, pressing RETURN generates both the CR and the LF (Line Feed) codes.

You can also disable/enable this feature with the host with escape sequences (see Sections 15.2.9.1 and 15.2.9.2).

13.6.11 Autowrap

Autowrap determines whether or not, as you type, the cursor automatically advances to the first position of the next line immediately after you enter a character into the last position of the current line.

With Autowrap disabled (default), when you get to the end of a line as you are typing, the cursor remains in the last column and successive characters that you enter overlay the last character on the line. With Autowrap on, the cursor and the newly entered text automatically wrap around to the beginning of the next line.

You can also select this feature with the host (see Sections 15.2.9.11 and 15.2.9.12).

13.6.12 Cursor Type

There are two cursor types, a blinking underscore (default) and a blinking block.

13.6.13 Reverse Screen

Enabled, this feature allows you to reverse the text color with the background color, whether these colors are the defaults, white and black, or whether you specified other colors either in Setup B (see Sections 13.6.19 and 13.6.21) or with a control sequence (see Section 15). Disabled is the default.

13.6.14 Control Codes Displayed

With this feature enabled, "hex" codes are displayed when you type control codes or escape sequences. For example, if you type:

```
ABC  CTRL-A   DEF
```

you see displayed:

```
ABC<01>DEF
```

13.6.15 Display Format

If you do not want the default, 24 x 80, select another display format by specifying the number of rows across and the number of lines down. These formats are available:

Rows and Columns

```
15 x 40
15 x 80
24 x 40
24 x 80
24 x 128
30 x 40
30 x 80
30 x 128
60 x 80
60 x 128
```

Note that some character distortion occurs for screen formats with 128 columns and/or 60 rows due to the smaller size characters required.

13.6.16 Solid Background

With this feature disabled (default), there is a graphics background. Otherwise, the background is solid. This option controls the display of bit planes 0-7 while you are in text mode.

13.6.17 Text Blinking

Text displayed on the Model One/10 can either remain steady or blink continuously once per second. Blinking can be disabled, causing different color text to be displayed but not to blink. The default is Text Blinking enabled.

13.6.18 Background Color

The screen's default background is Color Number 0 but it can be any color you select from the Model One/10's available colors. Use this feature to select a background shade. The default values, displayed when you initially select the Background Color feature in Setup B (see Figure 13.4), are R=000, G=000, B=000--in effect, a black background.

To change these values, press $\boxed{\text{->}}$ and $\boxed{\text{<-}}$ to toggle back and forth between the "+" and "-". After you have selected "+" to increase a value or "-" to decrease a value, press $\boxed{R}$, $\boxed{G}$, or $\boxed{B}$ to change the red, green, blue color components respectively. Note that these changes are automatically in increments of eight, all relative to the current color values.

```
SETUP MODE B  Press SET-UP to exit.
Background color   R=000 G=000 B=000   +
```

Figure 13.4 Setup Mode B Display: Background Color Defaults

13.6.19 Blinking Text Color

Use this feature if you have enabled blinking text and you want to specify a text color other than black.

Blinking text can be any color you select from the Model One/10's available color values. The default values, displayed when you initially select the Blinking Text Color feature in Setup B (see Figure 13.5), are R=000, G=000, B=000.

To change these values, press $\boxed{\text{->}}$ and $\boxed{\text{<-}}$ to toggle back and forth between the "+" and "-". After you have selected "+" to increase a value or "-" to decrease a value, press $\boxed{R}$, $\boxed{G}$, or $\boxed{B}$ to change the red, green, blue color components respectively. Note that these changes are automatically in increments of eight, all relative to the current color values.

```
SETUP MODE B  Press SET-UP to exit.
Blinking text color   R=000 G=000 B=000   +
```

Figure 13.5 Setup Mode B Display: Blinking Text Color Defaults

13.6.20 Normal Text Color

Normal text (non-blinking) can also be any color you select from the Model One/10's available color values. The method for changing these values is the same as with blinking text (see the previous section, 13.6.20, and the following figure, Figure 13.6).

```
SETUP MODE B  Press SET-UP to exit.
Normal text color   R=224 G=224 B=224   +
```

Figure 13.6 Setup Mode B Display: Normal Text Color Defaults

13.7 Changing Model One/10 Special Characters

Model One/10 firmware has default special charaters which you can change. Use the DISCFG command to display them and the SPCHAR command to change them.

14. Using Keyboard Controls and LED Indicators

The Model One/10's keyboard has 13 dark gray control keys, a set of 12 programmable function keys, a set of keypad keys, and seven LED indicators. Some of these keys are:

- $\boxed{\text{SET-UP}}$
- $\boxed{\text{ESC}}$
- $\boxed{\text{CTRL}}$
- $\boxed{\text{TAB}}$
- $\boxed{\text{PF1}}$

This section discusses the keyboard's control keys, function keys, and LED indicators. The effects of some of these keys vary according to the set-up features (see Section 13) you selected. Note that the following control key descriptions assume that you activated the Local Echo feature and that transmitted codes are "echoed" back to the terminal (the transmitted codes have no effect on the terminal unless they are echoed).

The Model One/10's keyboard is illustrated in Figure 14.1.

14.1 Keyboard Controls

14.1.1 $\boxed{\text{SET-UP}}$

Use this key to enter and exit Setup Mode (see Section 13). After you enter this mode, if Auto XON/XOFF is enabled, the terminal sends the XOFF code (DC3) when the input buffer becomes nearly full. The characters in the buffer are then displayed when you exit Setup mode.

While you are in Setup mode and Auto XON/XOFF is disabled, overrun errors can occur when the host continues to send data.

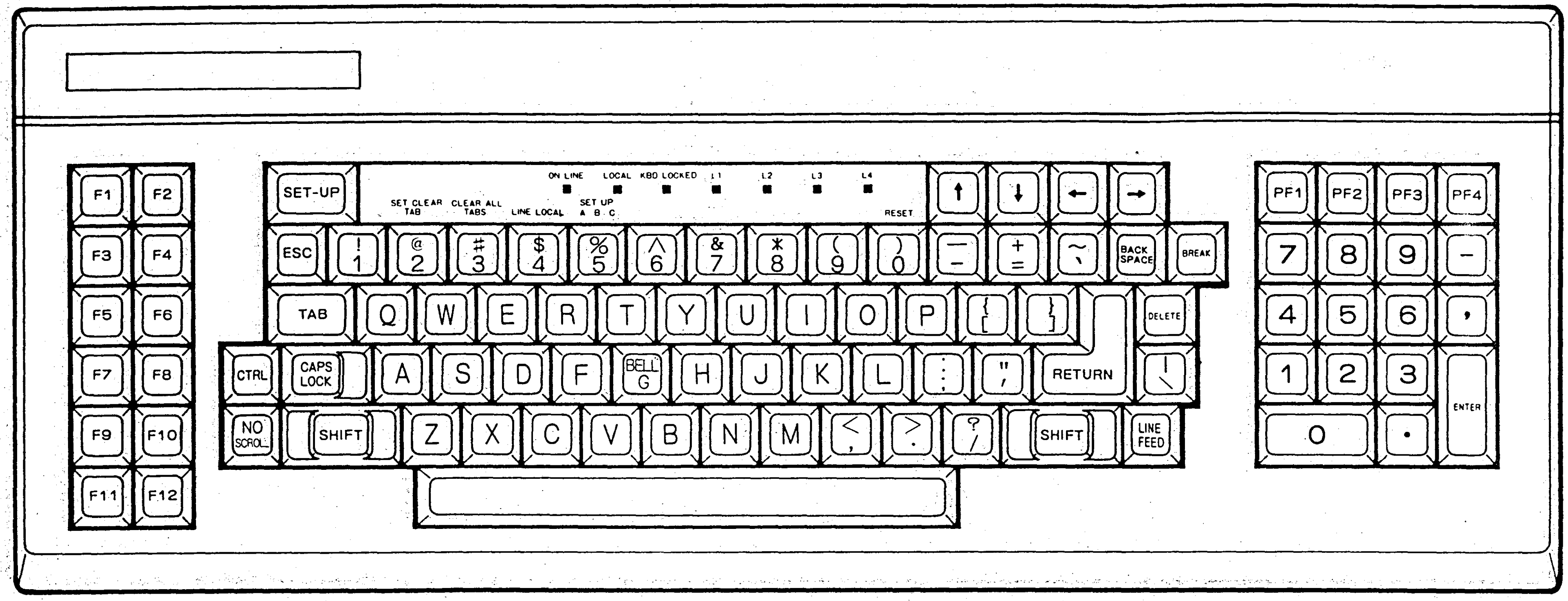


Figure 14.1 The Model One/10 Keyboard

14.1.2 $\boxed{\text{ESC}}$

The ESC code is normally used to initiate multi-code escape sequences (see Section 15). Pressing this key transmits the ESC control code (1BH).

14.1.3 $\boxed{\text{CTRL}}$

This key does nothing by itself. Use it simultaneously with other keys to send control codes to the host.

14.1.4 $\boxed{\text{SHIFT}}$

This key also does nothing by itself. Use it simultaneously with alpha keys to produce the code for upper-case letters and the code for the upper character on the keys which have two characters on them. Two SHIFT keys are on the Model One/10's keyboard, one on either side, for convenience.

14.1.5 $\boxed{\begin{array}{l}\text{CAPS}\\\text{LOCK}\end{array}}$

Pressing this key converts lower case alpha key codes to uppercase before they are transmitted. Note that neither numeric, symbol, nor special keys are affected by CAPS LOCK.

14.1.6 $\boxed{\begin{array}{c}\text{NO}\\\text{SCRL}\end{array}}$

The NOSCRL key (no scroll) alternately transmits XOFF and XON codes; this is equivalent to your pressing CTRL-S and CTRL-Q.

14.1.7 $\boxed{\text{TAB}}$

Pressing the TAB key transmits the HT control code (09H) and moves the cursor to the next tab setting.

14.1.8 BACK SPACE

Pressing this key transmits the BS (backspace) control code (octal 010).

14.1.9 RETURN

This key has two functions. First, it can transmit the CR control code (octal 015) which moves the cursor to the first column of the current line. With the New Line feature enabled (see Section 13.2.11), pressing RETURN transmits both CR and LF control codes, thus moving the cursor to the first column of the next line.

The second function is terminating both host and local Model One/10 command lines.

14.1.10 LINE FEED

Pressing LINEFEED transmits the LF control code (0AH).

14.1.11 BREAK

Pressing BREAK generates a WARMstart on the controller and initiates 600 millisecond spacing on the data-line.

14.1.12 DELETE

This key transmits the <DEL> code (7FH).

14.1.13 SPACE BAR

This key transmits the space code (20H).

14.1.14 ↑ ↓ → ←

The four cursor movement keys can send multicode sequences that are usually interpreted by an applications program (see Table 14.1). How they function is determined by Cursor Key Mode (see Section 15).

Table 14.1 Codes Transmitted When You Press The Cursor Movement Keys

	Codes Transmitted	
Key	ANSI Mode	ANSI Mode and Cursor Key Mode Enabled
↑	ESC [A	ESC O A
↓	ESC [B	ESC O B
→	ESC [C	ESC O C
←	ESC [D	ESC O D

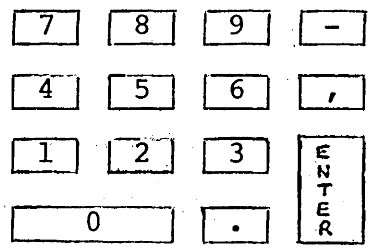
14.1.15 PF1 PF2 PF3 PF4

The four keypad function keys labeled PF1, PF2, PF3, and PF4 send multicode sequences that are usually interpreted by an applications program (see Table 14.2). With these keys, you can access commonly-used functions with only one keystroke.

Table 14.2 Codes Transmitted (With Alternate Keypad Mode Off) When You Press the Function Keys

Key	Code Transmitted
PF1	ESC O P
PF2	ESC O Q
PF3	ESC O R
PF4	ESC O S

14.1.16



The keypad is arranged in adding-machine style, for applications requiring a great deal of numeric input. With Alternate Keypad mode off, pressing numeric key transmits the same numeric code as the corresponding key in the main keyboard section. The ENTER key functions as the RETURN key does on the main keyboard. However, with Alternate Keypad mode enabled, keypad keys are programmed to send escape sequences (see Table 14.3).

Table 14.3 Codes Transmitted (With Alternate Keypad Mode Enabled) When You Press the Keypad Keys

Key	Code Transmitted
$\boxed{0}$	ESC O p
$\boxed{1}$	ESC O q
$\boxed{2}$	ESC O r
$\boxed{3}$	ESC O s
$\boxed{4}$	ESC O t
$\boxed{5}$	ESC O u
$\boxed{6}$	ESC O v
$\boxed{7}$	ESC O w
$\boxed{8}$	ESC O x
$\boxed{9}$	ESC O y
$\boxed{-}$	ESC O m
$\boxed{,}$	ESC O l (lowercase L)
$\boxed{.}$	ESC O n
$\boxed{\text{ENTER}}$	ESC O M

14.1.17 Keys With Special Functions in Setup Mode

When you enter Setup mode, some keys on the main keyboard acquire special functions (see Section 13).

14.2 LED Indicators

The Model One/10 keyboard has seven LED indicators, four that are programmable and three that have only default meanings.

14.2.1 ON LINE

This light indicates that the Model One/10 may send and receive data from the host computer.

14.2.2 LOCAL

This light indicates that the Model One/10 is electrically disconnected from the host computer. Any data entered at the keyboard is looped back through the Model One/10 and displayed on the screen.

14.2.3 KBD LOCKED (Keyboard Locked)

This light indicates that the keyboard is locked and cannot be used to send data to the host.

14.2.4 L1, L2, L3, L4

These four LEDs can be programmed. Note that L1 indicates, "in graphics mode."

14.3 Function Key Programming

The Model One/10's 12 function keys (F1 - F12) can be programmed to execute macros (see Section 3.11.1).

15. Using Control Codes and Escape Sequences

The Model One/10 operates according to the American National Standard Institute's (ANSI) X3.64 programming standard of 1977. This section explains the control codes and escape sequences.

With the Model One/10, you can use control codes and escape sequences in place of typing certain commands. These control codes and escape sequences can also be used in place of selecting some Setup Mode features. This is especially true for the sequences that select modes such as Autowwrap and Screen Format.

15.1 Control Codes

The Model One/10's control codes are codes from Columns 1 and 2 of the ASCII code chart (see Table 15.1) and the DEL code (7FH.) Not all standard control codes affect the Model One/10. Those that you can use are listed in Table 15.2.

Table 15.1 ASCII Code Chart

				BIT 6	0	0	0	0	1	1	1	1
				BIT 5	0	0	1	1	0	0	1	1
				BIT 4	0	1	0	1	0	1	0	1
BIT 3	BIT 2	BIT 1	BIT 0	COL / ROW	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	´	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	¢
1	1	0	1	13	CR	GS	.	=	M	]	m	}
1	1	1	0	14	SO	RS	•	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	—	o	DEL

Table 15.2 Control Code Summary

ASCII Control	Control Character	Hex Equivalent	Action
NUL	^ space	00	Ignored.
BEL	^ G	07	Rings bell.
BS	^ H	08	Backspaces cursor.
HT	^ I	09	Advances cursor to next tab stop.
LF	^ J	0A	Either moves cursor down one line or moves cursor down one line and left to first column.
VT	^ K	0B	Same as LF.
FF	^ L	0C	Same as LF.
DC1 (XON)	^ Q	11	With XON enabled, starts terminal transmission (default SPCHAR).
CR	^ M	0D	Moves cursor to first column.
SO	^ N	0E	Enables G1 character set.
SI	^ O	0F	Enables G0 character set.
DC3 (XOFF)	^ S	13	With XOFF enabled, stops terminal transmission (default SPCHAR).
ESC	^ [	1B	Initiates escape sequences.

15.2 Escape Sequences

This section describes the escape sequences you can use with the Model One/10 terminal. The sequences have the following patterns:

```
<esc> INTRO { {?} { P1 {;...Pn} } FINAL }
```

A description of the elements in these sequences follows.

- <ESC>

  This is the standard ASCII escape code (1BH).

- INTRO

  This is the ANSI introducer code. Some introducer codes terminate the escape sequence, for example <ESC> D and <ESC> C. The introducer code [indicates that more parameters follow.

- { }

  This indicates optional parameter values.

- ?

  This is an ANSI private parameter code used by the Model One/10. It is used to allow additional escape sequences.

- P1...Pn

  These are ASCII-decimal parameters (0-32767).

- ;

  This is the parameter separator you must use with escape sequences that require multiple parameters.

- FINAL

  This is the ANSI final code which specifies the command to be performed.

15.2.1 Cursor Movement Escape Sequences

There are several escape sequences that move the cursor by increments. These sequences fall into two groups, escape sequences that move the cursor out of the current display by scrolling that display and escape sequences that do not move the cursor outside of the defined scrolling region. There is also an escape sequence that moves the cursor to an absolute position.

15.2.1.1 Index

<ESC> D

This sequence moves the cursor down one position. If the cursor is positioned at either the bottom line of the screen or the bottom of the screen scrolling region, the contents of the screen or region will scroll up one line. This sequence is equivalent to pressing the LINEFEED key.

15.2.1.2 Reverse Index

<ESC> M

This sequence moves the cursor up one position. If the cursor is positioned at either the top line of the screen or the top of the screen scrolling region, the contents of the screen or region will scroll down one line.

15.2.1.3 Next Line

<ESC> E

This sequence moves the cursor to the beginning of the next line. If the cursor is positioned at either the bottom line of the screen or the bottom of the screen scrolling region, the contents of the screen or region will scroll up one line.

15.2.1.4 Conditional Cursor Movement Sequences

The following five cursor movement sequences have different effects depending on the defined scrolling region and the current state of the Origin mode.

If you define a sub-screen scrolling region (see Section 15.2.3), you can then use the four incremental cursor positioning sequences to position the curosr anywhere within the scrolling region. However, you cannot position the cursor outside of this region. The Cursor Address sequence is unaffected by the scrolling region, unless the Origin mode escape sequence has been received.

After you define the screen scrolling region, you can make line numbers on the screen either dependent or independent of this region. If, for example, Origin mode is off, line numbers are independent of this region. A Cursor Address sequence that you would use with line parameters of 1;1 positions the cursor to the upper left-hand corner of the screen.

On the other hand, if line numbers are dependent on this defined scrolling region (Origin mode is on), the same Cursor Address sequence would position the cursor at the upper left-hand corner of the scrolling region. The current state of Origin mode affects only the numbering of lines on the screen and thus only the Cursor Address sequence.

15.2.1.4.1 Cursor Up

<ESC> [n A

This sequence moves the cursor up the specified number of rows. Specify this number with n (the default is 1 row).

15.2.1.4.2 Cursor Down

<ESC> [n B

This sequence moves the cursor down the specified number of rows. Specify this number with n (the default is 1 row).

15.2.1.4.3 Cursor to the Right

<ESC> [n C

This sequence moves the cursor to the right the specified number of columns. Specify this number with n (the default is 1 column).

15.2.1.4.4 Cursor to the Left

<ESC> [n D

This sequence moves the cursor to the left the specified number of columnns. Specify this number with n (the default is 1 column).

15.2.1.4.5 Cursor Address

<ESC> [row;col f   or   <ESC> [row;col H

This sequence moves the cursor on an absolute basis. It moves the cursor to the specified row and column. Specify both row and column numbers with decimals, 1-24 for rows and 1-128 for columns (the defaults are 1 and 1). Use either f or H as the termination character.

If you try to position the cursor past the screen boundary, the cursor moves only as far as the boundary. Normally the Model One/10's cursor addressing is with respect to the full screen and not the currently defined scrolling region. If you send the Set Origin Mode escape sequence (<esc> [? 6 h) (see Section 15.2.9.9), however, subsequent cursor addressing will be relative to the top left corner of the scrolling region. When Origin mode is active, the cursor cannot be moved past the limits of the scrolling region.

15.2.2 Tab Escape Sequences

You can set and clear tabs with escape sequences.

15.2.2.1 Set Tab At Current Column

<ESC> H

This sequence sets a tab stop at the current column.

15.2.2.2 Clear Tab At Current Column

<ESC> [g

This sequence clears a tab stop at the current column.

15.2.2.3 Clear All Tabs

<ESC> [3 g

This sequence clears all tabs stops.

15.2.3 COLDstart Escape Sequence

<ESC> c

This sequence COLDstarts the Model One/10. Note that the Model One/10 performs self-testing up to 10 seconds after this command is issued and ignores data coming from the host during this period.

15.2.4 Define Scrolling Region Escape Sequence

<ESC> [row1;row2 r

This sequence sets the top (row1) and bottom (row2) lines of the screen scrolling region. Depending on the selected screen format, the lines on the screen are numbered from 1-15, 1-24, 1-30, or 1-60. Specify both rows with decimal numbers, 1-60 (the default is the entire screen). The smallest scrolling region size you can define is two lines. Redefining the scrolling region does not move the cursor, even if it is outside this region.

15.2.5 Character-erase Escape Sequences

You can erase characters on either a line or screen basis. Erasing does not reposition the cursor.

15.2.5.1 Erase to End of Line

<ESC> [K   or   <esc> [0 K

This sequence erases from the cursor to the end of the current line.

15.2.5.2 Erase to Beginning of Line

<ESC> [1 K

This sequence erases from the cursor to the beginning of the current line.

15.2.5.3 Erase Line

<ESC> [2 K

This sequence erase the entire current line.

15.2.5.4 Erase to End of Screen

<ESC> [J   or   <esc> [0 J

This sequence erases from the cursor to the end of the screen.

15.2.5.5 Erase to Beginning of Screen

<ESC> [1 J

This sequence erases from the cursor to the beginning of the screen.

15.2.5.6 Erase Screen

<ESC> [2 J

This sequence erases the entire screen.

15.2.6 Video Attribute Escape Sequences

You can select any combination of the following video attributes: bold, underlined, blinking, and reverse video.

15.2.6.1 Bold On

<ESC> [1 m

This sequence activates bold video.

15.2.6.2 Underline On

<ESC> [4 m

This sequence activates automatic underlining.

15.2.6.3 Blink On

<ESC> [5 m

With this feature activated, characters are drawn in the blink color (see Section 15.2.10.2). Characters also blink if Blinking has been enabled (see Section 15.2.12.1).

15.2.6.4 Reverse Video On

<ESC> [7 m

For character cells erased to the current text color, this sequence both erases a character cell to the current text color and draws the subsequent character in the background color.

15.2.6.5 Video Attributes Off

<ESC> [m

This sequence deactivates all currently selected video attributes.

15.2.7 Set/Clear LEDs Escape Sequence

<ESC> [parameter q

This sequence turns on/off the four programmable LEDs, L1 through L4. Use the parameters shown in Table 15.3.

Table 15.3 LED Parameters

Parameter	Meaning
0 (default)	Turns off all LEDs.
1	Turns on L1.
2	Turns on L2.
3	Turns on L3.
4	Turns on L4.

Note that L1 and L2 are normally used by the Model One/10 firmware to indicate "Graphics Mode" and "Error," respectively.

15.2.8 Report Sequences

These escape sequences are host generated. Use them to get status reports on the following questions:

15.2.8.1 What are you?

<ESC> [c or <ESC> [Z

Use this sequence to find out the terminal's configuration. The system responds:

<ESC> [1;0 c

which means, "Base VT100."

15.2.8.2 What is the terminal's status?

<ESC> [5 n

Use this sequence to determine if the terminal is communicating properly. The system's response is one of these:

```
<ESC> [ 0 n       if the terminal is O.K.
```

or

```
<ESC> [ 3 n       if the terminal is not O.K.
```

15.2.8.3 What is the alphanumeric cursor position?

```
<ESC> [ 6 n
```

Use this sequence to get the row and column locations of the cursor. The system's response is:

```
<ESC> [ row;column R
```

15.2.8.4 What are the terminal parameters?

```
<ESC> [ 0 x      or      <ESC> 1 x
```

Use either of these sequences to get a report of the current terminal parameters (see Table 15.4). The system's response is:

```
<ESC> [ <report type>; <parity>; <bits per character>; <EOL>;
<transmit speed>; <receive speed>; <baud rate multiplier>; <flags>
```

Table 15.4 Reporting Terminal Parameters

Parameter	Value	Meaning
Report type	3	This message is a report from the terminal and the terminal reports only on request from the host.
Parity	1	No parity
	4	Odd parity
	5	Even parity
Number of bits	1	8 bits per character
	2	7 bits per character
Transmit speed and Receive speed	48	300 baud
	56	600 baud
	64	1200 baud
	88	2400 baud
	104	4800 baud
	112	9600 baud
	120	19200 baud
Baud rate multiplier	1	Baud rate multiplier is 16.
Flags	0-32767	TBD

15.2.9 Mode Escape Sequences

These escape sequences do not usually change the display but, rather, determine the termnal's response to certain commands. Many of the modes described in this section can also be selected in Setup mode. You can thus set these modes either from the host or from the Model One/10 in Setup mode (see Section 13 for features that you can select in Setup mode).

Each mode has two possible selections: set or reset. Table 15.5 lists the modes applicable to the set/reset mode escape sequences and their associated parameters.

Table 15.5 Mode Escape Sequences and Their Parameters

Mode	Parameter
New Line	20
Local Echo	12
Cursor Key	1
Reverse Screen	5
Origin	6
Autowrap	7
Text Blinking	10
Solid Background	11
Overstrike	31

15.2.9.1 Set New Line Mode

<ESC> [20 h

This sequence sets New Line mode. When this mode is set, the RETURN key always generates both the CR and LF codes; received LF codes also generate a CR code.

15.2.9.2 Reset New Line Mode

<ESC> [20 l (lowercase L)

This sequence resets New Line mode. When this mode is reset, the RETURN key generates only the CR code; a received LF code also does not generate a CR.

15.2.9.3 Set Local Echo Mode

<ESC> [12 h

This sequence sets Local Echo mode. When this mode is set, transmitted data is also displayed locally.

15.2.9.4 Reset Local Echo Mode

<ESC> [12 l (lowercase L)

This sequence resets Local Echo Mode. When this mode is reset, there is no local echo.

15.2.9.5 Set Cursor Key Mode

<ESC> [? 1 h

This sequence is valid only when the terminal is in Alternate Keypad mode. With this mode set, the cursor positioning keys generate special escape sequences (see Table 14.1).

14.2.9.6 Reset Cursor Key Mode

<ESC> [? 1 l (lowercase L)

With this mode reset, the cursor movement keys transmit their normal ANSI mode sequences.

15.2.9.7 Set Reverse Screen

<ESC> [? 5 h

This sequence sets Reverse Video mode; the color values you previously gave for text and background are now reversed.

15.2.9.8 Reset Reverse Screen

<ESC> [? 5 l (lowercase L)

With this mode reset, the color values you previously gave for text and background are again displayed as you defined them.

15.2.9.9 Set Origin Mode

<ESC> [? 6 h

This sequence sets Origin Mode; line and column numbers are relative to the selected scrolling region. Cursor Address 1,1 refers to the left-most character on the top line of the scrolling region. The cursor is positioned to the 1;1 position of the physical region.

15.2.9.10 Reset Origin Mode

<ESC> [? 6 l (lowercase L)

With Origin Mode reset, line and column numbers are independent of the selected scrolling region. Cursor position 1,1 addresses the top left-most character of the physical screen. The cursor is positioned to the 1;1 position of the physical screen.

15.2.9.11 Set Autowrap Mode

<ESC> [? 7 h

This sequence sets Autowrap Mode. With this mode set, the cursor automatically advances to the first position of the next line when you enter characters that go beyond the last position of the current line.

15.2.9.12 Reset Autowrap Mode

<ESC> [? 7 l (lowercase L)

With this mode reset, the cursor does not wrap around. Characters you enter that fall beyond the last position overstrike the last character on the line.

15.2.10 Color Escape Sequences (Raster Technologies Private)

Use these exclusive Model One/10 escape sequences to select terminal color values relative to the current values. Many of the values selected with these sequences can also be specified from the Model One/10 in Setup mode (see Section 13).

Table 15.6 lists the text color escape sequences and their associated parameters.

Table 15.6 Text Color Escape Sequences and Their Parameters

Sequence	Parameter
Normal Text Color	0;r;g;b;
Blinking Text Color	2;r;g;b;
Background Color	3;r;g;b;

15.2.10.1 Select Normal Text Color

```
<ESC> [ ?  0;r;g;b m
```

This sequence sets the color for all normal (non-blinking) text. Specify red,green,blue values in the range 0-255. Blinking text can be any color you select from the Model One/10's available color values. The default values are R=000, G=000, B=000.

15.2.10.2 Select Blinking Text Color

```
<ESC> [ ?  2;r;g;b m
```

Use this sequence if you have enabled Text Blinking and you want to specify a text color for blinking text. Specify red,green,blue values in the range 0-255. Blinking text can be any color you select from the Model One/10's available color values. The default values are R=000, G=000, B=000.

15.2.10.3 Select Background Color

```
<ESC> [ ?  3;r;g;b m
```

The screen's background can be any color you select from the Model One/10's available colors. Use this sequence to select a background shade. The default values are R=000, G=000, B=000--in effect, a black background. Specify red, green, blue values in the range 0-255.

15.2.11 Display-screen Formats

The Model One/10 has ten available display-screen formats (see Table 15.7). If you do not want the default format, 24 rows,80 columns, select another format, specifying the number of the format (fmt) you want. Note that some character distortion occurs for screen formats with 128 columns and/or 60 rows due to the smaller size characters required.

Table 15.7 Available Screen Formats

format	Rows,Columns
0	15 x 40
1	15 x 80
2	24 x 40
3	24 x 80 (default)
4	24 x 128
5	30 x 40
6	30 x 80
7	30 x 128
8	60 x 80
9	60 x 128

15.2.11.1 Select Screen Format (Raster Technologies Private)

<ESC> [? fmt ~

This sequence specifies a screen format (see Table 15.7).

15.2.12 Blinking and Normal Text

You can enable and disable blinking text with two escape sequences.

15.2.12.1 Set Text Blinking Mode (Raster Technologies Private)

<ESC> [? 10 h

With this mode set, text displayed with the Blinking Video attribute (see Section 15.2.6.3) blinks.

15.2.12.2 <u>Reset Text Blinking Mode</u> (Raster Technologies Private)

<ESC> [? 10 l (lowercase L)

With Non-blinking Mode set, text displayed with the Blinking Video attribute (see Section 15.2.6.3) does not blink but this text is still displayed in the blink color.

15.2.13 <u>Overstrike Mode</u>

The Model One/10 has a special Overstrike mode. With Overstrike mode enabled, the contents of a character cell is not erased prior to drawing a new character. Overstrike mode allows characters to be drawn more quickly and allows characters to be combined. With Overstrike mode disabled, each character cell is erased before a character is drawn.

15.2.13.1 <u>Set Overstrike Mode</u> (Raster Technologies Private)

<ESC> [? 31 h

This sequence enables Overstrike mode.

15.2.13.2 <u>Reset Overstrike Mode</u> (Raster Technologies Private)

<ESC> [? 31 l (lowercase L)

This sequence disables Overstrike mode.

Appendix A

Model One/10
Command Set

A. Model One/10 Command Set

This appendix describes three categories of Model One/10 commands: commands that are unique to the Model One/10, commands that have been modified for the Model One/10, and standard Model One commands that do not apply to the Model One/10. For a detailed description of the entire firmware set, see the Raster Technologies Model One/10 Command Reference. For a summary of these command descriptions, see the Raster Technologies Model One/10 and Model One/80 Command Reference Card.

A.1 New Commands

The Model One/10 supports the following new commands.

- LUT16 index r,g,b [16H]

  This command corresponds to the LUT8 command but allows access to all 1024 LUT entries.

  An example of LUT16 is:

  ```
  ! * Change the color of overlay plane text to yellow.
  ! LUT16 768 255 255 0
  ```

- IN address [71H]

  Since the Model One/10's I/O is not memory-mapped, input and output instructions are needed to access various hardware registers. This command returns a byte value from the specified Z8001 I/O space address.

  An example if IN is:

  ```
  ! * Read the host serial port's status register.
  ! IN #E081
  FF15
  ```

- OUT address value [72H]

  This command outputs the specified byte value to the specified Z8001 I/O space address.

  An example of OUT is:

  ```
  ! * Send <CTRL-Y> to host serial port
  ! OUT #E080 #19
  ```

- RMSK16 mask [43H]

  This command specified the read mask for all 10 bits of image memory.

  An example of RMSK16 is:

```
! * Suppress display of graphics underneath the
! * alphanumeric text
! RMSK16 #0003
```

- WMSK16 mask [44H]

  This command specifies a write mask for each of the 10 bit planes in the low 10 bits of mask. The high 6 bits of mask are ignored.

  An example of WMSK16 is:

```
! * Write enable the overlay planes, set the current
! * color to write to them, make the lookup table entry
! * for them blue, and flood the overlay planes
! * to blue.
! WMSK16 #0003
! VAL16 #FF03
! LUT16 768 0 0 255
! FLOOD
```

A.2 Non-applicable Commands

The following Model One commands are not applicable to the Model One/10. If you try to execute any of these commands, they will be ignored but will not cause an error.

- LUTRTE

  The Model One/10 cannot reroute LUTs.

- OVRZM

  The Model One/10 cannot zoom.

- PIXCLP

  Pixel functions are not implemented in the Model One/10.

- PIXFUN

  Pixel functions are not implemented in the Model One/10.

- VGWAIT

  The Model One/10's vector generator does not have a FIFO buffer.

- ZOOM

  The Model One/10 cannot pixel-replicate zoom.

- ZOOMIN

  The Model One/10 cannot pixel-replicate zoom.

- Alpha Window Commands

  Alpha window support has been replaced by a new hardware scrolling window which is compatible with the ANSI X3.64 (VT100) specification for terminal protocols. The ALPHEM command remains.

  The new toggle feature lets you change the state of displayed/non-displayed text back and forth. Type ALPHEM TOGGLE.

A.3 Modified Commands

Several standard Model One commands function in a modifed way on the Model One/10. These modifications are described below.

- VAL1K

  This command specifies the use of the full 8 bits (equivalent to VAL8) instead of only the low 6 bits. For values over 63, you need to initialize Look-up table entries 64-255, as required by your application.

- VALUE

  The 8 bits from the red byte are used for the low 8 bits of the current value; the low 2 bits of the green byte are used for the high 2 bits of the current value. The blue byte is ignored.

o LUT8

This command affects only the low 256 entries in the R, G, and B LUTs (they have 1024 entries). The new command LUT16 can modify all other LUT entries.

o LUTA

This command affects only the low 256 entries in the R, G, and B LUTs.

o LUTB

This command affects only the low 256 entries in the blue LUT.

o LUTG

This command affects only the low 256 entries in the green LUT.

o LUTR

This command affects only the low 256 entries in the red LUT.

o LUTRMP

This command affects only the low 256 entries in the R, G, and B LUTs.

o WRMASK

For compatibility, this command functions as it does on the Model One/40. The new command WMSK16 specifies a write mask for all 10 individual bit planes in the Model One/10.

o RDMASK

This command applies to the low 8 bits of image memory. The new command RMSK16 masks all 10 bits of image memory.

o XHAIR

The Model One/10 can display several types of hardware cursors: small plus sign, full-screen crosshair, block cursors for text, and blinking versions (only one crosshair is supported by the firmware). Specify the crosshair type you want with the appropriate crosshair number:

- XHAIR 0 selects a 16 x 16 crosshair.
- XHAIR 1 selects a full-screen crosshair.
- XHAIR 2 selects a blinking 16 x 16 crosshair.
- XHAIR 3 selects a blinking full-screen crosshair.

o PIXEL8

This command writes data to the low 8 bits. The new command PIXL16 allows 10-bit pixel data to be sent to the Model One/10.

o PIXELS

This command transmits, pixel-by-pixel, an image to the Model One/10. The 8 bits from the RED data byte are put into the low 8 bits of image memory; the low 2 bits of the GRN data byte are put into the high 2 bits of image memory. The BLU data byte is ignored.

o WAIT

This command's delay is in 60ths of a second.

o BLINKD

This command applies to the low 256 bytes of the 1024-byte LUT.

o BLINKE

This command applies to the low 256 bytes of the 1024-byte LUT.

o SYSCFG, DISCFG

These Model One commands, although still useable, are virtually replaced with a setup screen (Setup mode), accessible with the Model One/10's keyboard SET-UP key (see Part II, Section 13).

Appendix B

Programming Example: Model One/10 Function Key Definition

B. Programming Example: Model One/10 Function Key Definition

This appendix contains a program to illustrate Model One/10 terminal programming. The sample program, FUNCKEYS.FOR, shows how to define the Model One/10's function keys. This program is a listing of FORTRAN source file which needs to be compiled and linked with the Model One/10 FORTRAN Library.

B.1 Example of Model One/10 Function Key Programming

```
c FUNCKEYS.FOR
c This program defines the Model One/10 function keys.
c
      logical*1 ff
      logical*1 crlf(4), tabs(4), erabuf(10)
c
      data ff / 12 /
      data crlf / 13, 10, 10,10 /
      data erabuf / 27, '[', '2', 'J',
     +              27, '[', '1', ';', '1', 'f' /
      data tabs / 9, 9, 9, 9 /
c
      call RTINIT ('tt',2)
      call ENTGRA
c
c Ensure enough time between characters for any HOSTO commands.
c
      call DELAY (20)
c
c Redefine the Model One/10's WARMSTART and BREAK special characters.
c
      call SPCHAR (2,1,22)
      call SPCHAR (1,1,25)
c
c F1 -- toggle text display (use ALPHEM TOGGLE).
c
      call MACDEF (16)
         call SEND1 (194)
         call SEND1 (2)
         call FLUSH
      call MACEND
      call BUTTBL (16,16)
c
c F2 -- toggle graphics display (Planes 0-7).
c
      call MACDEF (17)
         call RDMASK (0)
         call BUTTTBL (17,28)
         call FLUSH
      call MACEND
      call BUTTBL (17,17)
c
      call MACDEF (28)
         call RDMASK (255)
```

```
           call BUTTBL (17,17)
           call FLUSH
        call MACEND
c
c F3 -- erase the text display.
c
        call MACDEF (18)
           call ALPHAO (10,erabuf (1))
           call FLUSH
        call MACEND
        call BUTTBL (18,18)
c
c F4 -- erase the graphics display.
c
        call MACDEF (19)
        call VMOVE (6,0)
        call VAL8 (0)
        call FLOOD
        call VMOVE (0,6)
           call FLUSH
        call MACEND
        call BUTTBL (19,19)
c
c F5 -- display the fkey menu.
c
        call MACDEF (20)
           call ALPHAO (2, crlf)
           call ALPHAO (26, 'F1 -- TOGGLE TEXT display')
           call ALPHAO (1, tabs)
           call ALPHAO (30, 'F2 -- TOGGLE GRAPHICS display')
           call ALPHAO (2, crlf)
           call ALPHAO (25, 'F3 -- ERASE TEXT display')
           call ALPHAO (1, tabs)
           call ALPHAO (29, 'F4 -- ERASE GRAPHICS display')
           call ALPHAO (24, 'F5 -- display this MENU')
           call ALPHAO (1, tabs)
           call ALPHAO (23, 'F6 -- display LUT ramp')
           call ALPHAO (2, crlf)
           call ALPHAO (6, 'F8 -- ')
           call ALPHAO (2, crlf)
           call ALPHAO (6, 'F9 -- ')
           call ALPHAO (4, tabs)
           call ALPHAO (6, 'F10 -- ')
           call ALPHAO (2, crlf)
           call ALPHAO (6, 'F11 -- ')
           call ALPHAO (4, tabs)
           call ALPHAO (6, 'F12 -- ')
           call ALPHAO (2, crlf)
           call FLUSH
        call MACEND
        call BUTTBL (20,20)
c
c F6 -- LUT ramp display.
c
```

```
      call MACDEF (21)
         call WINDOW (-320,-240,319,271)
         do 100 iy = -240, -240+511, 2
            call VAL8 ( (511-(271-iy)) / 2)
            call MOVABS (311,iy)
            call RECREL (8,1)
 100     continue
         call VAL8 (63)
         call MOVABS (285,-238)
         call TEXT1 (3,'  0')
         call MOVABS (285,-111)
         call TEXT1 (3,' 64')
         call MOVABS (285,17)
         call TEXT1 (3,'128')
         call MOVABS (285,145)
         call TEXT1 (3,'192')
         call MOVABS (285,232)
         call TEXT1 (3,'240')
         call MOVABS (285,264)
         call TEXT1 (3,'255')
         call WINDOW (-320,-240,319,239)
      call MACEND
      call BUTTBL (21,21)
c
c F7 --(unused)
c
c F8 --(unused)
c
c F9 --(unused)
c
c F10 --(unused)
c
c F11 --(unused)
c
c F12 --(unused)
c
c Display the menu and quit.
c
      call EMPTYB
c
      call FLUSH
      call BUTTON (20)
c
      call QUIT
c
      call exit
      end
```

Address counters, 9-1
Addressing range, 6-2
Algorithm, digital-
line generating, 2-7
Alpha and Graphics modes, 3-1
Alphanumeric data, 3-1
ALPHEM, using, 11-5, A-4
Alternate Keypad mode, 14-6,
15-14
Aperture, pick, 4-10, 4-11
Applications, supported by
Model One/10, 2-1
Applications development
features, 2-4
ASCII code chart, 15-2
ASCII data, 13-7, 13-8
ASCII decimal parameter values, 3-1
ASCII-hex,
data, 13-8
format, 3-2
mode, 13-9
ASCII mnemonic names, 3-1
ASCII mode, 12-9
Attributes, 4-1, 4-4
Autowrap, 13-6, 13-10

BACK SPACE key, 14-4
Background,
color, 13-6, 13-12,
15-14, 15-16
graphics, 13-11
solid, 13-11
Basis vectors, 4-20
Baud rate, 2-2, 3-1, 6-2, 11-3
multiplier, reporting, 15-11
Binary,
characters, 3-2
data, 13-8
format, 3-2
mode, 13-9
Bits per character, 15-11
Blink On (<ESC> [5 m), 15-9
Blinking and normal text, 15-17
Blinking text color, 13-6, 13-12,
13-13
Blocks, 4-3
BNC connectors, 8-3
Bold On (<ESC> [1 m), 15-9
BREAK key, 14-4
Buffering, 3-1
Buffers,
FIFO, 13-9, 13-10, A-4
firmware-configurable, 6-2
Button table, 2-5, 3-13, 3-14
loading of, 3-13

CAPS LOCK key, 14-3
Central Processor, 2-5, 2-7, 3-1,
6-1, 7-1, 9-1, 11-2
description of, 6-1
functions of, 6-1
Changing Model One/10 Special
Characters, 13-13
Character-erase Escape Sequences,
15-8
Clear All Tabs (<ESC> [3 g), 15-7
Clear Tab At Current Column
(<ESC> [g), 15-7
Clearance, required, 11-2
Clipping, 3-3, 3-5, 3-6
Clipping windows, 3-6
corners of, 3-6
COLDstart escape sequence
(<ESC> c), 15-7
Color, 2-2
Color escape sequences, 15-15
Columns,
numbers of, 13-11
reporting location of, 15-11
Command interpreter, 3-2
Command stream translator, 2-5, 3-2
Commands,
ALPHEM, A-4
ALPHEM TOGGLE, A-4
ASCII, 3-2, 3-12
BLINKD, A-6
BLINKE, A-6
BUTTBL, 3-13
BUTTON, 3-13
CADD, 3-4
CALL, 5-1
CIRCLE, 3-1
CLEAR, 9-1
CMOVE, 3-4
CORORG, 3-4
CSUB, 3-4
DELPID, 4-4, 4-5
DFTCFG, 3-12
DIAG, 11-5
DISCFG, 3-12, 13-13, A-6
ENTERGRAPHICS, 3-1
EXMODE, 4-9, 4-10, 4-17
FLOOD, 9-1
HDCOPY, 11-5
INSPID, 4-6
LUT8, 8-1, A-5
LUT16, 8-1, A-2
LUTA, 8-1, A-5
LUTB, 8-1, A-5
LUTG, 8-1, A-5
LUTR, 8-1, A-5
LUTRMP, 8-1, A-5

Commands, (cont.)
LUTRTE, A-3
MACDEF, 3-12
MACEND, 3-12
OVRZM, A-3
PATFIL, 3-11
PIXCLP, A-3
PIXEL8, A-6
PIXELS, A-6
PIXFUN, A-4
POLYGN, 3-6
POP, 4-23
PRMFIL, 3-6, 3-9
PUSH, 4-23
QUIT, 3-1, 3-2, 3-12
RDMASK, 8-1, A-5
RDMODE, 4-5
RDPICK, 4-13 to 4-15
RDPID, 4-15
RDREG, 4-5
RDXFORM, 4-5
REPLAY, 3-2
RMSK16, 8-1, A-3
SAVCFG, 3-12, 13-4, 13-5
SCRORG, 9-1
SEGDEF, 4-6
SEGEND, 4-6
SEGINI, 4-3, 4-6
SEGINQ, 4-5, 4-6
SEGREF, 4-2, 4-6, 4-7
SETATR, 4-4
SETGL, 4-5, 4-12, 4-17
SETPAT, 3-11
SPCHAR, 3-1, 13-10, 13-13
SYSCFG, 3-1, 3-12, 6-2, A-6
SYSTAT, 4-3, 4-5, 4-6
VADD, 3-11
VAL1K, A-4
VALUE, A-4
VECPAT, 3-10, 7-1
VGWAIT, A-4
VMOVE, 3-11
VSUB, 3-11
WAIT, A-6
WINDOW, 3-6
WMSK16, 8-1, A-3
WRMASK, 8-1, A-5
XFORM2D, 4-18 to 4-25
XHAIR, A-5
ZOOM, A-4
ZOOMIN, A-4
Commands,
described, A-2
graphics,
format of, 3-1, 3-2
in mnemonic form, 3-2

Commands, (cont.)
listed by category, 2-2
modified for the Model One/10, A-4
processing, 3-1
purposes of
for binary graphics, 3-1
for control of LUTs, 8-1
for alphanumeric emulation, 2-3
for data readback, 2-2
for display lists, 2-3, 3-14, 4-1
for graphics primitives, 2-2
for image transmission, 2-3
for interactive device support, 2-3
for look-up tables, 2-2
for macro programming, 2-3
for register operations, 2-3
for software development, 2-3
specific to the Model One/10, A-2
Communications, 2-2, 3-1
control of, 3-1
external, interrupt-driven, 6-2
internal, 6-2
Computer-aided design, 2-1
Conditional cursor movement sequences, 15-5
Configuration, 2-1, 13-4
Connecting the monitor, keyboard, and tablet, 11-3
Control and status registers, 6-2
Control characters
changing, 3-1
CTRL-D, 3-1
CTRL-Q, 13-10, 14-3
CTRL-S, 13-9, 14-3
CTRL-X, 3-2
Control codes, 15-1
displayed, 13-11
uses of, 14-1
Control keys, 14-1
Control program, 3-1
Controller, 2-1
location of, 11-1
Converters, digital-to-analog (DACs), 8-1, 8-2
Coordinates, manipulating, 3-4, 3-6
Coordinate origin register, 3-3
Coordinate registers, 3-4
loading, 3-4
manipulating, 3-4
modifying, 3-6

Coordinates, storage of, 3-4
Coordinate systems, 3-3
  changing, 3-3
  default, 3-3
CORORG, using, 3-4
Crosshairs, 3-5, A-5, A-6
CTRL key, 3-1, 3-2, 13-9, 14-3
CTRL-D, 3-1
CTRL-Q, 13-10, 14-3
CTRL-S, 13-9, 14-3
CTRL-X, 3-2
Cursor Address (<ESC> [row;col f or <ESC> [row;col H), 15-6
Cursor address, 15-5, 15-6
Cursor Down (<ESC> [n B), 15-6
Cursor Key mode, 14-4, 14-5, 15-14
Cursor Movement escape sequences, 15-4, 15-5
Cursor movement, absolute, 15-6
Cursor movement keys, 14-4
Cursors, 10-1
  firmware, A-5
  hardware, A-5
Cursor to the Left (<ESC> [n D), 15-6
Cursor to the Right (<ESC> [n C), 15-6
Cursor tracking, 3-13
Cursor type, 13-6, 13-10
Cursor Up (<ESC> [n A), 15-6

DAC, 8-1, 8-2
Damage, reporting, 11-1
Data,
  alphanumeric, 3-1, 3-2
  graphics, 3-1
Data paths, 2-5, 2-6
Data tablet, 2-1
DCS, 4-19
Debugger, entering, 3-2
Debugging, 2-5, 3-2, 3-13
Define Scrolling Region, 15-7
Defining and executing macro commands, 3-12, 3-13
DELETE key, 14-4
DELPID, 4-4, 4-5
Development tools, 2-4
Device Coordinate System, 4-19
Diagnostic and development features, 3-2
Diagnostic tools, 2-5
DIAG command, using, 11-5
Digital-to-analog converters, 8-1, 8-2
Digitizing tablet, 10-1
  executing macros, 3-14
Dimensions, of the Model One/10, 11-2
Display control, 6-1, 6-2, 9-1
  codes, 13-6
Display List firmware, 3-14
  applications, 4-1
  commands, by functional groups, 4-26
Display lists,
  capacity of, 11-2
  processing of, 6-1
Display monitor, 2-1
Display scaling, 9-1
Display-screen formats, 13-6, 13-11, 15-17, See also Escape sequences and Setup mode
Double-buffering, with read masks, 8-1
Environmental and technical specifications, 11-2
EOL, reporting, 15-11
Erase Line (<ESC> [2 K), 15-8
Erase Screen (<ESC> [2 J), 15-8
Erase to Beginnning of Line (<ESC> [1 K), 15-8
Erase to Beginning of Screen (<ESC> [1 J), 15-8
Erase to End of Line (<ESC> [K or <ESC> [0 K), 15-8
Erase to End of Screen (<ESC> [J or <ESC> [0 J), 15-8
Error messages, 3-2
ESC key, 14-3
Escape sequences, 15-3, 15-4, 15-14
  Blink On (<ESC> [5 m), 15-9
  Bold On (<ESC> [1 m), 15-9
  Clear All Tabs, (<ESC> [3 g), 15-7
  COLDstart (<ESC> c), 15-7
  Cursor Address (<ESC> [row;col f or <ESC> [row;col H), 15-6
  Cursor Down (<ESC> [n B), 15-6
  Cursor to the Left (<ESC> [n D), 15-6
  Cursor to the Right (<ESC> [n C), 15-6
  Cursor Up (<ESC> [n A), 15-6
  Define Scrolling Region (<ESC> [rowl;row2 r), 15-7
  Erase Line (<ESC> [2 K), 15-8
  Erase Screen (<ESC> [2 J), 15-8

Erase to Beginnning of Line (<ESC> [1 K), 15-8
Erase to Beginning of Screen (<ESC> [1 J), 15-8
Erase to End of Line (<ESC> [K or <ESC> [0 K), 15-8
Erase to End of Screen (<ESC> [J or <ESC> [0 J), 15-8
for character-erasing, 15-8
for conditional cursor movement, 15-5
for cursor movement, 15-4, 15-5
for reports, 15-10
for tab stops, 15-7
for selecting display-screen formats, 15-17
for selecting terminal color values, 15-15
for setting blinking and normal text, 15-17
for setting modes, 15-12
for setting overstrike mode, 15-18
for setting video attributes, 15-9
Index (<ESC> D), 15-5
Next Line (<ESC> E), 15-5
Reset Autowrap mode (<ESC> [? 7 l), 15-15
Reset Cursor Key mode (<ESC> [? 1 l), 15-14
Reset Local Echo mode (<ESC> [12 l), 15-14
Reset New Line mode (<ESC> [20 l), 15-13
Reset Origin mode (<ESC> [? 6 l), 15-15
Reset Overstrike mode (<ESC> [? 31 l), 15-18
Reset Reverse Screen (<ESC> [? 5 l), 15-14
Reset Text Blinking mode (<ESC> [? 10 l), 15-18
Reverse Index (<ESC> M), 15-5
Reverse Video On (<ESC> [7 m), 15-9
Select Background Color (<ESC> [? 3;r;g;b m), 15-16
Select Blinking Text Color (<ESC> [? 2;r;g;b m), 15-16
Select Normal Text Color (<ESC> [? 0;r;g;b m), 15-16
Select Screen Format (<ESC> [? fmt ~), 15-17
Set Autowrap mode (<ESC> [? 7 h), 15-15
Set Cursor Key mode (<ESC> [? 1 h), 15-14
Set/Clear LEDs (<ESC> [parameter q), 15-9
Set Local Echo mode (<ESC> [12 h), 15-14
Set New Line mode (<ESC> [20 h), 15-13
Set Origin mode (<ESC> [? 6 h), 15-15
Set Overstrike mode (<ESC> [? 31 h), 15-18
Set Reverse Screen (<ESC> [? 5 h), 15-14
Set Tab at Current Column (<ESC> H), 15-7
Set Text Blinking mode (<ESC> [? 10 h), 15-17
Underline On (<ESC> [4 m), 15-9
uses of, 15-1
Video Attributes Off (<ESC> [m), 15-9
What are the terminal parameters? (<ESC> [0 x or <ESC> [1 x), 15-11
What are you? (<ESC> [c or <ESC> [Z), 15-10
What is the alphanumeric cursor position? (<ESC> [6 n), 15-11
What is the terminal's status? (<ESC> [5 n), 15-10

EXMODE, 4-9
- highlight, 4-9, 4-17
- modes, 4-9
- normal draw, 4-9
- pick, 4-9, 4-10
- unhighlight, 4-10, 4-17

Fills,
- area, 3-9, 3-10, 3-11
- graphics, 3-9, 3-10, 3-11
- with patterns, 3-11

Firmware, 2-2, 3-1, See also Commands
Flags, reporting, 15-11
Flooding the screen, 9-1
Formats,
- ASCII-hexadecimal, 3-2
- binary, 3-2

FORTRAN, 1-1, 2-3
FORTRAN Library, 5-1
  conventions, 5-2
  documentation, 5-2
  INSTALL command, 5-4
    See Also INSTALL command
  installing, 5-3
  specifying serial I/O, 5-7
  use of hardcopy terminal, 5-2
  use of library, 5-6
Function key programming, 14-8
Function keys, 2-5, 3-13, 14-1
    Function key/button mapping,
      3-13
Functional blocks, 2-5

Graphics,
  command interpreter, 3-1
  commands, interpreting and
      decoding, 2-7
  display of, 3-3, 9-1, 15-177
Graphics data base, 3-14
Graphics mode, 3-1
  exiting, 3-1
  indicated with LED L1, 14-8
Graphics primitives and area
    fills, 3-9

Hardware, 2-5
Hardware vector generator, 2-5
HDCOPY, using, 11-5
Hexadecimal codes, displayed, 13-11
Hexadecimal parameter values, 3-1
Highlighting, 4-9, 4-16
Host, 2-3, 3-1
  connecting to, 11-3
Host baud, 13-6, 13-7
Host command stream, 2-2
Host data bits, 13-6, 13-7,
Host interface, 2-2, 11-2
Host parity type, 13-6, 13-7
Host port, 11-3
Host serial communications
    command format, 3-2

IEEE mode, 13-6, 13-7
Image, displayed, 3-3
Image memory, 2-5, 3-3, 3-6, 6-1,
    7-1, 11-2
  capacity of, 8-1
  clearing, 9-1
  configuration of, 8-1
  screen origin in, 9-1
Image memory output, 8-1
Index (<ESC> D), 15-5
Input/output buffers, 6-1
Input/output ports, 6-2
INSPID command, 4-6
Installation, 11-1
INSTALL command, 5-3
  command procedures, 5-5
  results, 5-3
  sub-directories, 5-5
Integrated graphics terminal, 2-1
Interactive device support, 2-2,
    10-1

KBD LOCKED LED, 14-8
Keyboard, 2-1, 3-1, 3-2
  connecting, 11-3
Keyboard (cont.)
  illustrated, 14-2
Keyboard controls, 14-1
Keypad keys, 14-1, 14-6
  function keys, 14-5
Keys with special functions in
    Setup mode, 14-7

LED indicators, 14-1, 14-7
Leg, adjustable, 11-3
LINE FEED key, 14-4, 15-5
Line feeds, with New Line, 13-10
Line generation, 7-1
Line numbers, 15-15
Lines, with patterns, 7-1
Local command format, 3-2
Local device interfaces, 11-2
Local interactive devices, 2-2,
    10-1
LOCAL LED, 14-8
Local processing, 2-1, 2-2, 2-3,
    2-4, 2-5, 2-7, 3-2, 3-14, 6-1
Look-up tables (LUTs), 2-2, 2-5,
    8-2
  control of, 8-1
  function of, 8-1, 8-2
LUTs, See Look-up Tables

Macro programming, 2-5, 3-2, 3-4,
    3-11, 3-12, 14-8
  executing, 3-13
  processing, 6-1
  using menus, 3-14
Macros
  creating, 3-12
  defining, 3-12
  executing with a puck, 3-14
  executing with commands, 3-13
  executing with function keys,
    3-13
  storing, 3-12

Masks,
  read, 8-1
  write-enable, 8-1
Menus, 3-14
Mnemonics, 2-5, 3-1, 3-2
Mode Escape sequences, 15-12
Model One/10 command set, A-2
Modes,
  alpha, 3-1
  graphics, 3-1
Modified commands, A-4

Monitor, 2-2
  connecting, 11-3
Mouse, 2-1, 10-1, 11-3
Nesting segments, 4-7
  example, 4-8
New commands, A-2
Newline, 13-6, 13-10
Next Line (<ESC> E), 15-5
NO SCRl key, 14-3
Non-applicable commands, A-3
Normal draw, 4-9
Normal text color, 13-6, 13-13
  selecting, 13-13
NVRAM, saving configurations, 13-4
ON LINE LED, 14-8
Opcodes, 3-1
Operands, 3-1
  decimal dormat of, 3-2
  hexadecimal format of, 3-2
Optional input devices, 3-1
Origin mode, 15-5, 15-6
Outlet required, 11-2
Overstrike, 13-6, 13-9
Overstrike mode, 15-18
Overview of the Model One/10's ports, 11-2
Overview of the Model One/10, 2-1

Panning, with SCRORG, 9-1
Parity, 3-1, 6-2, 11-3
  reporting, 15-11
Pattern register, 7-1
Patterned lines, 7-1
Peripherals, 2-1
Physical memory,
  size of, 6-2
  uses of, 6-2
Pickability, 4-4
Pick buffer
  specifying information to be stored, 4-11
  overflows, 4-12
Pick counters, 4-5
Pick ID, 4-3
  examples, 4-4
Picking, 2-7, 3-14, 4-9
  applications, 4-16
  executing, 4-12
  process, 4-10
Pick hit, 4-10
Pick tree, 4-11
Pixel update time, 11-2
Pixel value registers, 3-11, 3-12
POP, 4-23
Ports, 11-2
  configuration of, 3-2, 6-2
  connecting, 11-3
  firmware-configurable, 3-1
Ports, (cont.)
  for external communications, 6-1, 6-2
  for host, 3-1, 3-2
  for keyboard, 3-1
  for tablet/mouse, 3-1
  in Alpha mode, 3-1
  in Graphics mode, 3-1
  RS-232C, 3-1, 6-2
  serial, 3-1, 11-2
Power requirements, 11-2
Printer interface, 11-5
  connecting, 11-5
  using, 11-5
Processor,
  graphics, 11-2
  system, 11-2
  See Also Central Processor
Processor memory, 6-2
Programming, 5-1, See also Macro programming
Purpose
  of the Model One/10, 2-1, 2-7
  of this manual, 1-1
PUSH, 4-23
Pushing/Popping transformations, 4-23

RAM, 6-2
Raster Technologies, contacting 11-6
RDPICK, 4-13 to 4-15
RDPID, 4-15
RDMODE, 4-5
RDREG, 4-5
RDXFORM, 4-5
Read masks, 8-1
Real time, 8-1
Receive speed, reporting, 15-11

Red-green-blue outputs, 8-3
Registers
  control, 7-1, 8-1
    for display, 6-2, 9-1
    for placement of screen origin, 9-1
    vector generator, 6-2
    write-only, 6-2
  coordinate, 3-3, 3-5, 3-13
  pixel value, 3-11 to 3-13
  status, 6-2
    functions of, 6-2
    read-only, 6-2
  vector pattern, 3-10
Repair, 11-1
Report sequences, 15-10, See also Escape sequences
Report type, reporting, 15-11
Reset Autowrap mode (<ESC> [? 7 l), 15-15
Reset Cursor Key mode <ESC> [? 1 l), 15-14
Reset Local Echo mode <ESC> [12 l), 15-14
Reset New Line mode <ESC> [20 l), 15-13
Reset Origin mode <ESC> [? 6 l), 15-15
Reset Overstrike mode (<ESC> [? 31 l), 15-18
Reset Reverse Screen (<ESC> [? 5 l), 15-14
Reset Text Blinking mode (<ESC> [? 10 l), 15-18
Resolution, 2-1, 11-2
RETURN key, 14-4, 15-13
Reverse Index (<ESC> M), 15-5
Reverse Screen, 13-6, 13-11
Reverse Video On (<ESC> [7 m), 15-9
ROM, 6-1, 6-2
Rows
  numbers of, 13-11
  reporting location of, 15-11
RS-232C, 11-2 See also Ports

Scaling, 9-1
Screen brightness, 13-3, 13-6
  adjusting, 13-6
Screen origin, 9-1
  modification of, 9-1
  placement of, 9-1
Screen refresh, 9-1
Scrolling regions, 15-4, 15-5, 15-15
SEGDEF, 4-6
SEGEND, 4-6
SEGINI, 4-3, 4-6
SEGINQ, 4-5, 4-6
Segment ID, 4-3
  registers, 4-5
Segments,
  child, 4-7, 4-9
  defining, 4-5
  description, 4-1
  entering, 4-12
  example, 4-2
  executing, 4-6, 4-7, 4-12
  modifying, 4-5, 4-6
  nesting, 4-7 to 4-9
  overview, 4-1
  parent, 4-7, 4-9
  reading back information, 4-5, 4-13
SEGREF, 4-2, 4-6, 4-7
Select Background Color (<ESC> [? 3;r;g;b m), 15-16
Select Blinking Text Color (<ESC> [? 2;r;g;b m), 15-16
Select Normal Text Color (<ESC> [? 0;r;g;b m), 15-16
Select Screen Format (<ESC> [? fmt ~), 15-17
Selecting setup features, 13-3
Selecting terminal features using setup mode, 13-1
Self-testing, with COLDstart, 15-7
Serial communications, interrupt-driven, 11-3
Serial I/O, 3-1
Serial numbers, 11-1, 11-3
Set Autowrap mode (<ESC> [? 7 h), 15-15
SETATR, 4-4
Set Cursor Key mode (<ESC> [? 1 h), 15-14
Set/Clear LEDs (<ESC> [parameter q), 15-9
SETCL, 4-13
Set Local Echo mode (<ESC> [12 h), 15-14
Set New Line mode (<ESC> [20 h), 15-13
Set Origin mode (<ESC> [? 6 h), 15-15
Set Overstrike mode (<ESC> [? 31 h), 15-18
Set Reverse Screen (<ESC> [? 5 h), 15-14

Set Tab at Current Column
  (<ESC> H), 15-7
Set Text Blinking mode
  (<ESC> [? 10 h), 15-17
SET-UP key, 14-1
Set/Clear LEDs escape sequence,
  15-9
Setup A features defined, 13-5
Setup A, display, 13-2
Setup B features defined, 13-6
Setup B, display, 13-3
Setup mode, 3-1, 6-2, 13-1, 14-1,
  15-1, 15-12, 15-15, A-6
  Autowrap, 13-6, 13-10
  Background color, 13-6, 13-12
Setup mode, (cont.)
  Blinking text color, 13-6, 13-12
  Cursor type, 13-6, 13-10
  Display control codes, 13-6,
    13-11
  Display format, 13-6, 13-11
  Entering, 13-1
  Exiting, 13-4
  Host baud, 13-6, 13-7
  Host data bits, 13-6, 13-7
  Host Graphics mode, 13-6, 13-9
  Host parity type, 13-6, 13-7
  Host parity, 13-6, 13-7
  IEEE mode, 13-6
  Newline, 13-6, 13-10
  Normal text color, 13-6, 13-13
  Overstrike, 13-6, 13-9
  Reverse screen, 13-6, 13-10
  Solid background, 13-6, 13-11
  Tablet type, 13-6, 13-9
  Text blinking, 13-6, 13-10
  XON/XOFF, 13-6, 13-9
SHIFT key, 14-3
SHIFT-S, 13-4
Solid background, 13-6, 13-11
SPACE BAR key, 14-4
Special characters,
  changing, 3-1, 13-10, 13-13
    See Also Control
    Characters
  default, 13-13
Stacking, 4-23
Storage
  of coordinates, 3-13
    See Also Coordinate
    registers
  of pixel values, 3-11 to 3-13
SYSTAT, 4-3, 4-5, 4-6
System and image memory, 8-1
System architecture, 2-5
System memory, 11-2

Tab escape sequences, 15-7
TAB key, 14-3
Tab stops, See tabs
Tablet, 10-1
  connecting, 11-3
  port, 11-3
  scaling ration, 10-1
  type, 13-6, 13-9
Tabs, 13-3
  changing, 13-5
  defaults, 13-5
Temperatures, operating, 11-2
Terminal,
  adjustable tilt, 11-3
  type, VT100
Terminal configurations, 15-10
  changing, 11-3, 13-4
  defaults, 11-3
  saving, 13-4, 13-5
Terminal features
  changing with control
    sequences, 13-1
  changing with Setup mode, 13-1,
    15-1
  defaults, 13-1
Testing, 3-2
Text, displayed or supressed, with
  ALPHEM, A-4
Text blinking, 13-6, 13-12
  color of, 13-11, 15-14, 15-16
Text, printing, 11-5
Transformations, 2-3, 3-14, 4-17
Transmit speed, reporting, 15-11

Underline On (<ESC> [4 m), 15-9
Unhighlight
Unpacking and Cabling
  the ports, 11-1
Using control codes and escape
  sequences, 15-1
Using keyboard controls and
  LED indicators, 14-1
Using Button Table to execute
  macros, 3-13
Using this manual, 1-1

Value registers, 3-11, 3-12
Values, image data, 3-11
VAX FORTRAN library, 5-1
  See Also FORTRAN library
Vector generator, 2-2, 2-7, 6-2,
  7-1
Vector pattern register, 3-10
Vector-to-raster conversion, 7-1

Vents, 11-2
Video amplifiers, 8-3
   stage, 8-3
   summing, 8-3
Video Attribute escape sequences, 15-9
Video Attributes Off (<ESC> [m), 15-9
Video DACs, 8-1, 8-2
Video frames, 3-2
Video monitor, 8-3
Video output, 2-5, 8-3
Video signal, 8-3
Video sync, 8-3
Viewing the status of Setup features, 13-1
Visibility, 4-4, 4-17
VLSI, 2-2
VREGs, 3-11, 3-12
VT100, 2-3, 15-10

WCS, 4-19
What are the terminal parameters?, (<ESC> [0 x or <ESC> [1 x), 15-11
What are you? (<ESC> [c or <ESC> [Z), 15-10
What is the alphanumeric cursor position? (<ESC> [6 n), 15-11
What is the terminal's status? (<ESC> [5 n), 15-10
Window, clearing, 9-1
Windows, clipping, 3-6
World Coordinate System, 4-19
Wraparound (autowrap), 13-6, 13-10
Write-enable masks, 8-1

XFORM2D, 4-18 to 4-25
   current point, 4-22
   description, 4-17
   examples of input, 4-22
   matrix elements, 4-20
   options, 4-19
XOFF and XON codes, 14-3
XON/XOFF
   protocol, 3-1, 6-2, 11-3
   setup, 13-6, 13-9, 14-1

Z8001 microprocessor, 2-2, 6-1, 6-2

Raster Technologies

Model One

ERROR MESSAGES REFERENCE GUIDE

Revision 2.0

November 15, 1985

Part # 552-00033-001

Raster Technologies, Inc.
9 Executive Park Drive
North Billerica, MA 01862
(617) 667-8900

RASTER TECHNOLOGIES
MODEL ONE

Chapter 1: Introduction to the Reference Guide

OVERVIEW

Purpose

This Error Messages Reference Guide provides an explanation of the error messages generated by the Raster Technologies Model One/10, Model One/80, and Model One/380 graphics systems.

This Guide lists all the error messages and provides the basic information needed to handle the error conditions indicated by the messages.

Error Messages

The Model One command interpreter sends error status information to the local alphanumeric terminal whenever an error occurs.

Use of READER

In addition to sending error status information to the local alphanumeric terminal, the Model One sets an internal byte to indicate the error number of the first error to occur after the COLDstart or since the last READER command.

You can examine this internal byte using the READER command. READER command clears the buffer which stores the error status byte.

Examples: If the first error to occur after a COLDstart was error 001 (bad command opcode), the READER command would return

001

If no errors exist, then READER returns

000

Note: Error message "000 Illegal Call to Routine ERROR" is very unlikely to occur (it results from a firmware/hardware error). You should assume that if READER returns "000", no error exists. If an application fails and no other error messages besides "000" are generated, then "000" may indicate an illegal call to ERROR.

Continued on next page

OVERVIEW, continued

Organization of the Guide

This Guide is organized into 3 chapters.

This first chapter provides an introduction to Model One error messages and to this Guide.

The second chapter provides a quick reference of basic information to help you correct error conditions without having to refer to any other documentation. Information provided includes command formats, a listing of acceptable ranges for common parameters, and configurations.

The third chapter lists each error message, in numerical order, and provides an explanation and/or suggestions to help you handle the error condition.

Other Documentation

Each Model One is shipped with a complete set of documentation. The table below highlights the types of information to be found in each document.

Note: The following table uses general titles for each type of document; the titles vary for each type of Model One (e.g., Model One/10, Model One/80, etc.). Additional documentation is included for most types of Model Ones.

Document	Contents
Introduction/Installation Guide	Information about configurations, diagnostics, and basic maintenance
Programming Guide	Description of features and how to use the command set
Command Reference	Description of the syntax, FORTRAN calls, opcodes, and ranges for parameters
Programming Card	Summary of command syntax
Release Notes	List of corrected problems/enhancements, and any known problems

TYPES OF ERROR MESSAGES

Introduction

The Model One error messages can be grouped into several major catagories.

This page classifies error messages into 4 basic groups:

- messages indicating improper use of commands
- messages indicating improper use of features
- self-test messages, and
- fatal hardware error messages.

Each of these groups is discussed in more detail below.

Groups Referenced in Error Message Listing in Chapter 3

The listing of error messages in Chapter 3 indicates for each error message the general type of the error message, as described on these pages.

Improper Use of Commands Messages

Description: The Model One generates several error messages indicating that a Model One command has been used improperly.

These messages can result from two basic types of user mistakes:
- "typos", and
- providing the wrong information for mnemonics/parameters.

Examples: The following are examples of error messages reflecting improper use of commands.

- 001 Bad command opcode
- 004 Number is out of range
- 005 String is not a number.

General suggestions: You should first check that you entered the command as you had intended. If the error doesn't seem to be a "typo", then you should check the documentation, including

- on-line help
- Chapter 2 of this Guide
- Programming Card
- Command Reference.

Continued on next page

TYPES OF ERROR MESSAGES, continued

Improper Use of Features

Description: The Model One generates several error messages indicating errors relating to the use of Model One features such as macros, display list, and alphanumeric windows. Some of these messages relate to how a specific command must be used, while other messages relate to how two or more commands must be used in conjunction with each other.

Examples: The following are examples of error messages reflecting improper use of Model One features:

- 019 Allowed only in macro definitions
- 072 Window not defined
- 085 Segment already defined.

General suggestions: Several of the messages in this group indicate that you cannot perform an action because of actions set up (or not set up) earlier in the application. Thus you need to change either the action you wish to perform, or change an earlier action.

If you need additional information to resolve the error condition, you should refer to the Command Reference and/or the Programming Guide. For some types of Model Ones the documentation is divided into separate documents relating to specific features, such as display lists or alphanumeric terminal emulation.

Self-Test Messages

Description: The Model One/80 and One/380 perform several self-test routines to identify possible hardware errors. These self-tests are run at power-up and COLDstart.

Examples: The following are examples of some of the self-test error messages generated by the Model One/80 and One/380:

- 059 Self-test error, serial port
- 066 Pixel readback self-test timeout
- 070 Unable to fill vector queue.

Continued on next page

TYPES OF ERROR MESSAGES, continued

Self-Test Messages, continued

General suggestions: If someone has just made any changes to the Model One hardware (e.g., changed PROMs or cabling), then these error messages may indicate that some mistake was made while modifying the hardware. Check that the PROMs, cabling, pinouts, and jumpers are configured properly. The Installation Guide provides information about how the system should be installed/congfigured.

If no modifications have been made recently to the Model One hardware, then the problem is probably at the board level. In this case, you should

- record the configuration of your system
- record the revision level of the boards and firmware
- record what actions you took before the error message occurred, and
- contact Raster Technologies or your local representative.

See the following page, called CONTACTING RASTER TECHNOLOGIES.

Fatal Hardware Errors

Description: There are several error messages that indicate a fatal hardware error condition that will not allow the Model One to be used until the failing hardware has been repaired/replaced.

Examples: The following are examples of fatal hardware error messages:

- 128 Base RAM data fault
- 161 MAPRAM verification
- 165 BP Signature analysis error
- 170 Pipeline register failure.

General suggestions: These messages indicate a serious hardware problem. You should contact Raster Technologies or your local representative. Be prepared to provide the following information:

- the actions you took leading up to the error condition
- the configuration of your system, and
- the revison level of the boards and firmware.

See the following page, called CONTACTING RASTER TECHNOLOGIES.

CONTACTING RASTER TECHNOLOGIES

Contact Customer Service

The Customer Service Department at the Raster Technologies main office is prepared to answer your questions concerning error messages relating to both hardware and firmware problems.

Main Office Address and Phone Number

The address and phone number of the Raster Technologies main office is

Raster Technologies
9 Executive Park Drive
North Billerica, MA 01862

(617) 667-8900

Contacting Your Local Raster Technologies Office

You can contact your local Raster Technologies office if you have questions relating to using the Model One command set. Hardware related questions should be directed to our main office (see above).

Raster Technologies has local offices in

- Houston, Texas	(713) 460-4741
- Orange, California	(714) 835-3791
- Sunnyvale, California	(408) 720-0440

Chapter 2: Quick Reference Of Basic Information

OVERVIEW

Purpose

This chapter serves as a quick reference to help you handle error messages, particulary relating to improper use of commands.

This chapter does not replace the other documentation for the Model One; rather, it provides some basic information relating to general command formats and acceptable ranges for common parameters.

Contents

This chapter is organized into the following sections:

- General Command Formats
- Listing of Acceptable Ranges for Common Parameters

GENERAL COMMAND FORMATS

Introduction

These pages describe the general formats for Model One commands. Failure to use the correct command format could result in error messages such as

- 002 Unrecognized command
- 005 String is not a number.

Types of Command Formats

You can send Model One commands from the local alphanumeric terminal.

In addition, the FORTRAN library includes routines corresponding to each Model One command.

Local Alphanumeric Terminal Command Format

The Model One expects commands over the local alphanumeric terminal port to be entered in a special English-like mnemonic form, rather than in binary or hexadecimal form.

Parameters can be specified using mnemonics (where appropriate) or decimal or hexadecimal numbers. Some commands require that text strings be specified.

Abbreviating Command Mnemonics

You can abbreviate the command mnemonic for most Model One commands. You abbreviate by deleting letters from a selected part of the mnemonic to the end.

You cannot abbreviate by deleting some letters and then including any letters that follow. For example, you could abbreviate MOVABS to MOVA, but not to MOAB.

Note: The Model One is designed to associate certain abbreviations with a specific command. Your abbreviation may send a different command than you may have intended. For example, if you enter "SEG" as an abbreviation for the SEGINI command, the Model One will in fact interpret the "SEG" as an abbreviation for the SEGREF command, which requires one parameter, not the two required for the SEGINI command. Abbreviations that could apply to more than one command are reconciled by the Model One by opcode order (the command with the lower opcode is selected).

Continued on next page

GENERAL COMMAND FORMATS, continued

Case

You can use upper case or lower case, or a mix of the two cases.

Spacing and Commas for Local Commands

You must include one space between
- the command mnemonic and its parameters, and
- each parameter (you can include more spaces without causing an error).

You may include commas between parameters (in addition to or instead of a blank space). You must not use commas when specifying numeric values (e.g., enter 1,000 as 1000).

Specifying Hexadecimal Numbers

You must precede each hexadecimal parameter value with a pound sign (#).

You can specify a combination of hexadecimal and decimal numbers for the same command (e.g., MOVREL #32 50)

Filling Leading Zeros Unnecessary

You do not have to include leading zeros, (i.e., you can enter just 4; you do not have to enter 004).

Examples of Alphanumeric Terminal Command Formats

The following examples illustrate proper formats for specifying local commands.

```
! MOVABS 4 6                  ; Move current point to 4,6.
! VALue 255, 255, 255         ; Set current pixel value to white.
! DR 100 -10                  ; Draw line from current point (4,6) to
                              ; 100,-10 (DR is an acceptable abbreviation
                              ; for the DRWABS command).
! RECREL #-14 #1E             ; Draw rectangle with diagonally opposite
                              ; corner displaced by -20,30 (decimal)
```

FORTRAN Calls

FORTRAN calls must be made consistent with standard FORTRAN syntax.

LISTING OF ACCEPTABLE RANGES FOR COMMON PARAMETERS

Introduction

Several error messages are caused by specifying a value for a parameter that does not fall within the acceptable range for that parameter. Examples of error messages that may result for using invalid values for parameters include

- 004 Number is out of range
- 073 Bad character size
- 075 Illegal window number
- 088 Illegal block size.

These pages list acceptable ranges for common parameters used with several Model One commands.

This is intended to save you from having to look up these values in other documentation.

Table of Acceptable Ranges for Common Parameters

Parameter	Acceptable Range	Associated Commands
Alphanumeric window numbers	0 to 7	DEFWIN, DELWIN
Button index	-64 to 63	BUTTBL, BUTTON
Cdest	0 to 63	CADD, CIRCI, CLOAD, CMOVE, CSUB, DRWI, MOVI, READCR, RECTI, XMOVE
Cdiff	0 to 63	CADD, CIRCI, CLOAD, CMOVE, CSUB, DRWI, MOVI, READCR, RECTI, XMOVE
Columns	1 to 1280	READW, READWE, RUNLEN. RNLEN8
Coordinate register	0 to 63	CADD, CIRCI, CLOAD, CMOVE, CSUB, DRWI, MOVI, READCR, RECTI, XMOVE
Creg	0 to 63	CADD, CIRCI, CLOAD, CMOVE, CSUB, DRWI, MOVI, READCR, RECTI, XMOVE

Continued on next page

LISTING OF ACCEPTABLE RANGES FOR COMMON PARAMETERS, continued

Table of Acceptable Ranges for Common Parameters, continued

Parameter	Acceptable Range	Associated Commands
Fact (for ZOOM)	1 to 16	ZOOM
Frames	0 to 255	BLINKR, WAIT
Index (button)	-64 to 63	BUTTBL, BUTTON
Index (LUT)	0 to 255 (Model One/10 allows 0 to 1023 for LUT16)	BLINKD, BLINKE, LUT8, LUT16, LUTA, LUTB, LUTG, LUTR, LUTRMP,
Look-up table index	0 to 255 (Model One/10 allows 0 to 1023 for LUT16)	BLINKD, BLINKE, LUT8, LUT16, LUTA, LUTB, LUTG, LUTR, LUTRMP,
Look-up table	0 to 7	BLINKD, BLINKE
Lut (look-up table value)	0 to 7	BLINKD, BLINKE
Macnum	0 to 255	BUTTBL, MACERA
Macro number	0 to 255	BUTTBL, MACERA
ncols	0 to 1280	READW, READWE, RUNLEN, RUNLN8
Nrows	0 to 1024	READW, READWE, RUNLEN, RUNLN8
Segment	-32,768 to 32,767	SEGAPP, SEGCOP, SEGDEF, SEGDEL, SEGEND, SEGINI, SEGINQ, SEGREF, SEGREN, SELWIN, SETATR

Continued on next page

LISTING OF ACCEPTABLE RANGES FOR COMMON PARAMETERS, continued

Table of Acceptable Ranges for Common Parameters, continued

Parameter	Acceptable Range	Associated Commands
Value register	0 to 63	AREA2, BLINKE, RDPIXR, READVR, VADD, VLOAD, VMOVE, VSUB
Vdif	0 to 63	AREA2, BLINKE, RDPIXR, READVR, VADD, VLOAD, VMOVE, VSUB
Vreg	0 to 63	AREA2, BLINKE, RDPIXR, READVR, VADD, VLOAD, VMOVE, VSUB
Vsum	0 to 63	AREA2, BLINKE, RDPIXR, READVR, VADD, VLOAD, VMOVE, VSUB
Window	0 to 7	DEFWIN, DELWIN
Zooming factor	1 to 16	ZOOM

Chapter 3: Listing of Error Messages

OVERVIEW

Introduction

This chapter provides a complete listing of all error messages for the Model One/10, Model One/80, and Model One/380. Error messages are listed in numerical order.

Information Provided

For each error message, the following information is provided:

- error message number
- error message text
- explanation of error message and suggestions for handling the error condition.

The general type of error message (as discussed in Chapter 1 (e.g., hardware, improper use of command)) is indicated for each error message.

Message Text

For a few error messages, the text varies slightly for different models. For example, Model One/10 error messages may refer to the Z8001, while the Model One/80 and Model One/380 refer to the Z8002. However, the meaning of the message is similar for all models, unless specifically noted.

Reserved Messages

Several numbers have been reserved for future error messages. These are indicated on the following pages by the word "RESERVED", with no explanatory text.

If you receive an error message listed as "RESERVED", contact Raster Technologies or local representative if necessary.

Access Aid

The top of each of the following pages indicates the error messages included on that page (e.g., 000 to 005 indicates that those 6 error messages appear on that page).

000 to 005

Error Messsage #	Message	Explanation/Suggestions
000	Illegal call to routine "ERROR"	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.
001	RESERVED	---
002	Unrecognized command	improper command usage - command mnemonic used does not exist - check for typos - the command HELP lists all the available commands
003	Unimplemented command	improper command usage - command mnemonic exists but has not been implemented - unimplemented commands do not hang the system, and may be left in an application for compatibility, although they do not perform any action.
004	Number is out of range	improper command usage - range of a parameter has been exceeded - see Chapter 2 for list of parameter ranges
005	String is not a number	improper command usage - parameter value must be a decimal or hexadecimal number - each hexadecimal must be preceded by #

006 to 009

Error Messsage #	Message	Explanation/Suggestions
006	RESERVED	---
007	Illegal parameter	improper command usage - parameter value within range (i.e., right number of bits), but is invalid (e.g., PRMFIL 2 (flag must be 0 or 1))
008	Comm Ctrl timeout-Z8002 output to host	hardware - the Z8002 cannot sent data to the host - problem could be caused by the host, serial lines, or Communications Controller - try a BREAK or WARMstart - if error persists, write down the circumstances under which it occurred, and - contact Raster Technologies or local rep.
009	Comm Ctrl timeout-Z8002 output to BP	hardware - the Z8002 cannot sent data to the Bipolar Processor (BP) - BP may be involved in lengthy task - problem could be caused by the BP, serial lines, or Communications Controller - try a BREAK or WARMstart - if error persists, write down the circumstances under which it occurred, and - contact Raster Technologies or local rep.

010 to 012

Error Messsage #	Message	Explanation/Suggestions
010	Comm Ctrl timeout-Z8002 input from BP	hardware - the Bipolar Processor (BP) did not provide data requested by the Z8002 - BP may be involved in lengthy task - problem could be caused by the BP, serial lines, or Communications Controller - try a BREAK or WARMstart - if error persists, write down the circumstances under which it occurred, and - contact Raster Technologies or local rep.
011	Comm Ctrl timeout-Z8002 output latch	hardware - the Z8002 output latch was not empty before changing Communications Controller (Comm Ctrl) modes - problem could be caused by the Comm Ctrl - try a BREAK or WARMstart - if error persists, write down the circumstances under which it occurred, and - contact Raster Technologies or local rep.
012	Comm Ctrl timeout-BP not waiting	hardware - the Bipolar Processor (BP) did not provide data requested by the Z8002 - BP may be involved in lengthy task - problem could be caused by the BP, serial lines, or Communications Controler - try a BREAK or WARMstart - if error persists, write down the circumstances under which it occurred, and - contact Raster Technologies or local rep.

013 to 016

Error Messsage #	Message	Explanation/Suggestions
013	Comm Ctrl timeout-BP output latch	hardware - the Bipolar Processor (BP) did not provide data requested by the Z8002 - BP may be involved in lengthy task - problem could be caused by the BP, serial lines, or Communications Controller - try a BREAK or WARMstart - if error persists, write down the circumstances under which it occurred, and - contact Raster Technologies or local rep.
014	Macro calls nested too deep	improper feature usage - macro calls may not be nested more than 16 levels deep
015	Comm Ctrl timeout-no EOT	hardware - Bipolar Processor (BP) didn't finish a data transmission - BP problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
016	Attempt to erase active macro	improper feature usage - a macro which is currently being executed cannot be erased (using MACERA). - if the macro is being executed as a result of a button table location 0, you can change the BUTTBL command.

017 to 022

Error Messsage #	Message	Explanation/Suggestions
017	Not enough space for definition	improper feature usage - the available space in RAM for macros and downloaded text has been used - on the Model One/80 or Model One/380 you may want to either erase macros (MACERA) or text (TEXTRE) or - you may want to reconfigure RAM using the CONFIG command. CONFIG destroys all currrently defined macros, text, alpha windows, and downloaded text. - The default configuration is macro definitions 8K bytes areafill/polyfill scratch 1K bytes downloaded text 2K bytes alpha windows 4K bytes downloaded code 1K bytes (MAP shows current space allocation)
018	RESERVED	---
019	Allowed only in macro definition	improper feature usage - the MACEND command can only be used inside a macro definition
020 to 022	RESERVED	---

023 to 028

Error Messsage #	Message	Explanation/Suggestions
023	Insufficient space to complete operation	improper feature usage - for the Model One/10, indicates a PUSH/POP under/overflow condition; there are 4K bytes of stack storage available - for the Model One/80 or One/380 the POLYGN, AREA1, AREA2, or rectangle command ran out of scratch space to fill the specified area. - on the Model One/80 or Model One/380 you may want to reconfigure RAM using the CONFIG command. CONFIG destroys all currrently defined macros, text, alpha windows, and downloaded text. - The default configuration is macro definitions 8K bytes areafill/polyfill scratch 1K bytes downloaded text 2K bytes alpha windows 4K bytes downloaded code 1K bytes (MAP shows current space allocation)
024 to 027	RESERVED	---
028	Bad record format	improper command usage - a DNLOAD record was incorrectly formatted. The proper format is - backslash character (/) - 2 bytes (start address for loading data) (for the end-record, address at which command is to start execution) - 1 byte (number of bytes of data in record (for end-record, n = 0) - 1 byte (checksum) - n bytes (data being downloaded) (for end-record, no data is specified) - 1 byte checksum

029 to 036

Error Messsage #	Message	Explanation/Suggestions
029	No start address	improper command usage - start address must be specified for the DNLOAD command (see Error 028); it cannot be 0
030	Bad checksum	improper command usage - one of the 2 checksums used with the DNLOAD command is bad (see Error 028)
031	Name table overflow	improper feature usage - command names specified with the DNLOAD command exceed available space - try shortening names of downloaded commands
032	Blink table overflow	improper command usage - up to 8 blink table entries (BLINKE) are allowed
033	Loading into protected area	improper command usage - start address cannot be used for downloading code (DNLOAD)
034 to 036	RESERVED	---

037 to 040

Error Messsage #	Message	Explanation/Suggestions
037	Input queue full on serial port	improper feature usage - reset the input queue for one of the serial ports, using the SYSCFG SERIAL command - the total of all queue sizes for the serial ports cannot exceed 6K
038	Overrun error on serial port	hardware - probably a firmware problem - interrupts not being handled quickly enough - write down circumstances under which it occurred, including configuration - contact Raster Technologies or local rep.
039	Parity error on serial port	improper feature usage - reset the parity for one of the serial ports, using the SYSCFG SERIAL comman - parity can be set to Even (E), Odd (O), High (parity bit always set (H), Low (L), or None (N).
040	Framing error on serial port	improper feature usage - reset the baud rate for one of the serial ports, using the SYSCFG SERIAL command - the baud rate can be set to 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 480, 9600, 19200, or 38400

041 to 048

Error Messsage #	Message	Explanation/Suggestions
041	Break detected on serial port	<u>improper feature usage</u> - BREAK received, but control characters not allowed by the serial port - reset the CTRL parameter, using the SYSCFG SERIAL command - CTRL = ON instructs the Model One to accept control characters from the port; CTRL = OFF ignores control characters
042 to 044	RESERVED	---
045	Can only be executed from Alpha port	<u>improper feature usage</u> - the DFTCFG, DISCFG, SAVCFG, and SYSCFG commands cannot be executed from any other port than the Alpha-numeric terminal port
046	RESERVED	---
047	Model One firmware failure	<u>hardware</u> - write down circumstances under which it occurred - contact Raster Technologies or local rep.
048	IEEE-488 bus error failure	<u>hardware</u> - write down circumstances under which it occurred - contact Raster Technologies or local rep.

049 to 054

Error Messsage #	Message	Explanation/Suggestions
049	Bad Z8002 vectored interrupt	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.
050	Illegal Z8002 instruction	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.
051	Privileged Z8002 instruction	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.
052	Z8002 segmentation trap	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.
053	Bad Z8002 non-vectored interrupt	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.
054	Bad Z8002 non-maskable interrupt	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.

<u>055 to 073</u>

Error Messsage #	Message	Explanation/Suggestions
055	Bad Z8002 system call	<u>hardware</u> - write down circumstances under which it occurred - contact Raster Technologies or local rep.
056	Insufficient memory space for the configuration	<u>improper command usage</u> - total memory (16K bytes) was exceeded while using the CONFIG command
057 to 061	RESERVED	---
062	Macro definition not allowed in debugger	<u>improper feature usage</u> - a macro cannot be defined while in debugger mode (DEBUG ON)
063 to 071	RESERVED	---
072	Window not yet defined	<u>improper feature usage</u> - an alphanumeric window must be selected before using it with an alphanumeric terminal emulation command such as ALPHEM, DELWIN, or SELWIN
073	Bad character size	<u>improper feature usage</u> - the character size for an alphanunumeric window may have exceeded the bounds of the alpha window

074 to 079

Error Messsage #	Message	Explanation/Suggestions
074	Bad window size	improper feature usage - window size, as defined with the DEFWIN command, must be given in Model One coordinates
075	Illegal window number	improper feature usage - alphanumeric window number must be 0 - 7
076	No window selected	improper feature usage - an alphanumeric window must be selected (using SELWIN) before using alphanumeric terminal emulation commands such as DIRCUR, GETCUR, or GETPOS
077	NVRAM checksum error detected: NVRAM re-intialized	hardware - the system automatically initializes NVRAM to the default condition - if you had a configuration other than the the default, use the SYSCFG and SAVCFG commands to re-establish the configuration you want - on some systems, with extreme failures, re-initializing NVRAM may result in changing Model One options. If this occurs, contact Raster Technologies or local rep.
078	Cannot delete selected window	improper feature usage - the currently active alphanumeric window cannot be deleted (using the DELWIN command)
079	RESERVED	---

080 to 085

Error Messsage #	Message	Explanation/Suggestions
080	Segment table full	improper feature usage - segment definitions exceed the available memory for segment definition storage - delete existing segments (SEGDEL) to make room for any additional segment definitions - for Model One/80s and One/380s, additional bulk memory cards may be added to most systems; contact Raster Technologies or local rep.
081	Display list memory full	improper feature usage - segment definitions exceed the available memory for segment definition storage - delete existing segments (SEGDEL) to make room for any additional segment definitions - for Model One/80s and One/380s, additional bulk memory cards may be added to most systems; contact Raster Technologies or local rep.
082	Segment already defined	improper feature usage - the segment number already used - either rename the segment (SEGREN) or delete the existing segment with the same segment number (SEGDEL)
083	Segment not found	improper feature usage - a segment must be defined (using the SEGDEF command) before using the SEGAPP command
084 to 085	RESERVED	---

086 to 095

Error Messsage #	Message	Explanation/Suggestions
086	Dispaly list integrity questionable	- segment definition has been corrupted - redefine the segment (SEGDEF) - if problem persists, write down the circumstances under which it occurred and contact Raster Technologies or local rep.
087	RESERVED	---
088	Illegal block size	improper command usage - block size for the SEGINI command must within the range of 32 to 16,384
089	Illegal transform	improper feature usage - illegal parameter specified for the XFORM2D or XFORM3D commands
090	Segment referenced nested too deeply	improper feature usage - segments may not be nested more than 16 levels
091	Allowed only during segment definition	improper feature usage - the SEGEND command cannot be used outside a segment definition
092 to 095	RESERVED	---

096 to 129

Error Messsage #	Message	Explanation/Suggestions
096	Invalid host baud rate	improper feature usage - reset host baud rate using SYSCFG SERIAL command - acceptable baud rates for the Model One are 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, or 38,400
097	RESERVED	---
098	Stack overflow or underflow	improper feature usage - an underflow condition is caused by trying to POP more items than were PUSHed - the Model One/80 and One/10 allow 2000 words for stacking - the Model One/380 allows 2000 words for each of its two stacks
099	MODDIS Reset not allowed during SEGREF or MACRO	improper feature usage - the MODDIS command cannot be reset while a segment or macro is being referenced
100 to 127	RESERVED	---
128	Base RAM data fault	self-test - write down circumstances under which it occurred - contact Raster Technologies or local rep.
129	RAM mis-hap: address	self-test - write down circumstances under which it occurred - contact Raster Technologies or local rep.

130 to 135

Error Messsage #	Message	Explanation/Suggestions
130	Paged RAM test failure- page	self-test - write down circumstances under which it occurred - contact Raster Technologies or local rep.
131	Serial port timeout failure: port	self-test - write down circumstances under which it occurred - contact Raster Technologies or local rep.
132	Serial port self-test error: port	self-test - write down circumstances under which it occurred - contact Raster Technologies or local rep.
133	PROM checksum error: U	self-test - try reseating PROMs (see Installation Guide or Release Notes) - if error persists, write down the circumstances under which it occurred - contact Raster Technologies or local rep.
134	Polynomial Generator input fault	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.
135	Polynomial Generator function fault	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.

136 to 163

Error Messsage #	Message	Explanation/Suggestions
136	GPU signature analysis error	hardware - write down circumstances under which it occurred - contact Raster Technologies or local rep.
137 to 159	RESERVED	---
160	WCS verification	hardware - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
161	MAPRAM verification	hardware - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
162	Vertical blanking failure	self-test - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
163	Fault in micro-address shift register	self-test - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.

164 to 168

Error Messsage #	Message	Explanation/Suggestions
164	COMM Ctrl - IPBD bus fault	hardware - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
165	BP signature analysis error	hardware - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
166	Unable to find start-of-frame	hardware - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
167	Comm Ctrl failur Z8002 input latch	hardware - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
168	VBLANK read failure	hardware - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.

169 to 254

Error Messsage #	Message	Explanation/Suggestions
169	Comm Ctrl failure Z8002 output latch	hardware - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
170	Pipeline register fault	hardware - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.
171 to 191	RESERVED	---
192 to 223	Powerup error: memory unit test	self-test - check the cabling (see Installation Guide) - if error persists, write down the circumstances under which it occurred - contact Raster Technologies or local rep.
224 to 254	Powerup error: bipolar processor test	self-test - Bipolar Processor problem (on GPU board) - write down circumstances under which it occurred - contact Raster Technologies or local rep.

Raster Technologies

Model One/10

COMMAND REFERENCE

Revision 1.1

February 10, 1986

Part # 552-00030-001

Raster Technologies, Inc.
Two Robbins Road
Westford, MA 01886
(617) 692-7900

RASTER TECHNOLOGIES
MODEL ONE

This document corresonds to Version 2.1 of the Model One/10 firmware.

NOTICE:

The information contained in this document is subject to change without notice.

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual may cause interference with to radio commmunications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

INTRODUCTION TO THE MODEL ONE/10 COMMAND REFERENCE

Purpose

The Model One/10 Command Reference describes how to use each of the Model One/10 commands.

The Command Reference is intended to be a reference guide for users who have a basic understanding of the Model One/10's capabilities. Refer to the Model One/10 Introduction and Installation Guide for a more comprehensive discussion of the Model One/10's features and how the commands are interrelated.

The Model One/80 and Model One/10 Command Reference Card 2.0 provides a quick reference for the command set.

Information Provided for Each Command

This Command Reference provides the same types of information for each command, in the same basic format. The following sections are included in each command description.

Note: The format is expanded for the SYSCFG and the XFORM2D commands, which involve several options.

SYNTAX

This section lists the ASCII syntax, the FORTRAN call, and binary stream.

FUNCTION

The Function section explains the function of the command, including what it does, the reasons for using it, and its variations.

ASCII PARAMETERS

The ASCII Parameters section lists and describes each parameter. The description specifies the range for the parameter, the default value (if any), the input formats, and any other necessary information.

INTRODUCTION TO THE MODEL ONE/10 COMMAND REFERENCE, continued

FORTRAN PARAMETERS

The FORTRAN Parameters section lists the type declaration for parameters in the FORTRAN subroutine. For subroutines which can accept parameters of different types, this section also notes how the parameters should be declared in the application program.

This section does not describe each parameter in detail. You can refer to the ASCII Parameters section for more detailed information.

EXAMPLE

For most commands, an example is provided to illustrate how the command is used.

Note: These examples assume that the default conditions are in effect.

List of Mnemonics

You can use ASCII mnemonics for coordinate registers (CREGs) 0 through 6, 9, and 10. You can also use mnemonics for value registers (VREGs) 0 through 3.

A listing of these mnemonics is included at the end of this Command Reference.

Compatibility With Other Model Ones

The following commands appear on the Model One/10 Help Screen, but are only included for compatiblity with other Model Ones. The following commands do not cause an error on the Model One/10, but do not perform any action.

BOLD	GETWIN	OVRSTK	SCROLL	UPDNX
DEFWIN	HDRAW	OVRVAL	SELWIN	UPDNY
DELWIN	HOME	OVRZM	SETCLASS	VGWAIT
DIRCUR	LUUTRTE	PATFIL	SETCUR	WRAP
FIRSTP	MODE1K	PIXCLP	SETSIZ	ZOOM
GETCUR	MOVCUR	PIXFUN	SETTYPE	ZOOMIN
GETPOS	OVRRD	PMCTL	TEST2	

MODEL ONE/10 COMMANDS BY FUNCTIONAL GROUPS

Display Control

BLANK	MODDIS	RMSK16	WMSK16
CURSOR	PIXCLP	VECPAT	WRMASK
FIRSTP	RDMASK	VIDFORM	XHAIR
HDCOPY	RGBTRU	WINDOW	ZOOM
			ZOOMIN

Display List

DELPID	PUSH	SEGDEF	SETATR
EXMODE	RDPICK	SEGDEL	SETGL
IGNORE	RDPID	SEGEND	SYSTAT
INCPID	RDREG	SEGINI	XFORM2D
INSPID	RDXFORM	SEGINQ	XMOVE
PICKID	SEGAPP	SEGREF	
POP	SEGCOP	SEGREN	

Graphics Primitives

ARC	DRWI	POLYGN	TEXTRE
AREA1	DRWREL	PRMFIL	VAL8
AREA2	FILMSK	RECREL	VALUE
CIRCI	FLOOD	RECTAN	VTEXT1
CIRCLE	MOV2R	RECTI	VTEXT2
CIRCXY	MOV3R	TEXT1	
CLEAR	MOVABS	TEXT2	
DRW2R	MOVI	TEXTC	
DRW3R	MOVREL	TEXTDN	
DRWABS	POINT	TEXTN	

Image Transmission

PIXEL8	PIXMOV
PIXELS	RUNLEN
PIXL16	RUNLN8
PIXFUN	

MODEL ONE/10 COMMANDS BY FUNCTIONAL GROUPS, continued

Interactive Device Support

BUTTBL
BUTTON
FLUSH
XYDIG

Look-Up Table

BLINKC	LUTA
BLINKD	LUTB
BLINKE	LUTG
BLINKR	LUTR
LUT8	LUTRMP
LUT16	

Macro Programming

MACDEF	MACERA
MACEND	MACRO

Readback

RDMODE	READF
RDPIXR	READP
READBU	READVR
READCR	READW
READER	READWE

Register Operations

CADD	SCRORG
CLOAD	VADD
CMOVE	VLOAD
CORORG	VMOVE
CSUB	VSUB

MODEL ONE/10 COMMANDS BY FUNCTIONAL GROUPS, continued

Software Development

*	
ALPHAO	NULL
DEBUG	REPLAY
DELAY	WAIT
HELP	

System Configuration

ASCII	SAVCFG
COLD	SPCHAR
DFTCFG	SYSCFG
DISCFG	TEKEM
QUIT	

SYNTAX

ASCII	ALPHAO string
FORTRAN Call	CALL ALPHAO (STRLEN, STRING)
Binary	[180] [strlen] ([char1] [char2]...[charn])
	180 decimal = 264 octal = B4 hex

FUNCTION

The ALPHAO (ALPHA Output) command outputs a text string on the local alphanumeric display screen. The text to be output is specified by string. If the command is being entered in ASCII mode from the local alphanumeric terminal or keyboard, string is the set of ASCII characters remaining on the command line. If the command is not being sent in ASCII mode, then strlen must be given. Strlen contains the number of characters in the string followed by a string with strlen bytes.

Nonprintable characters can be included in ALPHAO text strings. Characters with the high bit set or certain control characters (e.g., CTRL-S) can be put into string arguments in ASCII mode by using the backslash (\) to indicate that the immediately following characters are the numeric value of the desired character:

\ddd	Indicates 3 decimal digits representing the value of the desired character.
\#hh	Indicates the 2 hexadecimal digits representing the desired character.
\\	Indicates the backslash character (\) itself.

ASCII PARAMETER

string	The text to be printed.

FORTRAN PARAMETERS

INTEGER*2 STRLEN, STRING(1)
STRLEN is an integer specifying the number of characters that are to be output.

STRING is an integer array with two characters packed per 16-bit word, as in FORTRAN A2 format.

EXAMPLE

```
! ALPHAO ABCDEF 1 2 3          ; Output text string "ABCDEF 1 2 3".
! ALPHAO WXYZ                  ; Output text string "WXYZ".
```

FORTRAN EXAMPLE

```
CRLF = CHAR(10)//CHAR(12)
CALL ALPHAO(10,'12345678'//CRLF)
```

SYNTAX

ASCII	ALPHEM flag
FORTRAN Call	CALL ALPHEM (FLAG)
Binary	[194] [flag] (2 bytes)
	194 decimal = 302 octal = C2 hex

FUNCTION

The ALPHEM command turns the overlay planes on or off. The ALPHEM command can also be used to output text from the screen to the printer.

Flag = 0 or OFF turns off overlay planes and inhibits writing to overlay planes. Flag = 1 or ON displays overlay planes and allows writing to overlay planes. Flag = 2 or TOGGLE invokes the opposite state from the current state (e.g., if ALPHEM is currently set to ON, ALPHEM TOGGLE switches ALPHEM to OFF).

The command ALPHEM PRNTON causes any text on the screen to be output to the printer. To turn off this mode, use the command ALPHEM PRNTOFF.

ASCII PARAMETER

flag	Flag = 0 or OFF turns off overlay planes and inhibits writing to overlay planes
	flag = 1 or ON displays overlay planes and allows writing to overlay planes
	flag = 2 or TOGGLE invokes opposite state from current state
	flag = 3 or PRNTOFF inhibits the output of text on the screen to the printer
	flag = 4 or PRNTON causes any text on the screen to be output to the printer

FORTRAN PARAMETER

INTEGER*2 FLAG

EXAMPLE

```
! MOVABS 0 0            ; Move current point to 0,0.
! LUT8 1 255 0 0        ; Change the color out for LUT index 1 to red.
! VAL8 1                ; Set current pixel value to 1 (255,0,0: red).
! RECTAN 50 50          ; Draw a red square.
! MOVABS 100 100        ; Move current point to 100,100.
! LUT8 16 0 255 0       ; Change the color out for LUT table index 16 to
                          green.
! VAL8 16               ; Set current pixel value to 16 (0,255,0: green).
! CIRCLE 25             ; Draw green circle.
! ALPHEM 0              ; Turn off text overlay plane; text disappears.
! ALPHEM 2              ; Toggle ALPHEM, turning text overlay plane back
                          on.
```

ARC

SYNTAX

ASCII ARC rad, al, a2

FORTRAN Call CALL ARC (IRAD, IA1, IA2)

Binary

[17] [highrad][lowrad] [highal][lowal] [higha2][lowa2]

17 decimal = 021 octal = 11 hex

FUNCTION

The ARC command draws a circular arc with its center at the WCS current point (CREG 0), a radius of rad, the starting angle al, and ending angle a2. The angles are specified in integer degrees measured counter-clockwise. An angle of 0 degrees is the positive x axis. An angle of 90 degrees is the positive y axis. The arc is drawn counter-clockwise from the starting angle to the ending angle.

ASCII PARAMETERS

rad	16-bit integer value specifying the radius of the arc; range is -32,768 to +32,767.
al	16-bit integer value specifying the starting angle; range is -32,768 to +32,767.
a2	16-bit integer value specifying the ending angle; range is -32,768 to +32,767.

Note: Angle values used in the ARC command should normally be in the range of -360 to 360. Much larger values may cause the system to hang.

FORTRAN PARAMETERS

INTEGER*2 IRAD, IA1, IA2

EXAMPLE

```
! MOVABS 0 0          ; Move current point to location  0,0.
! ARC 75 45 135       ; Draw circular arc of radius 75, starting
                        at 45 degrees and ending at 135 degrees.
! ARC 100 -30 60      ; Draw circular arc of radius 100, starting
                        at 30 degrees and ending at 60 degrees.
! PRMFIL ON           ; Select filled primitives.
! ARC 40 -10 40       ; Draw filled (pie shape) arc of radius 40,
                        starting at -10 degrees, ending at 40 degrees.
```

SYNTAX

ASCII	AREA1
FORTRAN Call	CALL AREA1
Binary	[19] (1 byte)
	19 decimal = 023 octal = 13 hex

FUNCTION

The AREA1 command performs area filling. AREA1 sets all pixels in a given closed region to the current value (VREG 0). The region extends from the WCS current point (CREG 0) outward in all directions until a boundary whose pixel values differ from the pixel value at the current point is reached. The boundary pixel values and the original pixel values are ANDed with the fill mask (VREG 3) before the comparison is made. The fill mask is set by the FILMSK command.

Note: Transformations (XFORM2D) and picking (PICKID) should not be used with area filled primitives.

EXAMPLE

```
! LUT8 1 255,0,0        ; Set the color out for LUT index 1 to red.
! LUT8 2 255,255,255    ; Set the color out for LUT index 2 to white.
! LUT8 3 0,255,0        ; Set the color out for LUT index 3 to green.
! VAL8 1                ; Set current pixel value to 1 (red).
! CIRCLE 30             ; Draw red circle of radius 30.
! MOVABS 25 20          ; Move current point to 25,20.
! CIRCLE 35             ; Draw red circle of radius 35.
! VAL8 2                ; Set current pixel value to 2 (white).
! AREA1                 ; Begin area fill in white from 25,20 outward to
                          boundary.
! MOVABS 10 -10         ; Move current point to 10,-10.
! VAL8 3                ; Set current pixel value to 3 (green).
! AREA1                 ; Begin area fill in green from 10,-10 outward to
                          boundary (the intersection of the two circles).
```

SYNTAX

ASCII	AREA2 vreg
FORTRAN Call	CALL AREA2 (IVREG)
Binary	[20] [vreg] (2 bytes)
	20 decimal = 024 octal = 14 hex

FUNCTION

The AREA2 command performs area filling. AREA2 sets all pixels in a given closed region to the current value (VREG 0). The region extends from the WCS current point (CREG 0) outward until a boundary of pixels whose value is specified by value register vreg or VREG 0 is reached. The boundary pixel values and the value specified by value register vreg are ANDed with the fill mask (VREG 3) before the comparison is made. The fill mask is set by the FILMSK command.

Note: Transformations (XFORM2D) and picking (PICKID) should not be used with area filled primitives.

ASCII PARAMETER

vreg	The value register containing the boundary pixel value; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETER

INTEGER*2 IVREG

EXAMPLE

```
! LUT8 1 255,0,0         ; Set the color out for LUT index 1 to red.
! LUT8 2 0,0,255         ; Set the color out for LUT index 2 to blue.
! LUT8 3 0,255,255       ; Set the color out for LUT index 3 to cyan.
! VAL8 1                 ; Set current pixel value to 1 (red).
! CIRCLE 100             ; Draw red circle of radius 100.
! VAL8 2                 ; Set the current pixel value to 2 (blue).
! CIRCLE 50              ; Draw blue circle of radius 50.
! VAL8 3                 ; Set current pixel value to 3 (cyan).
! VLOAD 9 3,0,0          ; Load VREG 9 with 3,0,0.
! AREA2 9                ; Begin area fill in cyan. Boundary pixel value
                           is in VREG 9. (The inner blue circle is
                           overwritten because it is not drawn in boundary
                           pixel value.)
```

ASCII

SYNTAX

ASCII	ASCII flag
FORTRAN Call	CALL ASCII (IFLAG)
Binary	[155] [flag] (2 bytes)
	155 decimal = 233 octal = 9B hex

FUNCTION

The ASCII command sets the host input port. If flag = 0 or OFF, the host port input command stream is interpreted as pure ASCII format; all subsequent commands must be sent from the host to the Model One exactly as they would be typed in locally. If flag = 0 or OFF, the host port input command stream is interpreted as the default 8-bit binary or ASCII hex, as set with the SYSCFG command.

You never use the ASCII command in local mode. The ASCII command is usually sent from the host as a binary character string sequence.

Pure ASCII format requires many more characters to be sent from the host to execute a series of commands and should be used only when the command stream must be directly interpreted by the programmer or user rather than the Model One.

PARAMETER

flag	Flag = 1 or ON sets host port to pure ASCII format; flag = 0 or OFF (default) sets host port to binary or ASCII hex.

FORTRAN PARAMETER

INTEGER*2	IFLAG

* (ASTERISK)

SYNTAX *

FUNCTION

The * (asterisk) command allows use of program comments when in pure ASCII format mode, as set with the ASCII command. Any characters following the asterisk (and before the carriage return) are ignored.

EXAMPLE

```
!*              ; This is a comment which is ignored.
```

SYNTAX

ASCII	BLANK flag	
FORTRAN Call	CALL BLANK (IFLAG)	
Binary	[49] [flag]	(2 bytes)
	49 decimal = 061 octal = 31 hex	

FUNCTION

The BLANK command allows you to blank or unblank the screen. If flag = 1 or ON, the screen is blanked, and the contents of image memory are no longer displayed.

ASCII PARAMETER

flag	Flag = 1 or ON blanks the screen; flag = 0 or OFF restores normal video.

FORTRAN PARAMETER

INTEGER*2 IFLAG

EXAMPLE

```
! PRMFIL ON          ; Select filled primitives.
! CIRCLE 50          ; Draw circle of radius 50.
! BLANK 1            ; Blank screen.
! CIRCLE 100         ; Draw circle of radius 100.
! BLANK 0            ; Restore normal video.
```

BLINKC BLINKC

SYNTAX

ASCII	BLINKC
FORTRAN Call	CALL BLINKC
Binary	[35] (1 byte)
	35 decimal = 043 octal = 23 hex

FUNCTION

The BLINKC command clears the blink table, and disables blink of all LUT locations. The LUT entries are set to entry1 of their blink values, as set with the BLINKE command.

SYNTAX

ASCII	BLINKD lut, index
FORTRAN Call	CALL BLINKD (LUT, INDEX)
Binary	[33] [lut] [index] (3 bytes)
	33 decimal = 041 octal = 21 hex

FUNCTION

The BLINKD command disables blinking of the LUT location specified by lut. The index gives the pixel value in image memory that will address this location in the LUT.

BLINKD does not affect LUT entries from 256 to 1024.

Caution: The color that is left in the LUT location as a result of the BLINKD command is undefined. Therefore, you should use the appropriate LUT command to set the desired color.

ASCII PARAMETERS

lut	lut=1, disable the blue component. lut=2, disable the green component. lut=4, disable the red component. lut=7, disable all components. Note: Options are additive. For example, lut=6 disables the green and red components.
index	LUT index location; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2	LUT, INDEX

BLINKE

SYNTAX

ASCII	BLINKE lut, index, entry1, entry2
FORTRAN Call	CALL BLINKE (LUT, INDEX, IENT1, IENT2)
Binary	[32] [lut] [index] [entry1] [entry2]
	32 decimal = 040 octal = 20 hex

FUNCTION

The BLINKE command enables blinking of the LUT location specified by lut. The index specifies the address of the location in the LUT to blink. Entry1 and entry2 are swapped back and forth at the rate specified by the BLINKR command.

BLINKE does not affect LUT entries from 256 to 1024.

PARAMETERS

lut	lut=1, enable blue component.
	lut=2, enable green component.
	lut=4, enable red component.
	lut=7, enable all components.
	
	Note: Options are additive. For example, lut=6 enables the green and red components.

index	LUT index location; range is 0 to 255.
entry1	First LUT entry to blink between; range is 0 to 255.
entry2	Second LUT entry to blink between; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 LUT, INDEX, IENT1, IENT2

EXAMPLE

```
! BLINKE 7 63 255 125        ; Blink location 63 in all LUT components,
                               swapping between entries 255 and 125.
! BLINKC                     ; Location 63 has value 255 in all LUT
                               components.
! BLINKE 7 63 100 255        ; Blink between entries 100 and 255.
! BLINKC                     ; Location 63 has value 100 in all LUT
                               components.
```

BLINKR BLINKR

SYNTAX

ASCII	BLINKR frames
FORTRAN Call	CALL BLINKR (FRAMES)
Binary	[34] [frames] (2 bytes)
	34 = 042 octal = 22 hex

FUNCTION

The BLINKR command sets the blink rate to frames frame times. This rate is specified as the number of frame times between swapping between the two LUT entries. One frame time is 1/60 second.

ASCII PARAMETER

frames	Each frame time is 1/60 second; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 FRAMES

SYNTAX

ASCII	BUTTBL number, macnum
FORTRAN Call	CALL BUTTBL (INDEX, MACNUM)
Binary	[170] [number] [macnum] (3 bytes)
	170 decimal = 252 octal = AA hex

FUNCTION

The BUTTBL command is used to load the Button Table. Number gives the button number. Macnum gives the macro number to execute when the button is pressed (for positive number) or released (for negative number).

Buttons can be found on the Raster Technologies keyboard and on locator devices such as a tablet or mouse. The number of buttons available is a function of the type of device used. A button can always be generated by using the BUTTON command.

ASCII PARAMETERS

number	Button number; range is -63 to 63.
macnum	Macro number; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 INDEX, MACNUM

EXAMPLE

```
! MACDEF 37          ; Start definition of Macro 37.
$ CIRCLE 50          ; Draw a circle.
$ FLUSH              ; Flush the button queue.
$ MACEND             ; End macro definition.
! BUTTBL 13 37       ; Execute Macro 37 when Button 13 is pressed.
```

SYNTAX

ASCII	BUTTON index
FORTRAN Call	CALL BUTTON (INDEX)
Binary	[171] [index] (2 bytes)
	171 decimal = 253 octal = AB hex

FUNCTION

The BUTTON command executes the macro specified by the Button Table at location index. This command performs the same function as pressing function button index on the Raster Technologies keyboard or on locator devices such as a tablet or a mouse.

Negative buton index values indicate that the button macro will be executed when the buton is released rather than when the buton is pressed.

ASCII PARAMETER

index	Button number; range is -64 to 63.

FORTRAN PARAMETER

INTEGER*2	INDEX

EXAMPLE

```
! MACDEF 15         ; Begin definition of Macro 15.
$ CIRCLE 50         ; Draw a circle.
$ MACEND            ; End Macro definition.
! BUTTBL 21 15      ; Execute Macro 15 when Button 21 is pressed.
! BUTTON 21         ; Simulate having pushed Button 21.
```

SYNTAX

ASCII	CADD csum, creg
FORTRAN Call	CALL CADD (ICSUM, ICREG)
Binary	[162] [csum] [creg]
	162 decimal = 242 octal = A2 hex

FUNCTION

The CADD command adds the contents of the coordinate register specified by creg to the contents of the coordinate register specified by csum and places the result in the coordinate register specified by csum.

ASCII PARAMETERS

csum, creg	Coordinate registers for addition; range is 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

INTEGER*2	ICSUM, ICREG

EXAMPLE

```
! CLOAD 25 100 150       ; Load CREG 25 with 100,150.
! CLOAD 26 10 20         ; Load CREG 26 with 10,20.
! CADD 25 26             ; Add the contents of CREG 26 to CREG 25
                           and place result in CREG 25.
! READCR 25              ; Read the contents of CREG 25
  00110 00170              (Response from Model One.)
! CADD 25 26             ; Add the contents of CREG 26 to CREG 25
                           and place result in CREG 25.
! READCR 25              ; Read contents of CREG 25
  00120 00190              (Response from Model One.)
```

CIRCI

SYNTAX

ASCII	CIRCI creg
FORTRAN Call	CALL CIRCI (ICREG)
Binary	[16] [creg] (2 bytes)
	16 decimal = 020 octal = 10 hex

FUNCTION

The CIRCI command draws a circle with the center point of the circle at the WCS current point (CREG 0) and the point specified by coordinate register creg on the circumference of the circle. The CIRCI command is useful for drawing circles with the radius controlled by an interactive device such as the digitizing tablet.

ASCII PARAMETER

creg	Coordinate register for point on circumference; range is 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETER

INTEGER*2 ICREG

EXAMPLE

```
! MOVABS 100 100       ; Move to location 100,100.
! CIRCI 4              ; Draw circle whose center is 100,100 and
                         whose circumference includes the location
                         given in CREG 4.
! MOVABS 120 150       ; Move to location 120,150.
! CLOAD 27 200 220     ; Draw circle whose center is at 120,150 and
                         whose circumference includes the location
                         given in CREG 27 (200,220).
```

CIRCLE CIRCLE

SYNTAX

ASCII	CIRCLE rad
FORTRAN Call	CALL CIRCLE (IRAD)
Binary	[14] [highrad] [lowrad] (3 bytes)
	14 decimal = 16 octal = 0E hex

FUNCTION

The CIRCLE command draws a circle of radius rad, with the center at the WCS current point (CREG 0). The circle is drawn in the current pixel value. A circle of radius zero sets the current point to the current pixel value.

If the circle is transformed, it is drawn as a circle, with the radius defined so that the point represented by the coordinates (current-point-X plus rad, Y) is on the circumference.

ASCII PARAMETER

rad	16-bit integer value specifying the circle radius; range is -32,768 to 32,767.

FORTRAN PARAMETER

INTEGER*2	IRAD

EXAMPLE

```
! MOVABS 100 150          ; Move current point to 100,150.
! CIRCLE 30               ; Draw circle of radius 30 centered at 100,150.
! MOVREL 10 0             ; Move current point by 10,0 to 110,150.
! CIRCLE 20               ; Draw circle of radius 20 centered at 110,150.
! CIRCLE 10               ; Draw circle of radius 10 centered at 110,150.
```

CIRCXY CIRCXY

SYNTAX

ASCII	CIRCXY x, y
FORTRAN Call	CALL CIRCXY (IX, IY)
Binary	[15] [highx][lowx] [highy][lowy] (5 bytes)
	15 decimal = 017 octal = 0F hex

FUNCTION

The CIRCXY command draws a circle with the center of the circle at the WCS current point (CREG 0) and the point x,y on its circumference.

If the circle is transformed, it is drawn as a circle, with the circumference point interpreted according to the current transformation.

ASCII PARAMETERS

x, y	16-bit integer values specifying the coordinates for the point on the circumference; range is - 32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2 IX, IY

EXAMPLE

```
! MOVABS 0 0            ; Move current point to 0,0.
! CIRCXY 30 30          ; Draw circle centered at 0,0 with 30,30 on
                          its circumference.
! CIRCXY 30 40          ; Draw circle centered at 0,0 with 30,40 on
                          its circumference.
! CIRCXY 50 50          ; Draw circle centered at 0,0 with 50,50 on
                          its circumference.
```

SYNTAX

ASCII	CLEAR
FORTRAN Call	CALL CLEAR
Binary	[135] (1 byte)
	135 decimal = 207 octal = 87 hex

FUNCTION

The CLEAR command changes all pixels in the currently defined window to the current pixel value (VREG 0). Pixels outside the current clipping window are unchanged. The corners of the current window are held in CREGs 9 and 10.

CLEAR command uses the vector pattern register.

EXAMPLE

```
! WINDOW -50 -50 50 50  ; Set current clipping window to rectangle at
                          center of screen.
! CLEAR                 ; Clear current window to current pixel value
                          (default white).
! LUT8 3 0,0,255        ; Set the color out for LUT index 3 to blue.
! VAL8 3                ; Set current pixel value to 3 (blue).
! CLEAR                 ; Clear current window to current pixel value;
                          screen turns blue.
```

SYNTAX

ASCII	CLOAD creg, x, y
FORTRAN Call	CALL CLOAD (ICREG, IX, IY)
Binary	[160] [creg] [highx][lowx] [highy][lowy] (6 bytes)
	160 decimal = 240 octal = A0 hex

FUNCTION

The CLOAD command loads the coordinate register specified by creg with the values specified by x, y.

This command should not be used with CREG 3 (the coordinate origin).

ASCII PARAMETERS

creg	Coordinate register to be loaded; range is 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.
x, y	16-bit integer values specifying the (x, y) coordinate pair to be loaded into the specified coordinate register; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2 ICREG, IX, IY

EXAMPLE

```
! CLOAD 17 100 150       ; Load CREG 17 with 100,150.
! READCR 17              ; Read contents of CREG 17.
  00100 00150              (Response from Model One.)
! CLOAD 17 50 -50        ; Load CREG 17 with 50,-50.
! READCR 17              ; Read contents of CREG 17.
  00050 -00050             (Response from Model One.)
```

SYNTAX

ASCII	CMOVE cdst, csrc	
FORTRAN Call	CALL CMOVE (ICDST, ICSRC)	
Binary	[161] [cdst] [csrc]	(3 bytes)
	161 decimal = 241 octal = A1 hex	

FUNCTION

The CMOVE command copies the contents of the coordinate register specified by csrc to the coordinate register specified by cdst. Any data in cdst is replaced by the new data.

This command should not be used with CREG 3 (the coordinate origin) as the cdst register.

ASCII PARAMETERS

cdst, csrc	Coordinate registers; range is 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

INTEGER*2	ICDST, ICSRC

EXAMPLE

```
! CLOAD 25 100,150        ; Load CREG 25 with 100,150.
! CLOAD 26 20,-50         ; Load CREG 26 with 20,-50.
! READCR 26               ; Read contents of CREG 26.
  00020 -00050              (Response from Model One.)
! CMOVE 26 25             ; Move contents of CREG 25 into CREG 26.
! READCR 26               ; Read contents of CREG 26.
  00100 00150               (Response from Model One.)
```

COLD

SYNTAX

ASCII	COLD
FORTRAN Call	CALL COLD
Binary	[253]
	253 decimal = 375 octal = FD hex

FUNCTION

The COLD command performs a Model One/10 COLDstart, equivalent to turning the POWER button on the front panel. COLD executes the Model One/10's restart sequence and diagnostics. The COLDstart state of the Model One/10 is defined as follows.

```
MODDIS 1                 ; Load default LUT.
CORORG 0,0               ; Set Coordinate Origin to 0,0.
SCRORG 0,0               ; Set Screen Origin to 0,0.
WINDOW x1,y1,x2,y2       ; Set clipping window to image memory bounds.
PRMFIL 0                 ; Select unfilled primitives.
MACERA 0 through
  MACERA 255             ; Erase all macros.
TEXTRE                   ; Erase all user defined fonts.
VAL8 0                   ; Set current pixel value to 0.
VECPAT #FFFF             ; Draw solid lines.
CLEAR                    ; Clear image memory to 0.
VAL8 127                 ; Set current pixel value to 127.
BUTTBL 63,0              ; Set all button table entries to zero.
DELAY 10                 ; Set delay between characters for readback to 10.
WMSK16 #FF00             ; Enable all bit planes for writing.
RMSK16 #FF00             ; Enable all bit planes for reading.
```

The Model One/10 is in ALPHA mode after the COLDstart.

Notes: If you want to reset the Model One/10 to a state other than the above, you will want to set up a standard command to execute after the COLDstart command.

In addition, if the COLD command is sent from the host, you must wait roughly three seconds before sending any more data.

You can use MODDIS 1 to reset without clearing the text overlay planes. This is useful for an interim view display so that VT100 data is not lost.

EXAMPLE

```
! COLD                    ; COLDstart the Model One/10.
```

SYNTAX

ASCII	CORORG x, y
FORTRAN Call	CALL CORORG (X, Y)
Binary	[55] [highx] [lowx] [highy] [lowy] (5 bytes)
	55 decimal = 067 octal = 37 hex

FUNCTION

The CORORG command sets the coordinate origin register (CREG 3) to the point specified by x,y. The contents of this register are added to all incoming coordinates; all coordinates are relative displacements from the coordinate origin. The CORORG command resets all other coordinate registers and should be preceded by a COLDstart.

The CORORG command should be used in conjunction with the SCRORG command to maintain the coordinate origin at the center of the screen.

After a CORORG command, CREG 3 will contain the cumulative offset from the original origin after COLDstart (0, 0). To return to the original CORORG, execute a CORORG command with values set opposite:

```
READCR 3
  00100 -00100
CORORG -00100 00100
```

Note: Use a WAIT 0 when CORORG immediately follows a MODDIS, if the commands are issued from the host.

ASCII PARAMETERS

x, y	16-bit integer values specifying the new coordinate origin; range is -32,768 to +32,767.

FORTRAN PARAMETERS

INTEGER*2	X, Y

EXAMPLE

```
! COLD                ; Perform a COLDstart.
! CORORG 100 -100     ; Set coordinate origin register (CREG3) to
                        100,-100 (text jumps to the top of the screen).
! CIRCLE 50           ; Draw a circle (it appears in the middle of the
                        screen).
! SCRORG -100 100     ; Set screen origin in conjunction with the
                        change you made in the coordinate origin;
                        circle shifts to the lower right corner, text
                        appears in the upper left corner.
! READCR 3            ; Read back CREG 3 (coordinate origin).

  00100  -00100       ; Model One response.

! CORORG -100 100     ; Reset coordinate origin to orignal position;
                        circle goes further to the bottom left corner.
! READCR 3            ; Read back CREG 3 (coordinate origin).

  00000  00000        ; Model One response (coordinate origin returns
                        to 0,0, the default).
! SCRORG 100 -100     ; Reset screen origin to original position; circle
                        returns to the middle of the screen.
```

SYNTAX

ASCII	CSUB cdif, creg
FORTRAN Call	CALL CSUB (ICDIF, ICREG)
Binary	[163] [cdif] [creg] (3 bytes)
	163 decimal = 243 octal = A3 hex

FUNCTION

The CSUB command subtracts the contents of the coordinate register specified by creg from the contents of the coordinate register specified by cdif and places the result in the coordinate register specified by cdif.

This command should not be used with CREG 3 (the coordinate origin) as the cdif register.

ASCII PARAMETERS

cdif, creg	Coordinate registers; range is from 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

INTEGER*2	ICDIF, ICREG

EXAMPLE

```
! CLOAD 20 100 150      ; Load CREG 20 with 100,150.
! CLOAD 21 25 30        ; Load CREG 21 with 25,30.
! CSUB 20 21            ; Subtract the contents of CREG 21 from
                          CREG 20 and place result in CREG 20.
! READCR 20             ; Read contents of CREG 20.
  00075 00120             (Response from Model One.)
```

DEBUG

SYNTAX

ASCII	DEBUG flag
FORTRAN Call	CALL DEBUG (FLAG)
Binary	[168] [flag] (2 bytes)
	168 decimal = 250 octal = A8 hex

FUNCTION

The DEBUG command is used to enter and exit the command stream translator. When flag = 1 or ON, the central processor displays, in mnemonic form, the commands that are being executed by the command interpreter. If flag = 0, DEBUG is disabled.

ASCII PARAMETER

flag	Flag = 1 or ON enables Command Stream Translator; flag = 0 or OFF disables Command Stream Translator.

FORTRAN PARAMETER

INTEGER*2	FLAG

SYNTAX

ASCII	DELAY amount
FORTRAN Call	CALL DELAY (IAMT)
Binary	[182] [amount] (2 bytes)
	182 decimal = 266 octal = B6 hex

FUNCTION

The DELAY command inserts a delay between characters when sending readback data to the host over the HOSTSIO interface. The DELAY command is necessary because some host computers cannot accept data as fast as the Model One/10 can send it, even at a given baud. This is particularly true at the higher baud rates.

Amount specifies the amount of time to insert between characters. The recommended amount for a VAX 11/7xx is 10 units. One unit of delay is approximately equal to 750 microseconds.

delay = 750 * amount microseconds

ASCII PARAMETER

amount	Amount of delay; range is 0 to 255. Default is 10.

FORTRAN PARAMETER

INTEGER*2	IAMT

DELPID

SYNTAX

ASCII	DELPID
FORTRAN Call	CALL DELPID
Binary	[236] (1 byte)
	236 decimal = 354 octal = EC hex

FUNCTION

The DELPID command deletes all commands with the current pick ID (stored in PIDREG) and the current segment ID (stored in SEGREG). The commands are deleted from the current pick ID to whichever comes first:

- the next pick ID, or
- the end of the segment.

The pick identification number, however, still exists within its segment. You can reopen the segment and append commands to change the segment's contents with the SEGAPP command.

If you have not just picked this pick ID, thus making it current, first set both registers, PIDREG and SEGREG, with the SETGL command or with the RDPICK command.

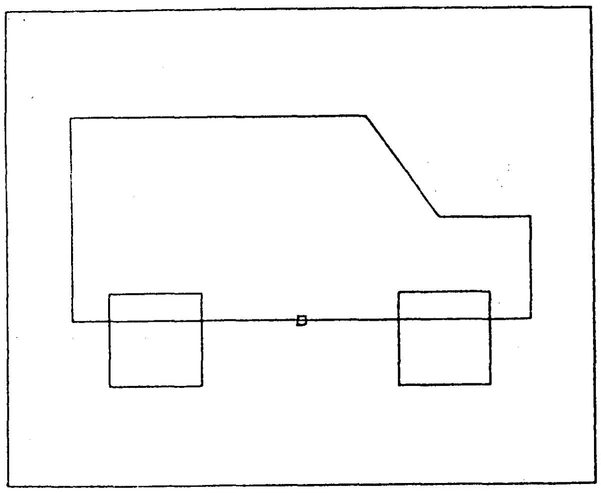
EXAMPLE

This example illustrates the use of the DELPID command. Segment 1 is already defined and contains two pick IDs. Pick ID 10 is the truck body and pick ID 20 is the chassis (bottom line) of the truck. Pick ID 20 contains two references to Segment 2, which is the definition of the wheel. Refer to the SEGDEF command for a complete definition of the truck.

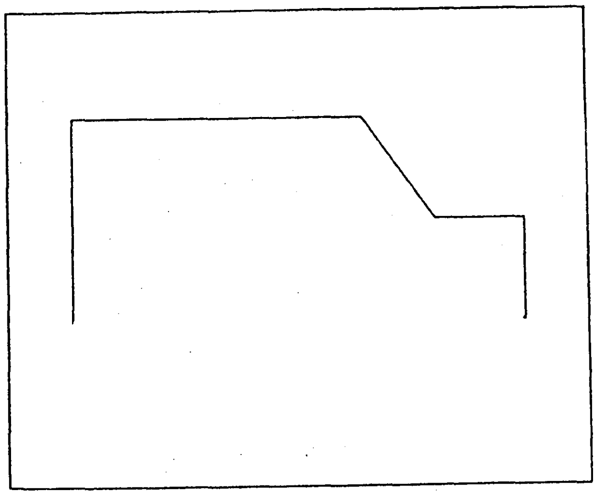
Note that Segment 2 is not deleted. However, pick ID 20 (which contains the references to Segment 2) is deleted. Therefore the wheels (Segment 2) are not drawn in this example after the DELPID command is executed.

```
! SEGREF 1                ; Execute (draw) Segment 1, truck with two wheels.
! CLOAD 19 0,-80          ; Set center of pick aperture.
! EXMODE PICK             ; Set execution mode to pick.
! SEGREF 1                ; Execute (pick) segment 1.
! RDREG 0                 ; Read back contents of segment ID and pick ID
                            registers.
 0000000001 0000000020    ; The Model One returns:  segment ID is 1, pick ID
                            is 20.
! EXMODE NORMAL           ; Restore normal draw execution mode.
! DELPID                  ; Delete all commands with pick ID 20 in Segment 1.
! VAL8 0                  ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1                ; Execute (draw) Segment 1, now truck body without
                            chassis or wheels.
```

BEFORE



AFTER



SYNTAX

ASCII DFTCFG

FUNCTION

The DFTCFG command restores all ports on the Model One/10 to the default configuration and sets the host serial communications mode to binary. In addition, it restores special characters and powerup status. The default configuration for the Model One/10 is:

PORT	RTS	CTS	STOP	BITS	XIN	XOUT	CTRL	PARITY	BAUD
TABLET	OFF	OFF	2	8	OFF	OFF	OFF	NONE	9600
HOST	OFF	OFF	1	8	OFF	ON	OFF	NONE	9600

The system configuration is modified through the use of the SYSCFG and SAVCFG commands. The SYSCFG command configures the Model One/10's ports; the SAVCFG command stores those configurations (which are then loaded whenever a COLDstart is performed).

The default configuration is not modified by SYSCFG or SAVCFG; the DFTCFG command should be used only when it is necessary to restore all the Model One/10's ports to a known state.

To display the current configuration, you can use the DISCFG command.

The DFTCFG command can be executed only from the local alphanumeric terminal, and cannot be included in a macro.

The DFTCFG command executes a WARMstart as part of the command, leaving the Model One/10 in ALPHA mode.

EXAMPLE

```
! SYSCFG SERIAL HOST XIN ON XOUT ON PARITY N

                            ; Configures port HOST as indicated.
                              (See SYSCFG for details.)
Are you sure??              ; Prompt from Model One.
y                           ; User response to prompt confirming that the
                              configuration is as desired.

  PORT     RTS   CTS   STOP   BITS   XIN   XOUT   CTRL   PARITY   BAUD

TABLET     OFF   OFF    2      8     OFF   OFF    OFF    NONE     9600
HOST       OFF   OFF    1      8     ON    ON     OFF    NONE     9600

                            ; Model One response (assuming configuration had
                              been set to default before example).  Graphics
                              mode must be re-entered (CTRL-D).
! SAVCFG                    ; Save the configuration of port HOST.

  PORT     RTS   CTS   STOP   BITS   XIN   XOUT   CTRL   PARITY   BAUD

TABLET     OFF   OFF    2      8     OFF   OFF    OFF    NONE     9600
HOST       OFF   OFF    1      8     ON    ON     OFF    NONE     9600

                            ; Model One redisplays the configuration, including
                              changes made.

Are you sure??              ; Prompt from Model One.
y                           ; User response to prompt confirming that the
                              configuration is as desired.

! DFTCFG                    ; Restore the default configuration for all
                              ports (not just port HOST).

Are you sure??              ; Prompt from Model One.
y                           ; User response to prompt confirming that the
                              configuration is as desired.  Graphics mode must
                              be re-entered (CTRL-D).
```

EXAMPLE, continued

```
! DISCFG                ; Display the current configuration.

  PORT     RTS   CTS   STOP   BITS   XIN   XOUT   CTRL   PARITY   BAUD

TABLET     OFF   OFF   2      8      OFF   OFF    OFF    NONE     9600
HOST       OFF   OFF   1      8      OFF   ON     OFF    NONE     9600

                        ; Note that host XIN is now set to OFF.
```

DIAG

SYNTAX

ASCII	DIAG test
FORTRAN Call	There is no FORTRAN call
Binary	[240] [test] (2 bytes)
	240 decimal = 360 octal = F0 hex

FUNCTION

The DIAG command can be used to run diagnostic tests or to align the monitor or to check on the current look-up table.

There are five diagnostic tests available.

Test	Description
DIAG 0	Displays a 16 x 16 monitor adjustment grid.
DIAG 1	Displays lower 256 Look-up table entries in a grid, going from 0 in the lower left corner to 255 in the upper right corner.
DIAG 2	Displays all Look-up table entries in four bars, going from 0 in the lower left corner to 1023 in the upper right corner.
DIAG 3	Increments pixel values starting at the bottom of the screen and moving up. This test runs until a key is pressed.
DIAG 255	Allows built in demos to be executed.

Note: To run the built in demos, do the following:

```
! DIAG 255
! MACRO 255
```

Then use the function keys to trigger the demos. A menu of options referring to these function keys will be displayed.

ASCII PARAMETER

test	The test number; acceptable values are 0 - 3 and 255 (see above).

DISCFG DISCFG

SYNTAX

ASCII DISCFG

FUNCTION

The DISCFG command displays the current configurations of the Model One/10's serial ports, as set with the SYSCFG command. The DISCFG command also displays the following information:

- the host mode (binary or ASCII hex)
- the ROM sequence number, and
- special characters.

Note: DISCFG command can be executed from the local terminal only, and may not be included in a macro.

DRW2R

SYNTAX

ASCII	DRW2R dx, dy	
FORTRAN Call	CALL DRW2R (DX, DY)	
Binary	[132] [dxdy]	(2 bytes)
	132 decimal = 204 octal = 84 hex	

FUNCTION

The DRW2R command is a two-byte version of the DRWREL command. The DRW2R command should only be called by the host in binary mode.

DRW2R draws a vector from the WCS current point (CREG 0) to the point specified by the (dx,dy) offset from the current point. The current point is changed to the end point after completion of the command. Only two bytes are required for the command. The range of dx and dy is -8 to 7, and dx,dy are specified as nibbles of a single byte.

ASCII PARAMETERS

dx, dy	The relative offset for the coordinate; range is -8 to 7.

FORTRAN PARAMETERS

INTEGER*2 DX, DY

The FORTRAN library manages the packing of the two numbers into a single byte.

EXAMPLE

Because nibbles are used to specify dx,dy, some examples may be useful:

DRW2R 17 draws (+1,+1): 17 = (16*1)+1 = 00010001

DRW2R -103 draws (-7,-7): -103 = 10011001

DRW2R -1 draws (-1,-1): -1 = 16*(NOT(1)+1)+(NOT(1)+1) = 11111111 = -1

The rules to generate this number are:

1. Negative numbers are encoded in binary as two's complement notation.

2. Positive binary is converted directly to decimal or hexadecimal.

3. Negative binary is converted to negative decimal by performing a two's complement binary weighting.

4. Negative binary is converted to hexadecimal by straight conversion on binary weighting and preceding both hex numbers with #FF.

DRW3R DRW3R

SYNTAX

ASCII	DRW3R dx, dy
FORTRAN Call	CALL DRW3R (IDX, IDY)
Binary	[131] [dx] [dy] (3 bytes)
	131 decimal = 203 octal = 83 hex

FUNCTION

The DRW3R command is a three byte form of the DRWREL command. The DRW3R command draws a vector from the WCS current point (CREG 0) to the point relative to the current point offset by dx and dy. The current point is changed to this new point. The range of dx and dy is -128 to 127. This command reduces the number of bytes which must pass between the Model One/10 and the host computer for vectors whose maximum displacement is within the given range.

ASCII PARAMETERS

dx, dy	The relative offset for the coordinate; range is -128 to 127.

FORTRAN PARAMETERS

INTEGER*2	IDX, IDY

DRWABS

SYNTAX

ASCII	DRWABS x, y
FORTRAN Call	CALL DRWABS (IX, IY)
Binary	[129] [highx] [lowx] [highy] [lowy] (5 bytes)
	129 decimal = 201 octal = 81 hex

FUNCTION

The DRWABS command draws a vector from the WCS current point (CREG 0) to the point specified by x,y and changes the current point (CREG 0) to x,y. The pixels along the line are drawn in the current pixel value (VREG 0).

Note: A vector with a length that exceeds 32,767 units will not be drawn correctly.

ASCII PARAMETERS

x, y	16-bit integers specifying the absolute x,y coordinate; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2	IX, IY

EXAMPLE

```
! MOVABS 50,50          ; Move current point to 50,50.
! DRWABS 60,50          ; Draw line to 60,50 (horizontal line 11 pixels
                          long, with both end point included).
! MOVABS 60,60          ; Move current point to 60,60.
! DRWABS 60,70          ; Draw line to 60,70 (vertical line 11 pixels
                          long).
! DRWABS 70 70          ; Draw line to 70,70 (diagonal line connected
                          to previous line).
! DRWABS 80 100         ; Draw line to 80,100.
```

DRWI

SYNTAX

ASCII	DRWI creg	
FORTRAN Call	CALL DRWI (ICREG)	
Binary	[133] [creg]	(2 bytes)
	133 decimal = 205 octal = 85 hex	

FUNCTION

The DRWI command draws a vector from the WCS current point (CREG 0) to the point given by coordinate register creg and changes the current point (CREG 0) to the new point.

Note: A vector with a length that exceeds 32,767 units will not be drawn correctly.

ASCII PARAMETER

creg — Coordinate register; range is 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETER

INTEGER*2 ICREG

EXAMPLE

```
! MOVABS -100 -50        ; Move current point to -100,-50.
! DRWI 4                 ; Draw vector from -100,-50 to location given
                           in CREG 4.
! MOVABS -30 -60         ; Move current point to -30,-60.
! CLOAD 33 100,150       ; Load CREG 33 with 100,150.
! DRWI 33                ; Draw vector from current point (-30,-60)
                           to location given in CREG 33 (100,150).
```

SYNTAX

ASCII	DRWREL dx, dy
FORTRAN Call	CALL DRWREL (IDX, IDY)
Binary	[130] [highdx][lowdx] [highdy][lowdy] (5 bytes)
	130 decimal = 202 octal = 82 hex

FUNCTION

The DRWREL command draws a vector from the WCS current point (CREG 0) to the point relative to the current point offset by dx and dy. The current point is set to the sum of the x-component of the old current point plus dx and the sum of the y-component of the old current point plus dy.

Note: A vector with a length that exceeds 32,767 units will not be drawn correctly.

ASCII PARAMETERS

dx, dy	16-bit integer values specifying the relative offset for the coordinate; range is -32,768 to 32,767.

FORTRAN PARAMETER

INTEGER*2	IDX, IDY

EXAMPLE

```
! MOVABS 50 30          ; Move to location 50,30.
! DRWREL 10 20          ; Draw line from 50,30 to 60,50.
! DRWREL 10 0           ; Draw line from 60,50 to 70,50.
! DRWREL 0 10           ; Draw line from 70,50 to 70,60.
```

SYNTAX

ASCII	EXMODE mode
FORTRAN Call	CALL EXMODE (MODE)
Binary	[123] [mode] (2 bytes)
	123 decimal = 173 octal = 7B hex

FUNCTION

The EXMODE command sets the display controller to the specified execution mode. The four execution modes are: NORMAL DRAW, PICK, HILITE, and UNHILITE.

If you are not already in pick mode, and you execute EXMODE PICK, the command also clears the pick buffer and calculates the pick aperture as set with the SETGL PICKAP command.

ASCII PARAMETER

mode Execution mode (8 bits).

0 = NORMAL DRAW	normal draw mode
1 = PICK	pick mode
2 = HILITE	highlight mode
3 = UNHILITE	unhighlight mode

FORTRAN PARAMETER

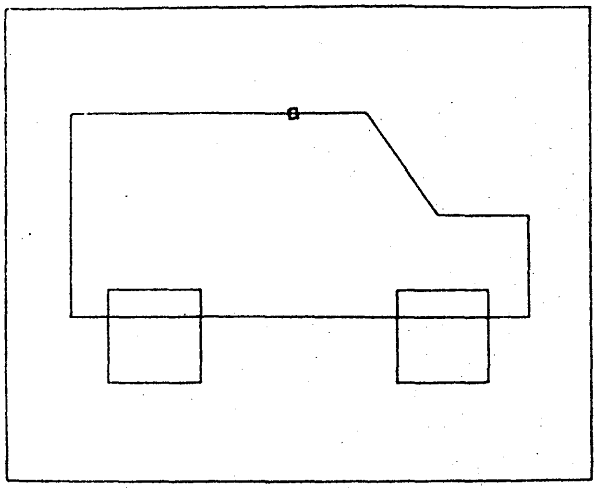
INTEGER*2 MODE

EXAMPLE 1

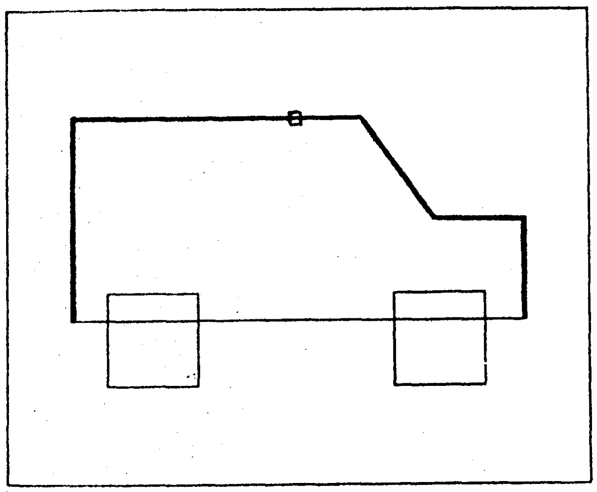
The following example illustrates the four different execution modes that you can set with the EXMODE command: normal draw, pick, highlight, and unhighlight. Segment 1 is already defined and contains two pick IDs. Pick ID 10 is the truck body and pick ID 20 is the chassis (bottom line) of the truck. Pick ID 20 contains two references to Segment 2, the wheel. Refer to the SEGDEF command for a complete definition of the truck.

```
! EXMODE NORMAL          ; Set execution mode to normal draw.
! SEGREF 1               ; Execute (draw) Segment 1.  (See the figure
                           labelled NORMAL DRAW.)
! SETGL PICKAP 8 8       ; Set the pick aperture size to 16 x 16.
! CLOAD 19 0,140         ; Specify the center of pick aperture. (See the
                           figures below.)
! VLOAD 45 48,0,0        ; Load VREG 45 with highlight color, red.
! EXMODE PICK            ; Set execution mode to pick.
! SEGREF 1               ; Execute (pick) Segment 1.
! RDREG 0                ; Read back contents of segment ID and pick ID
                           registers.
 0000000001 0000000010
                         ; The Model One returns: segment ID is 1, pick ID
                           is 10.
! EXMODE HILITE          ; Set execution mode to highlight.
! SEGREF 1               ; Execute Segment 1:  draw in red the geometry
                           labelled pick ID 10 in segment 1.  (See the
                           figure labelled HILITE.)
! EXMODE UNHILITE        ; Set execution mode to unhighlight.
! SEGREF 1               ; Execute Segment 1:  draw in normal color the
                           geometry that has been picked.
! EXMODE NORMAL          ; Resume normal draw mode.
```

NORMAL DRAW



HILITE



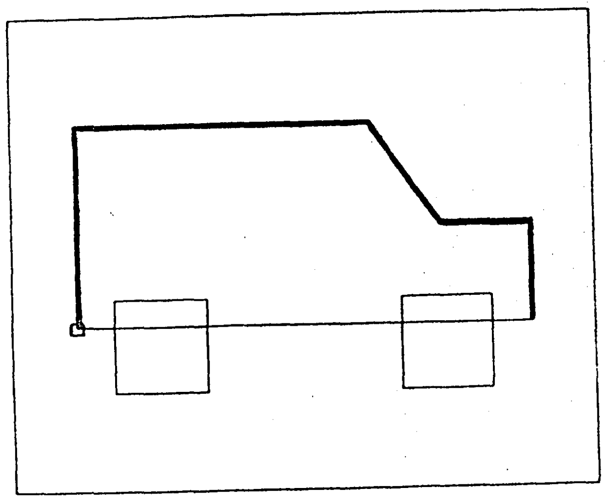
EXAMPLE 2

The following example illustrates a more complex case of picking and highlighting. In this case, the pick aperture is set at the intersection of the truck body (pick ID 10) and the chassis (pick ID 20). First we highlight the truck body (the first pick hit). Then we use the RDPICK command to update the pick ID and segment ID registers with the next pick hit (the chassis), and highlight the chassis in a different color.

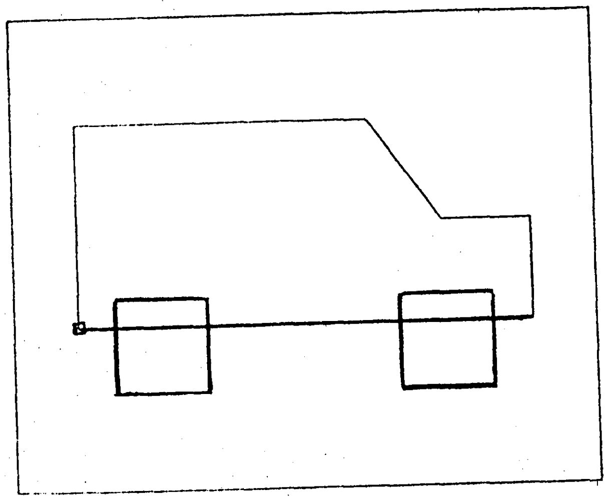
```
! CLOAD 19 -250,-80        ; Set center of pick aperture.  (See the figure on
                             the following page.)
! VLOAD 45 48,0,0          ; Load VREG 45 with highlight color, red.
! EXMODE PICK              ; Enter pick mode.
! SEGREF 1                 ; Execute (pick) Segment 1.
! RDPICK PICKS 0 0         ; Read back number of pick hits encountered and
                             number of pick hits recorded; registers are
                             not updated.
  00002 00002              ; The Model One returns:  two pick hits encountered
                             and recorded.
! RDREG 0                  ; Read back contents of segment ID and pick ID
                             registers.
 0000000001 0000000010     ; The Model One returns:  segment ID is 1, pick
                             ID is 10.  The registers contain the first pick
                             hit.
! EXMODE HILITE            ; Enter highlight mode.
! SEGREF 1                 ; Execute (highlight) Segment 1.  The truck body
                             (pick ID 10) is drawn in red.  (See the figure
                             labelled HILITE IN RED.)
! VLOAD 45 3,0,0           ; Load VREG 45 with new highlight color, blue.
! RDPICK SEGPID 1 2        ; Read back segment ID/pick ID pairs; update
                             registers with last entry read back.

 0000000001 0000000010 0000000001 0000000020

! RDREG 0                  ; Read back contents of segment ID and pick ID
                             registers.
 0000000001 0000000020     ; The Model One returns: segment ID is 1, pick ID
                             is 20.
! SEGREF 1                 ; Execute (highlight) Segment 1.  The chassis and
                             the two wheels are highlighted in blue.  (See
                             the figure labelled HILITE IN BLUE.)
! EXMODE NORMAL            ; Return to normal draw mode.
```

EXAMPLE 2, continued

HILITE IN RED



HILITE IN BLUE



FILMSK

SYNTAX

ASCII	FILMSK rmsk, gmsk, bmsk
FORTRAN Call	CALL FILMSK (IRMSK, IGMSK, IBMSK)
Binary	[159] [rmsk] [gmsk] [bmsk] (4 bytes)
	159 decimal = 237 octal = 9F hex

FUNCTION

The FILMSK command sets the fill mask (VREG 3) with the value specified by rmsk, gmsk, bmsk. The fill mask is ANDed with boundary values before the boundary check comparison is made in area fill commands (AREA1 and AREA2). The FILMSK command is equivalent to VLOAD 3 rmsk, gmsk, bmsk, as VREG 3 holds the fill mask.

The second and third bytes are ignored when using the fill mask. The fill mask does not affect bit planes 8 and 9.

ASCII PARAMETERS

rmsk	The red mask; range is 0 to 255.
gmsk	The green mask; range is 0 to 255. (Ignored)
bmsk	The blue mask; range is 0 to 255. (Ignored)

FORTRAN PARAMETERS

INTEGER*2	IRMSK, IGMSK, IBMSK

EXAMPLE

```
! PRMFIL ON              ; Select filled primitives.
! LUT8 1 255 0 0         ; Set color out for LUT index 1 to red.
! LUT8 2 0 255 0         ; Set color out for LUT index 2 to green.
! LUT8 3 0 0 255         ; Set color out for LUT index 3 to blue.
! LUT8 4 255 255 0       ; Set color out for LUT index 4 to yellow.
! LUT8 5 255 255 255     ; Set color out for LUT index 5 to white.
! LUT8 6 255 255 0       ; Set color out for LUT index 6 to yellow.

! MOVABS 0 0             ; Move current point to 0,0.
! VAL8 1                 ; Set current pixel value to 1 (red).
! CIRCLE 100             ; Draw a filled red circle.

! MOVABS -50 0           ; Move current point to -50,0.
! VAL8 2                 ; Set current pixel value to 2 (green).
! WMSK16 #0200           ; Write enable only bit plane 1.
! CIRCLE 100             ; Draw a filled green circle.  The intersection
                           of this circle and the red circle is blue, since
                           bit plane 0 is write protected.
! MOVABS 0 75            ; Move current point to 0,75: in the intersection
                           of the two circles.
! VAL8 4                 ; Set current pixel value to 4 (yellow).
! WMSK16 #0500           ; Write enable only bit planes 0 and 2.
! FILMSK 1,0,0           ; Check data on bit plane 1 only for area fill
                           tests.
! AREA1                  ; Perform area fill in yellow.  Fill the overlapped
                           region and the partial red circle.  Do not fill
                           the partial green circle, because the fill mask
                           is ANDed with the green boundary value before
                           the comparison is made.
```

FLOOD

SYNTAX

ASCII	FLOOD	
FORTRAN Call	CALL FLOOD	
Binary	[7]	(1 byte)

FUNCTION

The FLOOD command changes all displayed pixels to the current pixel value (VREG 0) in several frame times. All write-enabled memory is changed.

EXAMPLE

```
! LUT8 1 0,0,0          ; Set the color out for LUT index 1 to black.
! LUT8 2 0,0,255        ; Set the color out for LUT index 2 to blue.
! VAL8 1                ; Set current pixel value to 1 (black).
! FLOOD                 ; Flood displayed image; screen turns black.
! VAL8 2                ; Change current pixel value to 2 (blue).
! FLOOD                 ; Flood displayed image; screen turns blue.
```

FLUSH FLUSH

SYNTAX

ASCII	FLUSH	
FORTRAN Call	CALL FLUSH	
Binary	[21]	(1 byte)
	21 decimal = 025 octal = 15 hex	

FUNCTION

The FLUSH command empties the function button event queue. The event queue keeps a record of all function buttons which have been pushed at the local XY digitizer device. Each time the host issues a READBU command, one entry is taken out of the event queue and sent to the host. If the host is not interested in knowing which buttons have been pushed, the FLUSH command should be included in macros which use function buttons. The event queue holds eight entries. Overflow of the event queue results in subsequent function buttons being ignored.

EXAMPLE

```
! MACDEF 10             ; Start definition of Macro 10.
$ CIRCLE 50             ; Draw a circle.
$ FLUSH                 ; Empty event queue.
$ MACEND                ; End macro definition.
! BUTTBL 13 10          ; Execute Macro 10 in response to Button 13.
```

SYNTAX

ASCII	HDCOPY plotter, method, blackflag	
FORTRAN Call	CALL HDCOPY (IPLOT, METHOD, IBLACK)	
Binary	[112] [plotter] [method] [blackflag]	(3 bytes)
	112 decimal = 160 octal = 70 hex	

FUNCTION

The HDCOPY command prints the image memory, using the Diablo C-150 ink-jet printer (plotter 0). Method indicates the printing method to be used; because the C-150 can print dots in only 7 colors, some translation must be made. Blackflag allows the background to be defined as black (as on a CRT) or white (as on paper).

The only implemented plotter is the C-150 (plotter = 0).

There are five printinq methods available, as specified with the method parameter.

DITHER1 (0): Uses a statistical dithering algorithm, mapping each screen pixel directly to a single printed dot. This method works well for shaded images or images using many colors. However, it does give up some spatial resolution for color resolution, so some images may look grainy as a result. This method is completely unsuitable for line drawings.

PATTERN1 (1): Uses a patterned dither method, which allows more apparent colors. It maps one screen pixel to one printed dot. This method works well for images with a relatively low number of colors and especially for images with large blocks of a single color, such as business graphics. It is not a good method for line drawings.

DITHER2 (2): Uses a similar approach to that used by DITHER1. It maps one screen pixel to four printer dots. This results in better color resolution and very little spatial degradation. The printout is larger; a full screen uses most of a standard 8 1/2" x 11" sheet. If used for line drawings, the lines will appear soemwhat fuzzy.

FUNCTION, continued

PATTERN2 (3): Uses an approach similar to that used by PATTERN1. It maps one screen pixel to four printer dots. Its primary advantages over PATTERN1 are that it provides more colors, better spatial resolution, and provides a full-sized printout.

DIRECT (4): Allows the user to have direct control over the printer. Individual bit planes are mapped to each pen. Bit planes are mapped as follows:

Bit 7: black
Bit 5: magenta
Bit 3: yellow
Bit 1: cyan

To use the value commands to write only to the appropriate bit planes, these values should be used:

cyan:	2
yellow:	8
magenta:	32
black:	128
green:	10
red:	40
purple:	34
white:	0

Other values will cause the ink to "bleed" and blur the printout. This is the best method for line drawings.

You can print text/graphics in two different sizes. Use DITHER1 and PATTERN1 for a smaller size, and DITHER2 and PATTERN2 for a picture which is twice as large. The double size picture prints out sideways.

Note: HDCOPY will not print a black border when using the printing methods that double the size of the text. Users should also be aware the top two lines of the screen are sometimes not printed.

Blackflag, which selects the background color, may be BLACK (0) or WHITE (1). This flag is used for all methods except DIRECT. The default is BLACK (as it is on CRT screens).

FUNCTION, continued

Printing Alpha Text

The command ALPHEM PRNTON causes any text on the screen to be output to the printer. To turn off this mode, use the command ALPHEM PRNTOFF.

ASCII PARAMETERS

plotter	Specifies the type of plotter to be used; must be set to 0 or C150 (the Diablo C-150 ink-jet printer).
method	Specifies the printing method; must be 0 - 4 (see above).
blackflag	Blackflag = 0 or BLACK defines the background as black (default); blackflag = 1 or WHITE defines the background as white.

FORTRAN PARAMETERS

INTEGER*2	IPLOT, METHOD, IBLACK

EXAMPLE

```
! HDCOPY C150 DITHER1 WHITE          ; Prints the full screen using the
                                       DITHER1 method and white as the
                                       background color.
! HDCOPY 0 1 0                       ; Prints a smaller than full-screen
                                       using the PATTERN1 patterned dither
                                       method and black as the background
                                       color.
```

HELP

SYNTAX

ASCII		HELP	
	or	HELP command (mnemonic or opcode)	
Binary		[62]	(1 byte)
		62 decimal = 76 octal = 3E hex	

FUNCTION

The HELP command is normally used only in local communications to the alphanumeric terminal or when using a local keyboard. The information is sent to the alpha terminal or screen, regardless of where the command was initiated.

The HELP command has three forms.

HELP lists all commands that are available, in opcode order.

HELP command lists the command mnemonic and opcode and the data transmission format of the parameters for the command.

EXAMPLE

```
! HELP MOVABS              ; Request information on the MOVABS command.

     MOVABS: OPCODE = 001
1.   <16-BIT SIGNED NUMBER>
2.   <16-BIT SIGNED NUMBER>

! HELP 3                   ; Request information on command with opcode 3.

     MOV3R: OPCODE = 003
1.   <8-BIT SIGNED NUMBER>
2.   <8-BIT SIGNED NUMBER>
```

SYNTAX

ASCII	HOSTO string
FORTRAN Call	CALL HOSTO (STRLEN, STRING)
Binary	[181] [strlen] ([char1][char2]...[charn])
	181 decimal = 265 octal = B5 hex

FUNCTION

The HOSTO (HOST Output) command outputs a text string to the host computer over the host interface. The text is specified by string. If the command is being entered in ASCII mode from the local alphanumeric terminal or keyboard, the string to be output is the set of ASCII characters remaining on the command line. If the command is not being sent in ASCII mode, then the first byte of string contains the number of characters in the string (strlen) followed by strlen bytes containing the ASCII characters to be drawn.

Nonprintable characters can be included in HOSTO text strings. Characters with the high bit set or certain control characters (e.g., CTRL-S) can be put into string arguments in ASCII mode by using the backslash (\) to indicate that the immediately following characters are the numeric value of the desired character:

\ddd	Indicates 3 decimal digits representing the value of the desired character.
\#hh	Indicates the 2 hexadecimal digits representing the desired character.
\\	Indicates the backslash character (\) itself.

ASCII PARAMETER

string	The text to be printed.

FORTRAN PARAMETERS

```
INTEGER*2     STRLEN, STRING(1)
```

STRLEN is an integer specifying the number of characters that are to be output.

STRING is an integer array with two characters packed per 16-bit word, as in FORTRAN A2 format.

EXAMPLE

```
! HOSTO ABCDEF 1 2 3     ; Output text string "ABCDEF 1 2 3" to
                           the host.

! HOSTO WXYZ             ; Output text string "WXYZ" to the host.
```

IGNORE IGNORE

SYNTAX

ASCII	IGNORE flag
FORTRAN Call	CALL IGNORE (FLAG)
Binary	[124] [flag] (2 bytes)
	124 decimal = 174 octal = 7C hex

FUNCTION

The IGNORE command sets the display controller to ignore (flag = 1 or ON) or process (flag = 0 or OFF) subsequent incoming commands. With IGNORE ON, commands are interpreted but graphics primitives are not drawn, picked, highlighted, or unhighlighted, according to the current execution mode set with the EXMODE command.

Note: This command is intended only for use in systems without local display lists. Therefore, no example is given here.

ASCII PARAMETER

flag	Flag = 1 or ON, primitives are not drawn, picked, highlighted, or unhighlighted.
	Flag = 0 or OFF, primitives are either drawn, picked, highlighted, or unhighlighted, according to the current execution mode.

FORTRAN PARAMETER

INTEGER*2	FLAG

INCPID

SYNTAX

ASCII	INCPID
FORTRAN Call	CALL INCPID
Binary	[215] (1 byte)
	215 decimal = 327 octal = D7 hex

FUNCTION

The INCPID command increments the current pick ID number by one.

EXAMPLE 1

```
! SEGDEF 1             ; Begin definition of Segment 1.
$ PICKID 50            ; Assign pick ID 50 to the following commands.
$   .
$   .
$ INCPID               ; Assign pick ID 51 to the following commands.
$   .
$   .
$ INCPID               ; Assign pick ID 52 to the following commands.
$    .
$    .
$ SEGEND               ; End definition of Segment 1.
```

EXAMPLE 2

In this example, the INCPID command is used in the definition of Segment 20. Each time Segment 20 is referenced by Segment 10, the pick ID in Segment 20 is incremented by 1.

```
! SEGDEF 10            ; Begin definition of Segment 10.
$ PICKID 1             ; Assign pick ID 1 to the following commands.
$ MOVABS -100,0        ; Move current point to position circle 1.
$ SEGREF 20            ; Nest Segment 20.
$ MOVABS 100,0         ; Move current point to position circle 2.
$ SEGREF 20            ; Nest Segment 20.
$ SEGEND               ; End pick ID 1; end definition of Segment 10.

! SEGDEF 20            ; Begin definition of Segment 20.
$ INCPID               ; Increment pick ID by 1.
$ CIRCLE 50            ; Draw circle of radius 50.
$ SEGEND               ; End definition of Segment 20.
```

SYNTAX

ASCII	INSPID
FORTRAN Call	CALL INSPID
Binary	[119] (1 byte)
	119 decimal = 167 octal = 77 hex

FUNCTION

The INSPID command opens a segment for insertion of graphics primitives. The segment number is contained in SEGREG (segment ID register) and the primitives are inserted following the PICKID or INCPID command whose argument is equal to the quantity contained in the PIDREG (pick ID register).

Any command which is legal in segment definition mode can be inserted until a SEGEND command is encountered. An error will be generated if the specified PICKID within the specified segment does not exist.

Note: The SEGREG and PIDREG can be set by picking or using the SETGL or RDPICK commands.

LUT8 LUT8

SYNTAX

ASCII	LUT8 index, r, g, b
FORTRAN Call	CALL LUT8 (INDEX, IRENT, IGENT, IBENT)
Binary	[28] [index] [r] [g] [b] (5 bytes)
	28 decimal = 034 octal = 1C hex

FUNCTION

The LUT8 command changes the color entries in the given index of the LUT to the red, green, and blue values specified by r, g, and b.

The Model One/10 has a single 8-bit-in, 24-bit-out look-up table. The color values that are stored are the values that will be loaded into their respective digital-to-analog converters (DACs) when a pixel of value index is encountered when screen refresh is being performed.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
r, g, b	Red, green, and blue entries; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2	INDEX, IRENT, IGENT, IBENT

EXAMPLE

```
! LUT8 50 255,0,170       ; Set the color out for LUT index 50 to 255,0,170
                            (pink).
! VAL8 50                 ; Set current pixel value to 50 (pink).
! FLOOD                   ; Flood displayed pixels to current pixel
                            value; screen turns pink.
! LUT8 50 100 100 100     ; Change the color out for LUT index 50 to
                            100,100,100; screen turns grey.
! LUT8 50 200 100 50      ; Change the color out for LUT index 50 to
                            200,100,50; screen turns orange.
```

LUT16

SYNTAX

ASCII	LUT16 index, r, g, b
FORTRAN Call	CALL LUT16 (INDEX, IRENT, IGENT, IBENT)
Binary	[22] [highindex] [lowindex] [rentry] [gentry] [bentry] (6 bytes) 22 decimal = 26 octal = 16 hex

FUNCTION

The LUT16 command changes the red, green, and blue components of the Model One/10's look-up table at the specified index. Index is a 16-bit value and specifies the location from 0 to 1023 in the look-up table. The LUT16 command is the only way of changing the look-up table indices from 256 to 1023. The command functions similarly to the LUT8 command.

The Model One/10 has a single 8-bit-in, 24-bit-out look-up table. The color values that are stored are the values that will be loaded into their respective digital-to-analog converters (DACs) when a pixel of value index is encountered when screen refresh is being performed.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 1023.
r, g, b	Red, green, and blue entries; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2	INDEX, IRENT, IGENT, IBENT

EXAMPLE

```
! WMSK16 #FF03          ; Set the write and read masks to write into and
                          read from all 10 bit planes:
                          FF is 8 bits for graphics
                          03 is for the overlay planes.
! LUT16 350 0 255 0     ; Load look-up table index 350 with all green.
! VAL16 #5401           ; Set current pixel value to value of look-up
                          table index 350 (decimal).  Note that 350 deci-
                          mal = 154 hexadecimal, which for VAL16 has its
                          bytes swapped, so that the "54" is in the high
                          (left) byte, and the "01" is in the low byte.
! DRWREL 100 0          ; Draw a green vector.
```

SYNTAX

ASCII	LUTA index, entry	
FORTRAN Call	CALL LUTA (INDEX, ENTRY)	
Binary	[27] [index] [entry]	(3 bytes)
	27 decimal = 033 octal = 1B hex	

FUNCTION

The LUTA command changes the red, green, and blue components of the color at the specified LUT index to entry. The result is a shade of grey. The LUTA command is most useful for monochrome applications.

The entry stored in the LUT is passed to the red, green, and blue digital to analog converters (DACs) when a pixel of value index is encountered when reading from image memory to refresh the display screen.

The LUTA command affects only the values at indices 0 through 255; to change indices 256 to 1023, the LUT16 command must be used.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
entry	The entry at the LUT index; range is 0 to 255.

FORTRAN PARAMETERS

```
INTEGER*2      INDEX, ENTRY
```

EXAMPLE

```
! LUT8 1 255,255,255     ; Set the color out for LUT index 1 to white.
! VAL8 1                 ; Set current pixel value to 1 (white).
! FLOOD                  ; Flood displayed pixels to current pixel value;
                           screen turns white.
! LUTA 1 0               ; Change the color out for LUT index 1 to 0,0,0;
                           screen turns black.
! LUTA 1 100             ; Change the color out for LUT index 1 to
                           100,100,100; screen turns grey.
```

LUTB LUTB

SYNTAX

ASCII	LUTB index, entry
FORTRAN Call	CALL LUTB (INDEX, ENTRY)
Binary	[26] [index] [entry] (3 bytes)
	26 decimal = 032 octal = 1A hex

FUNCTION

The LUTB command changes the blue component of the color value at the look-up table index to the value entry.

The entry stored in the LUT is passed to the blue digital to analog converter (DAC) when a pixel of value index is encountered when reading from image memory to refresh the display screen.

Only the values at indices 0 to 255 are changed by the LUTB command. To change the values from 256 through 1023, LUT16 must be used.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
entry	The blue component entry at the LUT index; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 INDEX, ENTRY

EXAMPLE

```
! LUT8 1 255,255,255    ; Set the color out for LUT index 1 to white.
! VAL8 1                ; Set current pixel value to 1 (white).
! FLOOD                 ; Flood all displayed pixels to current pixel
                          value; screen turns white.
! LUTB 1 0              ; Set blue component at LUT index 1 to 0;
                          screen turns yellow.
```

LUTG

SYNTAX

ASCII	LUTG index, entry
FORTRAN Call	CALL LUTG (INDEX, ENTRY)
Binary	[25] [index] [entry] (3 bytes)
	25 decimal = 031 octal = 19 hex

FUNCTION

The LUTG command changes the green component of the color value at the look-up table index to the value entry.

The green component entry stored in the LUT is passed to the green digital to analog converter (DAC) when a pixel of value index is encountered when reading from image memory to refresh the display screen.

Only the values at indices 0 to 255 are changed by the LUTG command. To change the values from 256 through 1023, LUT16 must be used.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
entry	The green component entry at the LUT index; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 INDEX, ENTRY

EXAMPLE

```
! LUT8 1 255,255,255    ; Set the color out for LUT index 1 to white.
! VAL8 1                ; Set current pixel value to 1 (white).
! FLOOD                 ; Flood all displayed pixels to current pixel value;
                          screen turns white.
! LUTG 1 0              ; Set green component at LUT index 1 to 0; screen
                          turns magenta.
```

SYNTAX

ASCII	LUTR index, entry
FORTRAN Call	CALL LUTR (INDEX, ENTRY)
Binary	[24] [index] [entry] (3 bytes)
	24 decimal = 030 octal = 18 hex

FUNCTION

The LUTR command changes the red component of the color value at look-up table index to the value entry.

The red component entry stored in the LUT is passed to the red digital to analog converter (DAC) when a pixel of value index is encountered when reading from image memory to refresh the display screen.

Only the values at indices 0 to 255 are changed by the LUTR command. To change the values from 256 through 1023, LUT16 must be used.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
entry	The red component entry at the LUT index; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2	INDEX, ENTRY

EXAMPLE

```
! LUT8 5 255,255,255     ; Set the color out for LUT index 5 to white.
! VAL8 5                 ; Set current pixel value to 5 (white).
! FLOOD                  ; Flood all displayed pixels to current pixel value;
                           screen turns white.
! LUTR 5 0               ; Set red component at LUT index 5 to 0; screen
                           turns cyan.
```

SYNTAX

ASCII	LUTRMP code, sind, eind, sval, eval
FORTRAN Call	CALL LUTRMP (NUM, ISIND, IEIND, ISENT, IEENT)
Binary	[29] [code] [sind] [eind] [sval] [eval] (6 bytes)
	29 decimal = 035 octal = 1D hex

FUNCTION

The LUTRMP command loads the LUT component(s) specified by code from LUT index sind to index eind with a ramp function linearly interpolated from the start entry sval to the end entry eval. The LUTRMP command is useful whenever multiple, successive LUT entries are to be set to either a ramp function or to a constant value. You cannot load LUT entries 256 through 1023 with the LUTRMP command. To load those indices, you must use the LUT16 command.

The LUT is not sent to a linear ramping function by default, and the LUTRMP command may have somewhat unexpected results.

Note: To reset the LUT to the default, use the COLD command or the MODDIS 1 command, or use a series of individual LUT commands. The COLD command and the MODDIS 1 command both erase the screen.

ASCII PARAMETERS

code	The LUT components to load:
	code = 1, load blue component
	code = 2, load green component
	code = 4, load red component
	code = 7, load all components

sind, eind	Starting and ending indices in the LUT to load; range is 0 to 255.
sval, eval	Starting and ending entries to load into the indices; range is 0 to 255.

FORTRAN PARAMETERS

```
INTEGER*2     NUM, ISIND, IEIND, ISENT, IEENT
```

EXAMPLE

```
! LUT8 1 255,255,255        ; Set the color out for LUT index 1 to white.
! LUT8 2 200,200,200        ; Set the color out for LUT index 2 to grey.
! LUT8 3 150,150,150        ; Set the color out for LUT index 3 to a darker
                              shade of grey.
! LUT8 4 100,100,100        ; Set the color out for LUT index 4 to a darker
                              shade of grey.
! PRMFIL ON                 ; Select filled primitives.
! LUTRMP 7 0 255 0 255      ; Load indices 0 to 255 in all components of
                              LUT with values between 0 and 255 (ramp
                              function).
! VAL8 1                    ; Set current pixel value to 1 (white).
! CIRCLE 100                ; Draw a white circle.
! VAL8 2                    ; Set current pixel value to 2 (grey).
! CIRCLE 80                 ; Draw a circle in a light shade of grey.
! VAL8 3                    ; Set current pixel value to 3 (darker grey).
! CIRCLE 60                 ; Draw a circle in a darker shade of grey.
! VAL8 4                    ; Set current pixel value to 4 (darker grey).
! CIRCLE 40                 ; Draw a circle in a darker shade of grey.
! LUTRMP 7 0 255 255 0      ; Load indices 0 to 255 in all components of
                              LUT with values between 255 and 0 (reverse
                              ramp function). The circles change colors;
                              the outer circle is now black.
! LUTRMP 1 0 255 0 255      ; Load indices 0 to 255 in the blue component
                              of the LUT with values between 0 and 255.
! LUTRMP 2 100 255 0 0      ; Load indices 100 to 255 in the green component
                              of the LUT with 0.
```

MACDEF

SYNTAX

ASCII	MACDEF num	
FORTRAN Call	CALL MACDEF (NUM)	
Binary	[139] [num]	(2 bytes)
	139 decimal = 213 octal = 8B hex	

FUNCTION

The MACDEF command defines a new macro specified by num. After you enter the MACDEF command from the local terminal, the system displays the prompt $. You can then enter a series of commands which are included in the macro definition, and are not executed until the macro is executed. Macro definition ends when you type the command MACEND.

A macro may include any combination of valid command strings, including nested MACDEF commands and user-defined commands. Up to 16 levels of macros can be nested. Macros cannot contain QUIT or ASCII commands. The length of a macro command is limited only by the available memory space.

ASCII PARAMETER

num	The macro number; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 NUM

Note: Readback commands within macro definitions are inserted into the macro definition but the host does not read any data.

EXAMPLE

```
! MACDEF 40          ; Start definition of Macro 40.
$ CIRCLE 50          ; Draw circle of radius 50.
$ CIRCLE 30          ; Draw circle of radius 30.
$ MACEND             ; End definition of Macro 40.
! MACRO 40           ; Execute Macro 40; draw two concentric circles.
```

MACEND

SYNTAX

ASCII	MACEND
FORTRAN Call	CALL MACEND
Binary	[12] (1 byte)
	12 decimal = 014 octal = 0C hex

FUNCTION

The MACEND command ends a macro definition. If no macro is being defined, an error results. A MACEND command must be used for each MACDEF command.

EXAMPLE

```
! MACDEF 17          ; Start definition of Macro 17.
$ MOVABS 50 50       ; Move current point to 50,50.
$ DRWABS 100, 150    ; Draw a line to 100,150.
$ MACEND             ; End definition of Macro 17.
! MACRO 17           ; Execute Macro 17.
```

MACERA

SYNTAX

ASCII	MACERA num
FORTRAN Call	CALL MACERA (NUM)
Binary	[140] [num] (2 bytes)
	140 decimal = 214 octal = 8C hex

FUNCTION

The MACERA command clears macro definition number num. Macro number num cannot be executed after the MACERA command has been issued. Attempting to execute a macro that has been erased has no effect and does not result in any errors.

ASCII PARAMETER

num	The macro number; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 NUM

EXAMPLE

```
! MACDEF 23              ; Begin definition of Macro 23.
$ CIRCLE 50             ; Draw a circle of radius 50.
$ MACEND                ; End definition of Macro 23.
! MOVABS 0 0            ; Move current point to 0,0.
! MACRO 23              ; Execute Macro 23; draw a circle with center
                          at 0,0.
! MACERA 23             ; Erase the definition of Macro 23.
! MOVABS 50 50          ; Move current point to 50,50.
! MACRO 23              ; Execute Macro 23; nothing happens because
                          Macro 23 has been erased.
```

SYNTAX

ASCII	MACRO num
FORTRAN Call	CALL MACRO (NUM)
Binary	[11] [num] (2 bytes)
	11 decimal = 013 octal = 0B hex

FUNCTION

The MACRO command executes macro number num. You can also execute macros by pressing buttons (see the BUTTBL command) or by using the BUTTON command. Executing a macro that has not been defined has no effect and does not result in any errors.

ASCII PARAMETER

num	The macro number; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 NUM

Note: If the macro contains readback commands, their interpretation is left up to the application program.

EXAMPLE

```
! MACDEF 23          ; Begin definition of Macro 23.
$ CIRCLE 50          ; Draw a circle of radius 50.
$ MACEND             ; End definition of Macro 23.
! MACRO 23           ; Execute Macro 23.
```

SYNTAX

ASCII	MODDIS flag
FORTRAN Call	CALL MODDIS (IFLAG)
Binary	[44] [flag] (2 bytes)
	44 decimal = 054 octal = 2C hex

FUNCTION

The MODDIS command resets the following to their default states: look-up tables, value registers, coordinate registers, text size and rotation, blink table, READF, RDMODE, current transformation, and tablet scaling factor. The command also resets the states of the RGBTRU and ALPHEM commands to the defaults stored in NVRAM. The MODDIS command clears the screen. Transformations are reset to the identity matrix. Only macro definitions, segment definitions, and downloaded text font (font 2) remain unchanged.

Note: A MODDIS command sent from the host should be followed immediately by a WAIT 0.

ASCII PARAMETERS

flag	Display mode flag; values of 0 (OFF) or 1 (ON) both have the same effect.

FORTRAN PARAMETERS

INTEGER*2	IFLAG

MOV2R

SYNTAX

ASCII	MOV2R dx, dy	
FORTRAN Call	CALL MOV2R (IX, IY)	
Binary	[4] [dxdy]	(2 bytes)

FUNCTION

The MOV2R command is a two-byte version of the MOVREL command. The MOV2R command should only be called by the host in binary mode.

MOV2R changes the WCS current point (CREG 0) to the point specified by the (dx,dy) offset from the current point. Only two bytes are required for the command. The range of dx and dy is -8 to 7, and dx,dy is specified as nibbles of a single byte.

ASCII PARAMETERS

dx, dy	The relative offset for the coordinate; range is -8 to 7.

FORTRAN PARAMETERS

INTEGER*2	IX, IY

The FORTRAN library manages the packing of the two numbers into a single byte.

EXAMPLE

Because nibbles are used to specify dx,dy, some examples may be useful:

MOV2R 17 moves (+1,+1): 17 = (16*1)+1 = 00010001

MOV2R -103 moves (-7,-7): -103 = 10011001

MOV2R -1 moves (-1,-1): -1 = 16*(NOT(1)+1)+(NOT(1)+1) = 11111111 = -1

The rules to generate this number are:

1. Negative numbers are encoded in binary as two's complement notation.

2. Positive binary is converted directly to decimal or hexadecimal.

3. Negative binary is converted to negative decimal by performing a two's complement binary weighting.

4. Negative binary is converted to hexadecimal by straight conversion on binary weighting and preceding both hex numbers with #FF.

MOV3R

SYNTAX

ASCII	MOV3R dx, dy
FORTRAN Call	CALL MOV3R (IDX, IDY)
Binary	[3] [dx] [dy] (3 bytes)

FUNCTION

The MOV3R command is a three byte form of the MOVREL command. The MOV3R command changes the current point (CREG 0) by the relative amount specified by dx,dy. This command reduces the number of bytes which must pass between the Model One and the host computer when the displacement of the current point is within the given range.

ASCII PARAMETERS

dx, dy	The relative offset for coordinate; range is -128 to 127.

FORTRAN PARAMETERS

INTEGER*2	IDX, IDY

MOVABS MOVABS

SYNTAX

ASCII	MOVABS x, y	
FORTRAN Call	CALL MOVABS (IX, IY)	
Binary	[1] [highx][lowx] [highy][lowy]	(5 bytes)

FUNCTION

The MOVABS command changes the WCS current point (CREG 0) to the point specified by x,y. All subsequent graphics primitives (lines, circles, arcs, polygons, etc.) are drawn beginning at the location of the current point.

ASCII PARAMETERS

x,y	16-bit integers specifying the absolute x,y coordinate; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2	IX, IY

EXAMPLE

```
! MOVABS 50 70          ; Move the current point to 50,70.
! DRWABS 100 -10        ; Draw a line from current point to 100,-10.
! CIRCLE 15             ; Draw a circle centered at 100,-10 and with
                          radius 15.
! MOVABS 0 0            ; Move the current point to 0,0.
! CIRCLE 20             ; Draw a circle centered at 0,0 and with
                          radius 20.
```

MOVI

SYNTAX

ASCII	MOVI creg
FORTRAN Call	CALL MOVI (ICREG)
Binary	[5] [creg] (2 bytes)

FUNCTION

The MOVI command changes the WCS current point (CREG 0) to the address specified in coordinate register creg. The MOVI command is most often used to access the current coordinate from the digitizing tablet which is stored in CREG 2.

The MOVI command has the same effect at the command CMOVE 0 creg, which copies a given coordinate register into CREG 0.

ASCII PARAMETER

creg	Coordinate register; range is 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETER

INTEGER*2	ICREG

EXAMPLE

```
! CLOAD 15 100 150      ; Load 100,150 into CREG 15.
! MOVI 15               ; Move to location given in CREG 15.
! DRWABS 140 100        ; Draw line from current point to 140,100.
! MOVI 2                ; Move to the location given in CREG 2 (the
                          current digitizing tablet location).
! CIRCLE 25             ; Draw a circle of radius 25 centered at the
                          current point.
```

MOVREL

SYNTAX

ASCII	MOVREL dx, dy
FORTRAN Call	CALL MOVREL (IDX, IDY)
Binary	[2] [highdx][lowdx] [highdy][lowdy] (5 bytes)

FUNCTION

The MOVREL command changes the WCS current point (CREG 0) by a relative amount specified by dx and dy. The new current point is set to the sum of the x-component of the old current point plus dx and the sum of the y-component of the old current point plus dy.

ASCII PARAMETERS

dx, dy	16-bit integer values specifying the relative offset for the coordinate; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2	IDX, IDY

EXAMPLE

```
! MOVABS 100 -130        ; Move the current point to 100,-130.
! CIRCLE 50             ; Draw circle of radius 50 centered at 100,-130.
! MOVREL 20 20          ; Move current point by 20,20 to 120,-110.
! CIRCLE 30             ; Draw circle of radius 30 centered at 120,-110.
```

NULL NULL

SYNTAX

ASCII	NULL
FORTRAN Call	CALL NULL
Binary	[0 or 10 or 13 or 128 or 138 or 141] (1 byte)
	Values in hex are: 00 or 0A or 0D or 80 or 8A or 8D.

FUNCTION

The NULL command is analogous to a NOP (No OPeration) and has no effect. The NULL command has several opcodes, any one of which executes the NULL command.

The NULL command can be used to pad a command data buffer so that the Model One can be used on systems capable of transmitting only fixed length blocks.

The opcodes of the NULL command were chosen so that the Model One ignores carriage returns and linefeeds sent between commands for hosts that cannot be made to inhibit sending these characters.

SYNTAX

ASCII	PEEK addr
FORTRAN Call	CALL PEEK (IADDR, IWORD)
Binary	[189] [highaddr][lowaddr] (3 bytes)
	189 decimal = 275 octal = BD hex

FUNCTION

The PEEK command sends the contents of the central processor memory location addr to the requesting device. The PEEK command is intended to be used in conjunction with the POKE command as a debugging tool for implementers, and not as a general purpose command for users.

The address must be an even number (word address); if the address is odd, bit 0 will be ignored. The data is displayed in ASCII hexadecimal.

If the PEEK command is issued by the host, the Model One will wait for an ACK (06 hex or 86 hex) character from the host before continuing command interpretation. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

ASCII PARAMETER

addr	16-bit integer specifying the address in central processor memory from which to read data.

FORTRAN PARAMETERS

Input parameter:	INTEGER*2	IADDR
Output parameter:	INTEGER*2	IWORD

IWORD is set equal to the contents of the location specified by IADDR. The FORTRAN library handles transmission of the ACK character.

EXAMPLE

```
! PEEK 0                 ; Display contents of location 0.
  FFFF                     (Possible response from Model One.)
! PEEK #0FFE             ; Display contents of location 0FFE hex.
  003F                     (Possible response from Model One.)
```

PICKID PICKID

SYNTAX

ASCII	PICKID pickid
FORTRAN Call	CALL PICKID (PID)
Binary	[217] [pickid3][pickid2][pickid1][pickid0] (5 bytes)
	217 decimal = 331 octal = D9 hex

FUNCTION

The PICKID command is only useful within the definition of a segment.

The PICKID command assigns a pick identification number to all the graphics primitives and other commands immediately following the PICKID command until the next instance of the PICKID command or until the end of the segment.

When a segment is executing, the current pick ID remains in effect until either the next PICKID command or the current segment completes execution. When the current segment completes execution, the pick ID is set to what it was before execution of the segment.

The primitives tagged with a PICKID label can later be deleted with the DELPID command; new primitives can be added to the end of a segment with the SEGAPP command.

ASCII PARAMETER

pickid	A 32-bit pick identification number; range is 00000000 to FBFFFFFF (hex). PICKIDs in the range FC000000 to FFFFFFFF are reserved.

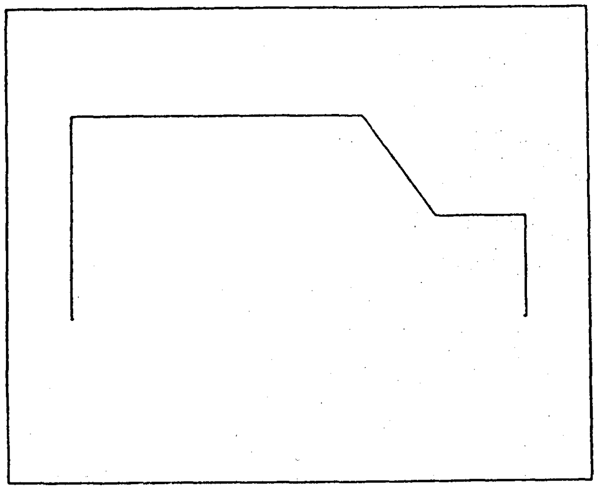
FORTRAN PARAMETER

INTEGER*4 PID

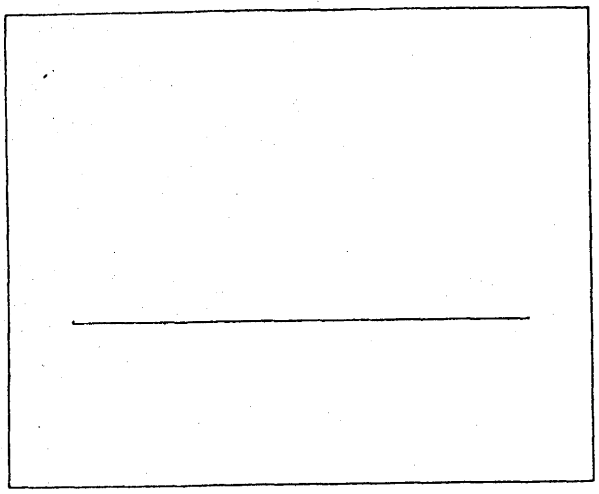
EXAMPLE

```
! SEGINI 0 128          ; Initialize display list structures with a
                          block size of 128 bytes.
! SEGDEF 1              ; Begin definition of Segment 1, which defines
                          the truck and the position of the 2 wheels.
$ PICKID 10             ; Assign pick ID 10 label to the truck body.
$ PUSH CREG CURPNT      ; Push WCS current point onto the stack.
$ MOVABS -250,-80       ; Move current point to lower left corner of truck.
$ DRWREL 0,220          ; Draw 5 vectors to define truck body.
$ DRWREL 320,0
$ DRWREL 80,-110
$ DRWREL 100,0
$ DRWREL 0,-110
$ PICKID 20             ; End pick ID 10 (body); assign pick 20 label to
                          the chassis (bottom line) of truck.
$ DRWREL -500,0         ; Draw chassis of truck.
$ MOVABS -160,-100      ; Move current point to position rear wheel.
$ SEGREF 2              ; Nest Segment 2, the wheel.
$ MOVABS 155,-100       ; Move current point to position front wheel.
$ SEGREF 2              ; Nest Segment 2, the wheel.
$ POP CREG CURPNT       ; Pop WCS current point from the stack.
$ SEGEND                ; End pick ID 20 (chassis); end definition of
                          Segment 1, the truck body and position of wheels.

! SEGDEF 2              ; Begin definition of Segment 2, which defines the
                          shape of the wheel.
$ PICKID 30             ; Assign pick ID 30 label to the wheel.
$ PUSH CREG CURPNT      ; Save current point (center of wheel).
$ MOVREL -50,-50        ; Move current point to left corner of wheel.
$ RECREL 100,100        ; Draw wheel.
$ POP CREG CURPNT       ; Restore current point (center of wheel)
$ SEGEND                ; End pick ID 30 (wheel); end definition of
                          Segment 2, the wheel.
!                       ; Model One returns to normal command mode.
```

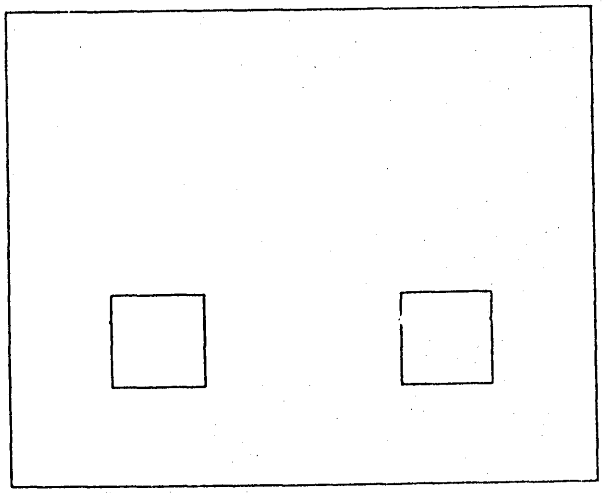
Primitives Tagged "PICKID 10"



Primitives Tagged "PICKID 20"



Primitives Tagged "PICKID 30""



PIXEL8

SYNTAX

ASCII	PIXEL8 nrows, ncols, val, ...
FORTRAN Call	CALL PIXEL8 (NROWS, NCOLS, IDATA)
Binary	[41] [highnrows] [lownrows] [highncols] [lowncols] ([val]) ... (5 + nrows * ncols bytes)
	41 decimal = 51 octal = 29 hex

FUNCTION

The PIXEL8 command transmits an image to the Model One pixel by pixel. The array of transmitted data is nrows high and ncols wide. The upper left corner of the array is defined by the current point (CREG 0). Pixels in image memory are filled left to right, top to bottom. Each pixel value is sent as an 8-bit quantity, which is used as an index to the look-up table. The PIXEL8 command is most useful in loading one of the image memory banks with pixel data, as in an 8-bit pseudo color application.

The PIXEL8 command does not change the current point. The PIXEL8 command is not clipped. If the pixel data goes beyond the physical bounds of image memory, it will wrap around. You should avoid this situation, because the result is not always predictable.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is from 0 to 32,767.
val	8-bit value specifying the index in the look-up table; range is 0 to 225.

FORTRAN PARAMETERS

Input Parameters:	INTEGER*2	NROWS, NCOLS
Output Parameter:	LOGICAL*1	IDATA(1)

IDATA is a byte array which contains the pixel values. The first pixel value is in the first element of IDATA, and subsequent values are stored in successive array elements. IDATA must be dimensioned to at least NROWS * NCOLS elements.

EXAMPLE

To transmit a rectangular window 10 pixels wide by 5 pixels long,

- move the current point to the upper left corner of where you want to display image

- execute the command PIXEL8 5 10, followed by 50 single-byte values.

PIXELS

SYNTAX

ASCII	PIXELS nrows, ncols, r, g, b, ...
FORTRAN Call	CALL PIXELS (NROWS, NCOLS, IRED, IGRN, IBLU)
Binary	[40] [highnrows][lownrows] [highncols][lowncols] ([r] [g] [b]) ... (5 + 3 * nrows * ncols bytes)
	40 decimal = 50 octal = 28 hex

FUNCTION

The PIXELS command transmits an image to the Model One pixel by pixel. The array of transmitted data is nrows high and ncols wide. The upper left corner of the array is given by the current point (CREG 0). Pixels in image memory are filled left to right, top to bottom. Each pixel value is sent as a full, 24-bit quantity, one byte each of red, green, and blue.

Only the red byte is used to determine the location in the look-up table. For this reason, the PIXELS command is very inefficient and you should only use it if you require compatibility with other Model Ones. Otherwise, use the PIXEL8 command.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is from 0 to 32,767.
r, g, b	8-bit integers specifying the red, green, and blue color values for each pixel; range is 0 to 255.

FORTRAN PARAMETERS

Input Parameters:	INTEGER*2	NROWS, NCOLS
Output Parameters:	LOGICAL*1	IRED(1), IGRN(1), IBLU(1)

IRED, IGRN, and IBLU are byte arrays which contain the red, green, and blue pixel values. Each array must be dimensioned to at least NROWS * NCOLS elements.

EXAMPLE

To transmit a rectangular window 10 pixels wide by 5 pixels long,

- move the current point to the upper left corner of where you want to display image

- execute the command PIXELS 5 10, followed by 50 three-byte values.

PIXL16 PIXL16

SYNTAX

ASCII	PIXL16 nrows, ncols, val, ...
FORTRAN Call	CALL PIXL16 (NROWS, NCOLS, IDATA)
Binary	[65] [highnrows][lownrows] [highncols][lowncols] ([val]) ... (5 + nrows * ncols bytes)
	65 decimal = 101 octal = 41 hex

FUNCTION

The PIXL16 command transmits an image to the Model One pixel by pixel. The array of transmitted data is nrows high and ncols wide. The upper left corner of the array is defined by the current point (CREG 0). Pixels in image memory are filled left to right, top to bottom. Each pixel value is sent as a 16-bit quantity, which is used as an index to the look-up table (as in the VAL16 command).

The PIXL16 command does not change the current point. The PIXL16 command is not clipped. If the pixel data goes beyond the physical bounds of image memory, it will wrap around. You should avoid this situation, because the result is not always predictable.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is 1 to 512 for nrows and 1 to 1024 for ncols.
val	16-bit value specifying the index in the look-up table; valid range is 0 to 1023.

FORTRAN PARAMETERS

Input Parameters:	INTEGER*2	NROWS, NCOLS
Output Parameter:	INTEGER*2	IDATA(NROWS * NCOLS)

IDATA is an integer array which contains the pixel values. The first pixel value is in the first element of IDATA, and subsequent values are stored in successive array elements. IDATA must be dimensioned to at least NROWS * NCOLS elements.

EXAMPLE

To transmit a rectangular window 10 pixels wide by 5 pixels long,

- move the current point to the upper left corner of where you want to display image

- execute the command PIXL16 5 10, followed by 50 single-byte values.

SYNTAX

ASCII	PIXMOV
FORTRAN Call	CALL PIXMOV
Binary	[187] (1 byte)
	187 decimal = 273 octal = BB hex

FUNCTION

The PIXMOV command performs a block pixel move. This command moves pixel data from a rectangular source window to a destination window of the same size.

The source window is specified by CREG 11 and CREG 12, which contain diagonally opposite corners of the window. The destination window is specified by CREG 13 and CREG 14. The pixel at the location in CREG 11 (the source window corner) is transferred to the location in CREG 13 (the destination window corner). CREG 14 controls the direction of scanning of pixels into the destination window. The direction of the displacement of CREG 14 relative to CREG 13 defines the position of the diagonal corner of the destination window, allowing mirroring.

EXAMPLE

```
! MOVABS 0 0              ; Move current point to 0,0.
! DRWABS 30 30            ; Draw a triangle.
! DRWABS 45 0
! DRWABS 0 0
! CLOAD 11 -50 50         ; Define source window in center of screen.
! CLOAD 12 50 -50
! CLOAD 13 -256 255       ; Define destination window in upper left corner
                            of screen.
! CLOAD 14 -156 155
! PIXMOV                  ; Move triangle in center window to window in
                            upper left corner of screen.
! CLOAD 13 -156 155       ; Redefine destination window.  Specify corners
                            in opposite order.
! CLOAD 14 -256 255
! PIXMOV                  ; Move triangle in center window to window in
                            upper left corner of screen. Triangle is
                            mirrored about both the x and y axes.
```

SYNTAX

ASCII	PIXMZM xzoom, yzoom
FORTRAN Call	CALL PIXMZM (IXZOOM, IYZOOM)
Binary	[242] [xzoom] [yzoom] (3 bytes)
	242 decimal = 363 octal = F3 hex

FUNCTION

The PIXMZM command performs a block pixel move. In addition, the block is zoomed (through pixel replication) at the time of the move. For example, PIXMZM 2,2 zooms the block by twice in both the x and y dimensions. This command moves the block of pixels in the rectangular window specified by CREGs 11 and 12 into the window specified by CREGs 13 and 14.

If xzoom and yzoom are both zero or one, this command is equivalent to the PIXMOV command (i.e., no zooming). If either xzoom or yzoom is zero (regardless of the other parameter's value), the command is equivalent to a PIXMOV.

The window size is specified by CREGs 11 and 12 (the source window). CREG 13 corresponds to CREG 11; the pixel at the location in CREG 11 is transferred to the location in CREG 13. CREG 14 indicates the direction in which the window extends from CREG 13. CREG 13 should always be the lower-left corner.

The PIXMZM command determines the upper-right corner of the destination region, even if CREG 14 is loaded incorrectly.

ASCII PARAMETERS

xzoom	The factor by which to zoom in the x dimension.
yzoom	The factor by which to zoom in the y dimension.

FORTRAN PARAMETERS

```
INTEGER*2       IXZOOM, IYZOOM
```

EXAMPLE

```
! CLOAD 13 -256 255          ; Set up destination window.
! CLOAD 14 -156 155          ; Set up destination window.
! CLOAD 11 -50 50            ; Set up source window.
! CLOAD 12 50 -50            ; Set up source window.
! PIXMZM 2 2                 ; Perform a block pixel move, zooming the
                               block by twice in both the x and y dimen-
                               sion.
```

SYNTAX

ASCII	POINT
FORTRAN Call	CALL POINT
Binary	[136] (1 byte)
	136 decimal = 210 octal = 88 hex

FUNCTION

The POINT command sets the current point (CREG 0) to the current pixel value (VREG 0). The current point and the current pixel value remain unchanged.

EXAMPLE

```
! LUT8 5 255,0,0         ; Set the color out for LUT index 5 to red.
! LUT8 6 255,255,255     ; Set the color out for LUT index 6 to white.
! MOVABS 0 0             ; Move current point to 0,0.
! PRMFIL ON              ; Select filled primitives.
! VAL8 5                 ; Change current pixel value to 5 (red).
! CIRCLE 30              ; Draw a red circle.
! VAL8 6                 ; Change current pixel value to 6 (white).
! POINT                  ; Set pixel at 0,0 to 6 (white).  You can see a
                           white point in the center of the red circle.
```

SYNTAX

ASCII	POKE addr, data
FORTRAN Call	CALL POKE (IADDR, IDATA)
Binary	[190] [highaddr][lowaddr] [highdata][lowdata] (5 bytes)

190 decimal = 276 octal = BE hex

FUNCTION

The POKE command writes a given word of data into a given address, addr, in central processor memory. The POKE command is intended to be used in conjunction with the PEEK command as a debugging tool for implementers, and not as a general purpose command for users.

ROM memory runs from 0 to 7FFF hex. POKEs into this memory have no effect.

RAM memory runs from 8000 to FFFF hex. You should not POKE into the memory area from 8000 to BFFF, which is the base RAM. POKEs into this area can crash the central processor.

ASCII PARAMETERS

addr	16-bit integer specifying the address in central processor memory in which to place data.
data	16-bit integer data.

FORTRAN PARAMETERS

INTEGER*2 IADDR, IDATA

EXAMPLE

```
! PEEK #C000            ; Display contents of memory location C000 hex.
  02DC                    (Possible response from Model One.)
! POKE #C000 #0000      ; Change contents of location C000 to 0.
! PEEK #C000            ; Display contents of location C000.
  0000                    (Response from Model One.)
```

POLYGN POLYGN

SYNTAX

ASCII	POLYGN npoly, nverts, x, y, ...
FORTRAN Call	CALL POLYGN (NPOLY, NVERT, VERTS)
Binary	[18] [npoly] [highnverts][lownverts] [highx1][lowx1] [highy1][lowy1]...
	18 decimal = 22 octal = 12 hex

FUNCTION

The POLYGN command draws polygons in image memory in the current pixel value (VREG 0). The total number of polygons is given by npoly. Each polygon is then specified by giving the number of vertices (nverts), followed by the list of vertices. Each vertex is specified by an x and y coordinate value.

All of the vertices are relative to the current point. The current point is unchanged by the POLYGN command.

Because the POLYGN command allows specification of more than one polygon at a time, the Model One supports arbitrary polygons with interior holes. The polygons may be drawn filled or unfilled, by setting the primitive fill flag with the PRMFIL command. With PRMFIL ON, the nested interior polygons are drawn unfilled; the surrounding polygon is filled.

ASCII PARAMETERS

npoly	8-bit parameter specifying the number of polygons to draw; range is 0 to 255.
nverts	16-bit parameter specifying the number of vertices for each polygon; you may specify up to a total of 251 vertices for all the unfilled polygons you define (4K of RAM is allocated for polygon fills).
x, y	16-bit parameters specifying the x and y coordinates for each vertex; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2	NPOLY, NVERT(1), VERTS(2,1)
NPOLY	Number of polygons to be drawn.
NVERT	One dimensional array containing the number of vertices in each polygon.
VERTS	Two dimensional array containing the dx and dy displacements from the current point of all vertices in the defined polygons. VERTS(1,j) contains the jth dx value. VERTS(2,j) contains the jth dy value. If more than one polygon is being defined, the vertices describing the second polygon should be placed in the array immediately after the vertices describing the first polygon.

EXAMPLE

```
! MOVABS 0 0             ; Move current point to 0,0.

! POLYGN  1 3 10,10  10,-10  -10,-10

                         ; Draw a triangle.
! MOVABS 100 100         ; Move current point to 100,100.

! POLYGN 1 3 10,10  10,-10  -10,-10

                         ; Draw same triangle as before, but with vertices
                           relative to 100,100.

! MOVABS -100 100        ; Move current point to -100,100.

! POLYGN 1 4 -30,30  30,30  -30,-30  30,30

                         ; Draw a 4-sided polygon.

! MOVABS -100 100        ; Move current point to -100,100.

! PRMFIL ON              ; Select filled primitives.

! POLYGN 2 3 10,-10  10,-10  -10,-10  4 20,20  20,-20  -20,-20  -20,20

                         ; Draw a triangle nested inside a rectangle.
                           The rectangle is filled and the triangle is not.
```

POP

ASCII SYNTAX

```
POP  item, arg     (general form)

POP  CREG, creg        (coordinate register)
POP  VREG, vreg        (value register)
POP  XFORM, xformtype  (transformation matrix)
POP  VAR, vartype      (system variable or flag)
```

FORTRAN Calls

```
CALL  POP (TYPE, CREG, VREG, XFORM, SYSVAR, FPREG)   (general form)

CALL POP0 (CREG)
CALL POP1 (VREG)
CALL POP2 (XFORM)
CALL POP3 (SYSVAR)
```

Binary

```
[118] [item = 0] [creg]          (3 bytes)
[118] [item = 1] [vreg]          (3 bytes)
[118] [item = 2] [xformtype]     (3 bytes)
[118] [item = 3] [vartype]       (3 bytes)

118 decimal = 166 octal = 76 hex
```

FUNCTION

The POP command restores the item that you specify by popping it off the stack. The POP command is the inverse of the PUSH command, which pushes items onto the stack.

You must pop stacked items in the exact opposite order as they were pushed.

The PUSH and POP commands perform stack overflow and underflow checking. An error is generated if you exceed the stack bounds.

No type checking is performed by the PUSH and POP commands. Invalid parameter values may produce unpredictable results.

FUNCTION, continued

The items that you can push and pop include

- the contents of registers
  - coordinate registers
  - value registers
- the current 2-D transformation matrix
- system variables or flags
  - primitive fill flag
  - vector pattern.

ASCII PARAMETERS

item	The item to be popped (1 byte) 0 = CREG Coordinate register. 1 = VREG Value register. 2 = XFORM Transformation. 3 = VAR System variable or flag.
creg	Coordinate register number (1 byte). You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.
vreg	Value register number (1 byte). You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.
xformtype	Transformation type (1 byte). 0 2-D transformation matrix.
vartype	System variable or flag (1 byte). 0 = PRMFIL Primitive fill flag. 1 = VECPAT Vector pattern.

FORTRAN PARAMETERS

```
CALL POP (TYPE, CREG, VREG, XFORM, SYSVAR, FPREG)

INTEGER*2     TYPE, CREG, VREG, XFORM, SYSVAR, FPREG
```

Note: The parameter FPREG does not apply to the Model One/10.

EXAMPLE

```
! XFORM2D ABS 0 .5 0 0 .5 0 0

                           ; Define an absolute 2-D transformation which
                             scales by one half.
! SEGREF 1                 ; Draw geometry scaled by one half.
! PUSH XFORM 0             ; Push the current 2-D transformation onto the
                             stack.

! XFORM2D REL 0 .707 .707 -.707 .707

                           ; Define a relative 2-D transformation which
                             rotates by 45 degrees.  This transformation
                             is concatenated with the current transformation.
! SEGREF 1                 ; Draw geometry scaled by one half and rotated
                             by 45 degrees.
! POP XFORM 0              ; Pop the half scale transformation which was
                             saved on the stack.
! SEGREF 1                 ; Draw geometry scaled by one half and not
                             rotated.
```

PRMFIL

SYNTAX

ASCII	PRMFIL flag
FORTRAN Call	CALL PRMFIL (IFLAG)
Binary	[31] [flag] (2 bytes)
	31 decimal = 037 octal = 1F hex

FUNCTION

The PRMFIL command sets the primitive fill flag to indicate whether graphics primitives are drawn filled or unfilled. When flag = 1 or ON, any subsequent graphics primitives are drawn filled. The current pixel value is used for all interior pixels, as well as for the perimeter. When flag = 0 or OFF, just the perimeter is drawn in the current pixel value.

The graphics primitives which are affected by the PRMFIL flag are: ARC, CIRCI, CIRCLE, CIRCXY, POLYGN, RECREL, RECTAN, and RECTI.

You can use the VECPAT command to set patterned area and primitive fills.

ASCII PARAMETER

flag	Flag = 1 or ON enables filling; flag = 0 or OFF disables filling.

FORTRAN PARAMETER

INTEGER*2	IFLAG

EXAMPLE

```
! LUT8 1 0,0,255         ; Set the color out for LUT index 1 to blue.
! VAL8 1                 ; Set current pixel value to 1 (blue).
! PRMFIL ON              ; Select filled primitives.
! CIRCLE 50              ; Draw filled blue circle.
! MOVABS 100 100         ; Move current point to 100,100.
! PRMFIL OFF             ; Select unfilled primitives.
! CIRCLE 50              ; Draw unfilled blue circle.
```

ASCII SYNTAX

```
    PUSH  item, arg      (general form)

    PUSH  CREG, creg          (coordinate register)
    PUSH  VREG, vreg          (value register)
    PUSH  XFORM, xformtype    (transformation matrix)
    PUSH  VAR, vartype        (system variable or flag)
```

FORTRAN Calls

```
    CALL  PUSH (TYPE, CREG, VREG, XFORM, SYSVAR, FPREG)   (general form)

    CALL PUSH0 (CREG)
    CALL PUSH1 (VREG)
    CALL PUSH2 (XFORM)
    CALL PUSH3 (SYSVAR)
```

Binary

```
    [117] [item = 0] [creg]          (3 bytes)
    [117] [item = 1] [vreg]          (3 bytes)
    [117] [item = 2] [xformtype]     (3 bytes)
    [117] [item = 3] [vartype]       (3 bytes)

    117 decimal = 165 octal = 75 hex
```

FUNCTION

The PUSH command saves the specified item by pushing it onto the stack. You can then restore the item with the POP command.

You must pop stacked items in the exact opposite order as they were pushed.

The PUSH and POP commands perform stack overflow and underflow checking. An error is generated if you exceed the stack bounds.

No type checking is performed by the PUSH and POP commands. Invalid parameter values may produce unpredictable results.

FUNCTION, continued

The items that you can push and pop include

- the contents of registers
  - coordinate registers
  - value registers
- the current 2-D transformation matrix
- system variables or flags
  - primitive fill flag
  - vector pattern.

ASCII PARAMETERS

item	The item to be pushed.	
	0 = CREG	Coordinate register.
	1 = VREG	Value register.
	2 = XFORM	Transformation.
	3 = VAR	System variable or flag.
creg	Coordinate register number (1 byte). You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.	
vreg	Value register number (1 byte). You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.	
xformtype	Transformation type (1 byte).	
	0	2-D transformation matrix.
vartype	System variable or flag (1 byte).	
	0 = PRMFIL	Primitive fill flag.
	1 = VECPAT	Vector pattern.

FORTRAN PARAMETERS

CALL PUSH (TYPE, CREG, VREG, XFORM, SYSVAR, FPREG)

INTEGER*2 TYPE, CREG, VREG, XFORM, SYSVAR, FPREG

Note: The parameter FPREG does not apply to the Model One/10.

EXAMPLE

```
! XFORM2D ABS 0 .5 0 0 .5 0 0

                          ; Define an absolute 2-D transformation which
                            scales by one half.
! SEGREF 1                ; Draw geometry scaled by one half.
! PUSH XFORM 0            ; Push the current 2-D transformation onto the
                            stack.
! XFORM2D REL 0 .707 .707 -.707 .707

                          ; Define a relative 2-D transformation which
                            rotates by 45 degrees.  This transformation
                            is concatenated with the current transformation.
! SEGREF 1                ; Draw geometry scaled by one half and rotated
                            by 45 degrees.
! POP XFORM 0             ; Pop the half scale transformation which was
                            saved on the stack.
! SEGREF 1                ; Draw geometry scaled by one half and not
                            rotated.
```

SYNTAX

ASCII	QUIT
FORTRAN Call	CALL QUIT
Binary	[255] (1 byte)
	255 decimal = 377 octal = FF hex

FUNCTION

The QUIT command exits Graphics mode and returns to Alpha mode. This command should be used to return to Alpha mode when the host or local alphanumeric terminal is finished issuing graphics commands.

If the Model One has been configured for pure ASCII format (with the ASCII command), the system is restored to binary or ASCII hex format, as set with the SYSCFG command.

RDMASK RDMASK

SYNTAX

ASCII	RDMASK mask
FORTRAN Call	CALL RDMASK (MASK)
Binary	[158] [mask] (2 bytes)
	158 decimal = 236 octal = 9E hex

FUNCTION

The RDMASK command sets the read mask for the Model One's image memory planes. The 8-bit read mask is ANDed with the data from image memory immediately before the data enters the look-up table. If a bit is set (1), the corresponding bit plane is read-enabled; if the bit is cleared (0), the corresponding bit plane is not displayed.

The read mask corresponds directly to the eight bit planes of image memory. You can also set the read mask with the RMSK16 command.

The read mask may be used in conjunction with the write mask to store multiple images in image memory and select them for display. The WRMASK command sets the write mask.

The RDMASK and WRMASK commands can also be used for double buffering. The image memory on each memory board is divided into two 4-bit banks. If you use the WRMASK command to write enable bit planes in only one of these 4-bit banks, and the RDMASK command to read enable bit planes in only the other 4-bit bank, then the graphics processor will have full memory bandwidth into the selected bit planes.

ASCII PARAMETER

mask	The 8-bit read mask; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2	MASK

RDMODE

SYNTAX

ASCII	RDMODE flag
FORTRAN Call	CALL RDMODE (FLAG)
Binary	[211] [flag] (2 bytes)
	211 decimal = 328 octal = D3 hex

FUNCTION

The RDMODE command sets the format for reading back data to the host computer in response to a readback command. The two modes are ASCII decimal (flag = 0 or OFF) and binary (flag = 1 or ON).

You should use the default RDMODE OFF (ASCII mode). Binary mode is intended for DMA interfaces; the Model One/10 uses a serial interface.

When ASCII decimal data is sent, it is followed by a carriage return. When binary data is sent, no carriage return is included.

The commands which read back information are: READBU, READCR, READER, READP, READVR, READW, READWE and the Display List readback commands (RDPICK, RDPID, RDREG, RDXFORM, SEQINQ, and SYSTAT).

ASCII PARAMETER

flag	Flag = 0 or OFF, readback in ASCII decimal;
	flag = 1 or ON, readback in binary (default)

FORTRAN PARAMETER

INTEGER*2	FLAG

SYNTAX

ASCII	RDPICK type, startent, totalent
FORTRAN Call	CALL RDPICK (TYPE, START, NUM, PICENC, PICREL, PICIDS, SEGIDS, DEPTH, TREES)
Binary	[239] [type] [startent] [totalent] (6 bytes)
	239 decimal = 357 octal = EF hex

FUNCTION

The RDPICK command reads back information which is stored in the pick buffer. The command also updates the segment ID register (SEGREG) and the pick ID register (PIDREG) with the last entry in the pick buffer that is read back.

The type parameter specifies the type of information to read back. You can read back the following six types of information.

- The number of pick hits encountered and the number of pick hits recorded (this information is taken from counters; SEGREF and PIDREG are not updated).

- Pick ID numbers of specified pick hits (PIDREG is updated with last entry read back).

- Segment ID numbers (SEGREG is updated).

- Segment ID/pick ID pairs (SEGREG and PIDREG are updated).

- Unpacked trees of nested segment IDs and pick IDs (SEGREG and PIDREG are updated).

- Packed trees of nested segment IDs and pick IDs (SEGREF and PIDREG are updated).

If RDPICK fails to identify any geometry, it returns -1 as the segment ID and the pick ID.

The startent parameter specifies the pick buffer entry to start reading from.

FUNCTION, continued

The totalent parameter specifies the total number of entries to read back. If you specify 0, the Model One updates the registers with the specified entry (startent), but does not return any data.

The following are two examples of updating the registers without returning data.

- RDPICK SEGID 1 0	Update the segment ID register with the first entry in the pick buffer.
- RDPICK SEGPID 2 0	Update both registers with the second entry in the pick buffer.

Note: The parameter you select with this command (except for RDPICK PICKS) must correspond to a parameter you previously specified with SETGL PICKBUF. The specific type of information you ask for with RDPICK is stored in the pick buffer and therefore available for readback only after an appropriate choice with SETGL PICKBUF. For example, SETGL PICKBUF SEGPID saves information on segment IDs, pick IDs, and segment ID/pick ID pairs, but does not save information on tree hierarchies. On the other hand, SETGL PICKBUF TREE only saves information on tree hierarchies. Without your selecting a correct correspondence, the pick buffer will not contain the data you request and it therefore will return minus ones. This return could be confused with "no pick hits" or "no pickable entities" when, in fact, there was a hit.

ASCII PARAMETERS

type Type of information to be returned (8 bits).

0 = PICKS	Number of pick hits encountered (16 bits) and number of pick hits recorded (16 bits).
1 = PICKID	An array of pick ID numbers (32 bits).
2 = SEGID	An array of segment ID numbers (32 bits).
3 = SEGPID	An array of segment ID/pick ID pairs (32 bits).
4 = TREEU	Unpacked trees of nested segment ID/pick ID pairs (32 bits).
5 = TREEP	Packed trees of nested segment ID/pick ID pairs (32 bits).

ASCII PARAMETERS, continued

startent	Integer which specifies the pick buffer entry to start reading from (16 bits). With type 0 (PICKS), specify 0 (parameter is not relevant).
totalent	Integer which specifies the total number of entries to read back (16 bits). With type 0 (PICKS), specify 0 (parameter is not relevant). With types 1-5, specifying 0 updates the registers without returning data.

FORTRAN PARAMETERS AND SUBROUTINE CALLS

CALL RDPICK (TYPE,START,NUM,PICENC,PICREL,PICIDS,SECIDS,DEPTH,TREES)

Input Parameters:	INTEGER*2	TYPE, START, NUM
Output Parameters:	INTEGER*2	PICENC, PICREC
	INTEGER*4	PICIDS(1), SEGIDS(1), DEPTH(1)
	INTEGER*4	TREES(1,1)

The following subroutines are provided for convenience. These subroutines correspond to the ASCII types 0 through 5.

```
RDPCK0 (PICENC, PICREC)
RDPCK1 (START, NUM, PICIDS)
RDPCK2 (START, NUM, SEGIDS)
RDPCK3 (START, NUM, SEGIDS, PICIDS)
RDPCK4 (START, NUM, DEPTH, UTREES)
RDPCK5 (START, NUM, DEPTH, PTREES)
```

EXAMPLE

The following three examples illustrate many of the different ways that you can use the RDPICK command.

First we define Segment 1, which is a truck including a body, chassis, and two wheels. (This is the same segment that is defined in the SEGDEF command example, and is repeated here for easy reference.)

Following the segment definition, we set the center of the pick aperture, execute the segment in pick mode, and then execute the RDPICK command to read back information. We repeat this procedure three times with the pick aperture set in different locations.

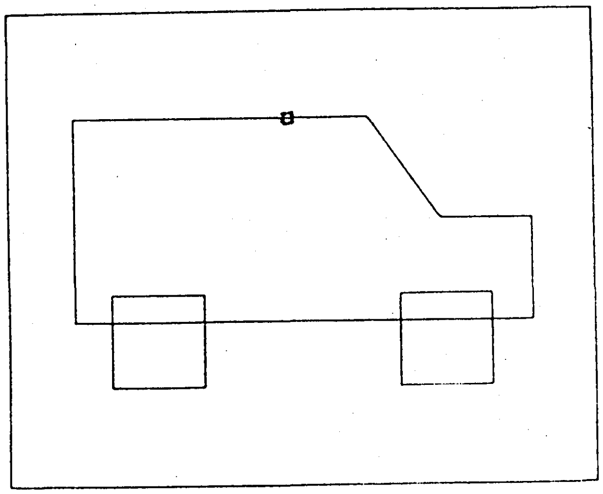
Segment Definition

```
! SEGINI 0 128          ; Initialize display list structures with a
                          block size of 128 bytes.
! SEGDEF 1              ; Begin definition of Segment 1, which defines
                          the truck and the position of the 2 wheels.
$ PICKID 10             ; Assign pick ID 10 label to the truck body.
$ PUSH CREG CURPNT      ; Push WCS current point onto the stack.
$ MOVABS -250,-80       ; Move current point to lower left corner of truck.
$ DRWREL 0,220          ; Draw 5 vectors to define truck body.
$ DRWREL 320,0
$ DRWREL 80,-110
$ DRWREL 100,0
$ DRWREL 0,-110
$ PICKID 20             ; End pick ID 10 (body); assign pick 20 label to
                          the chassis (bottom line) of truck.
$ DRWREL -500,0         ; Draw chassis of truck.
$ MOVABS -160,-100      ; Move current point to position rear wheel.
$ SEGREF 2              ; Nest Segment 2, the wheel.
$ MOVABS 155,-100       ; Move current point to position front wheel.
$ SEGREF 2              ; Nest Segment 2, the wheel.
$ POP CREG CURPNT       ; Pop WCS current point from the stack.
$ SEGEND                ; End pick ID 20 (chassis); end definition of
                          Segment 1, the truck body and position of wheels.

! SEGDEF 2              ; Begin definition of Segment 2, which defines the
                          shape of the wheel.
$ PICKID 30             ; Assign pick ID 30 label to the wheel.
$ PUSH CREG CURPNT      ; Save current point (center of wheel).
$ MOVREL -50,-50        ; Move current point to left corner of wheel.
$ RECREL 100,100        ; Draw wheel.
$ POP CREG CURPNT       ; Restore current point (center of wheel)
$ SEGEND                ; End pick ID 30 (wheel); end definition of
                          Segment 2, the wheel.
!                       ; Model One returns to normal command mode.
```

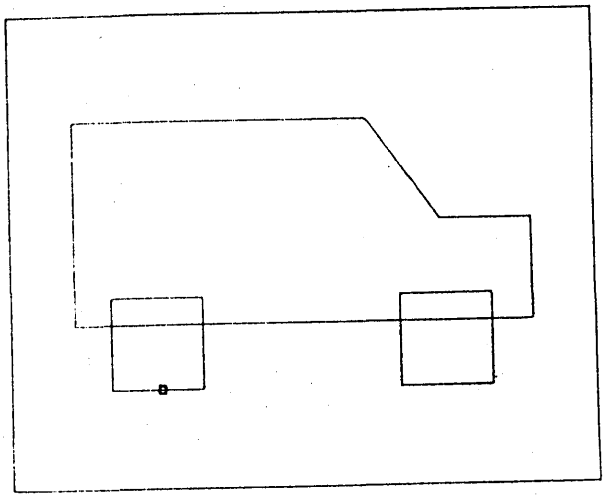
Drawing of Segment 1 with Pick Aperture Locations

The figures below show Segment 1, and the three different locations where we will set the pick aperture in the three examples which follow.

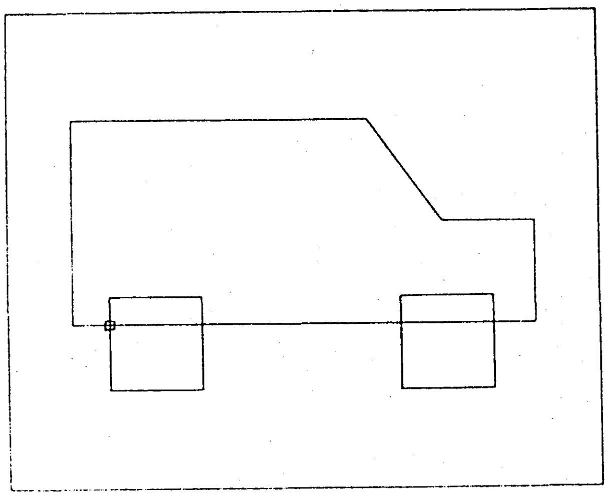
EXAMPLE 1: (0,140)



EXAMPLE 2: (-150,-150)



EXAMPLE 3: (-210,-80)



EXAMPLE 1: Reading Back One Pick Hit in Top Level Segment

```
! SETGL PICKAP 8 8       ; Set the pick aperture size to 16 x 16.
! SETGL PICKBUF SEGPID   ; Specify to save segment ID/pick ID pairs in
                           pick buffer (default).
! CLOAD 19 0,140         ; Specify the center of pick aperture. (See the
                           preceding figure labelled EXAMPLE 1.)
! EXMODE PICK            ; Set execution mode to pick.
! SEGREF 1               ; Execute (pick) Segment 1.
! RDPICK PICKS 0 0       ; Read back number of pick hits encountered and
                           number of pick hits recorded; registers are not
                           updated.
  00001 00001            ; The Model One returns:  1 pick hit encountered
                           and recorded.
! RDPICK PICKID 1 1      ; Read back pick ID; update PIDREG with pick ID.
 0000000010              ; The Model One returns: pick ID is 10.
! RDPICK SEGID 1 1       ; Read back segment ID; update SEGREG.
 0000000001              ; The Model One returns: segment ID is 1.
! RDPICK SEGPID 1 1      ; Read back segment ID/pick ID pairs; update
                           PIDREG and SEGREG.
 0000000001 0000000010   ; The Model One returns: segment ID is 1, pick ID
                           is 10.
! EXMODE NORMAL          ; Restore execution mode to normal draw.
```

EXAMPLE 2: Reading Back a Tree Hierarchy for One Pick Hit

```
! SETGL PICKBUF TREE     ; Specify to save entire trees in pick buffer.
! CLOAD 19 -150,-150     ; Specify the center of the pick aperture.  (See
                           the preceding figure labelled EXAMPLE 2.)
! EXMODE PICK            ; Set execution mode to pick.
! SEGREF 1               ; Execute (pick) Segment 1.
! RDPICK PICKS 0 0       ; Return number of pick hits encountered and
                           recorded.
  00001 00001            ; The Model One returns:  1 pick hit encountered
                           and recorded.
! RDPICK TREEP 1 1       ; Read back tree hierarchy in packed format.
  00002
 0000000001 0000000020 0000000002 0000000030

                         ; The Model One returns: one tree hit, nesting
                           level is two deep.  The top level is Segment 1,
                           pick ID 20. The second level is Segment 2, pick
                           ID 30.
! EXMODE NORMAL          ; Restore execution mode to normal draw.
```

EXAMPLE 3: Reading Back Tree Hierarchies for Two Pick Hits

```
! SETGL PICKBUF TREE      ; Specify to save entire trees in pick buffer.
! CLOAD 19 -210,-80       ; Set center of pick aperture.  (See preceding
                            figure labelled EXAMPLE 3.)
! EXMODE PICK             ; Set execution mode to pick.
! SEGREF 1                ; Execute (pick) Segment 1.
! RDPICK PICKS 0 0        ; Read back number of pick hits encountered and
                            recorded; registers are not updated.
  00002 00002             ; Model One returns:  2 pick hits encountered and
                            recorded.
! RDPICK TREEP 1 2        ; Read back tree hierarchy in packed format;
                            update registers.
  00001
 0000000001 0000000020
  00002
 0000000001 0000000020 0000000002 0000000030

                          ; The Model One returns:  two trees hit.  For first
                            tree, nesting level is one deep, the top level
                            is Segment 1, the hit was on graphics labelled
                            pick ID 20.  For second tree, nesting level is
                            two deep, the top level is Segment 1, pick ID 20,
                            and the second level is Segment 2, pick ID 30.
! EXMODE NORMAL           ; Restore execution mode to normal draw.
```

SYNTAX

ASCII	RDPID bf
FORTRAN Call	CALL RDPD80 (MAXSIZ, NVALS, BYTRAY)
Binary	[237] [0] (2 bytes)
	237 decimal = 355 octal = ED hex

FUNCTION

The RDPID command reads back all the commands which have a pick ID equal to PIDREG and a segment ID equal to SEGREG. The RDPID command returns a list of commands which contains opcodes and parameters. The information is in the format (n, nray), where n gives the number of bytes returned and nray is a list of primitives n bytes long. If PIDREG is equal to -1, then all commands in all the pick IDs in the current segment are returned.

For serial communications, the bf parameter (blocking factor) tells the Model One how many elements to send before inserting a carriage return into the output stream. If the end of the window is reached before the block is filled, the block is padded with zeros and sent. After sending each block, the Model One then waits for an ACK (06 hex or 86 hex) character from the host before sending out another block of data. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

ASCII PARAMETER

bf	Blocking factor (8 bits). Range is 1 to 255.

FORTRAN PARAMETERS

Input Parameter:	INTEGER*2	MAXSIZ
Output Parameters:	INTEGER*2	NVALS
	LOGICAL*1	BYTRAY(1)

MAXSIZE	The maximum size of the array in bytes.
NVALS	The number of elements returned.
BYTRAY	Array containing elements returned.

EXAMPLE

```
! SEGDEF 2
$ PICKID 30
$ PUSH CREG CURPNT
$ MOVREL -100 -100
$ RECREL 200 200
$ POP CREG CURPNT
$ SEGEND


! SETGL SEGREG 2           ; Load the segment ID register (SEGREG) with 2.
! SETGL PIDREG 30          ; Load the pick ID register (PIDREG) with 30.
! RDPID 10                 ; Read back commands with segment ID 2 and pick
                             ID 30.  Insert a carriage return after each 10
                             elements.
 00016
117 000 000 002 255 156 255 156 137 000
200 000 200 118 000 000 000 000 000 000

                           ; The 16 is the number of bytes returned.  The
                             following numbers are the opcodes and parameters
                             of the commands labelled pick ID 30 in Segment 2.
```

SYNTAX

ASCII	RDPIXR vreg	
FORTRAN Call	CALL RDPIXR (IVREG)	
Binary	[175] [vreg]	(2 bytes)
	175 decimal = 257 octal = AF hex	

FUNCTION

The RDPIXR command reads the pixel value from image memory at the current point (CREG 0) and places the value into the value register specified by vreg.

The Model One/10 stores three bytes of information, although it only uses the first 8 bits (i.e., the red component) for value information.

ASCII PARAMETER

vreg	Value register; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETER

INTEGER*2 IVREG

EXAMPLE:

```
! VAL8 45              ; Change current pixel value to 45.
! POINT                ; Set current point to current pixel value.
! RDPIXR 13            ; Read pixel value at current point and place
                         value in VREG 13.
! READVR 13            ; Display contents of VREG 13.
  045 000 000            (Response of Model One.)
```

SYNTAX

ASCII	RDREG reserved
FORTRAN Call	CALL RDREG (ISEG, IPID)
Binary	[224] [reserved = 0] (2 bytes)
	224 decimal = 340 octal = E0 hex

FUNCTION

The RDREG command reads back the current segment ID number (stored in SEGREG) and current pick ID number (stored in PIDREG), as set by one of these commands:

- EXMODE PICK followed by a SEGREF (picking)
- SETGL SEGREG and SETGL PIDREG, or
- RDPICK.

The RDREG command returns two 32-bit quantities.

This command can be used to determine if there was a pick hit after picking. If there was no pick hit, RDREG returns two negative ones.

ASCII PARAMETER

reserved	An 8-bit reserved parameter (must = 0).

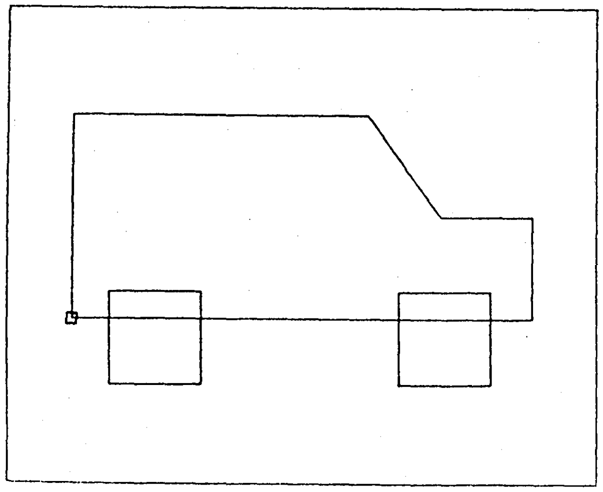
FORTRAN PARAMETER

Output Parameters: INTEGER*4 ISEG, IPID

ISEG	Current segment ID number.
IPID	Current pick ID number

EXAMPLE

This example illustrates how to use the RDREG command. It also shows how the EXMODE PICK and RDPICK commands update the segment ID and pick ID registers. The Segment 1 executed in this example is defined in the SEGDEF command example, and is illustrated below.



```
! SETGL PICKBUF SEGPID  ; Specify to save segment ID/pick ID pairs.
! CLOAD 19 -250,-80     ; Set center of pick aperture (see figure).
! EXMODE PICK           ; Enter pick mode.
! SEGREF 1              ; Execute (pick) Segment 1.
! RDPICK PICKS 0 0      ; Read back number of pick hits encountered and
                          number of pick hits recorded.
  00002 00002           ; The Model One returns:  two pick hits encountered
                          and recorded.
! RDREG 0               ; Read back contents of segment ID and pick ID
                          registers.
 0000000001 0000000010  ; The Model One returns: segment ID is 1, pick ID
                          is 10.  The registers are loaded with the first
                          pick hit.
! RDPICK SEGPID 1 2     ; Read back segment ID/pick ID pairs, starting with
                          the first pick buffer entry and returning infor-
                          mation on 2 entries.

 0000000001 0000000010 000000001 0000000020

! RDREG 0               ; Read back contents of segment ID and pick ID
                          registers.
 0000000001 0000000020  ; The Model One returns: segment ID is 1, pick ID
                          is 20.  The registers have been updated by the
                          RDPICK command with the last entry read back.
```

RDXFORM

SYNTAX

ASCII	RDXFORM type
FORTRAN Call	CALL RDXFRM (FUNC,MATRIX,XBASIS,YBASIS,ZBASIS,OFFSET)
Binary	[214] [type] (2 bytes)
	214 decimal =326 octal = D6 hex

FUNCTION

The RDXFORM command returns the matrix elements of the current 2-D transformation. The two translation elements are returned as 16-bit numbers. The four rotation/scaling elements are displayed (and returned in ASCII mode) as:

```
SIIIII.FFFFFF
```

where	S	is the sign of element: space if positive, "-" if negative
	I	is the integer part of the element
	.	is a decimal point
	F	is the fractional part of the element.

ASCII PARAMETERS

type The type of transformation to be returned (8 bits).

0	The current 2-D transformation.
1	Reserved.

FORTRAN SUBROUTINE CALLS AND PARAMETERS

```
CALL RDXFRM (FUNC, MATRIX, XBASIS, YBASIS, ZBASIS, OFFSET)

Input Parameter:    INTEGER*2   FUNC
Output Parameters:  REAL*4      MATRIX(1), XBASIS(3), YBASIS(3), ZBASIS(3),
                                OFFSET(3)
```

FUNC — The type of transformation to be returned. On the Model One/10, the only transformation type is 2-D (FUNC = 0).

MATRIX — A singly subscripted array containing a 2-D absolute transformation matrix converted from 16.16 fixed point to REAL*4.

```
    MATRIX is returned in 3 x 2 form so that
         MATRIX(1,1) =  x basis vector, x component
         MATRIX(1,2) =  x basis vector, y component
         MATRIX(2,1) =  y basis vector, x component
         MATRIX(2,2) =  y basis vector, y component
         MATRIX(3,1) =  displacement vector, x component
         MATRIX(3,2) =  displacement vector, y component
```

The parameters XBASIS, YBASIS, ZBASIS, and OFFSET do not apply to the Model One/10.

The following subroutine calls are provided for convenience.

```
CALL RDXFM0 (MATRIX)     (Read back 2-D transformation matrix)

CALL RDXF0X (VECTOR)     (Read back 2-D transformation matrix into a 6
                          element vector)

    VECTOR is returned so that
         VECTOR (1) =  x basis vector, x component
         VECTOR (2) =  x basis vector, y component
         VECTOR (3) =  y basis vector, x component
         VECTOR (4) =  y basis vector, y component
         VECTOR (5) =  displacement vector, x component
         VECTOR (6) =  displacement vector, y component
```

EXAMPLE

```
! XFORM2D ABS  0 2. 0 0 2. 0 0
                        ; Define an absolute 2-D transformation.
! RDXFORM 0             ; Read back the current 2-D transformation.
 00002.000000
 00000.000000
 00000.000000
 00002.000000
 00000
 00000
```

SYNTAX

ASCII	READBU flag, cflag
FORTRAN Call	CALL READBU (IFLAG, ICFLAG, IBUTT, X, Y)
Binary	[154] [flag] [cflag] (3 bytes)
	154 decimal = 232 octal = 9A hex

FUNCTION

The READBU command returns to the port in Graphics mode the function button number of a button that is pressed or released, as well as the location of the XY digitizer at the time the button was pressed or released. The READBU command removes one entry from the function buttón queue, which is 14 events deep.

The flag parameter indicates whether the Model One should respond immediately, and whether button releases should be recognized as well as button hits. If flag = 0 or OFF, the Model One responds immediately. If there is a button hit in the queue, the button number is sent along with the coordinates of the XY digitizer at the time the button was pressed. Otherwise, a button number of zero is sent along with the coordinates 0,0. If flag = 1 or ON, the Model One waits until there is a button hit entry in the button queue, and then sends the data. If flag = 2 or 3, the READBU comand works a similar way as with flag = 0 or 1 (i.e., flag = 2 and flag = 0 are similar), except that both button hits and releases are recognized with flag = 2 or 3.

The cflag parameter indicates whether the raw XY digitizer coordinate data (cflag = 0) or the scaled XY digitizer coordinate data (cflag = 1) should be returned. CREG 1 contains the unscaled coordinates and CREG 2 contains the scaled coordinates.

If RDMODE is 0 (which it should be), the function button number and coordinate data are sent in ASCII decimal format. The format is FORTRAN I3, 2I6 followed by a carriage return.

If the READBU command was sent from the host, the host must send an ACK (06 hex or 86 hex) to the Model One to resume normal command interpretation. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

ASCII PARAMETERS

flag	Event queue flag.	
	0 = OFF	Return next button hit immediately, or return button 0 if the event queue is empty.
	1 = ON	Return the next button hit as soon as there is one, and wait if necessary.
	2	Return the next button hit or release immediately, or return button 0 if the event queue is empty.
	3	Return the next button hit or release as soon as there is one, and wait if necessary.
cflag	Coordinate flag.	
	0 = OFF	Return raw coordinate data in CREG 1.
	1 = ON	Return scaled coordinate data in CREG 2.

FORTRAN PARAMETERS

Input Parameters:	INTEGER*2	IFLAG, ICFLAG
Output Parameters:	INTEGER*2	IBUTT
	INTEGER*4	X, Y

IBUTT returns the number of the button pressed or released. X and Y return the coordinates of the XY digitizer at the time the button was pressed or released.

EXAMPLE

The command sequence in this example is executed after the user has pressed button 4 once. The location of the XY digitizer at the time of the button press was at 10,-15.

```
! READBU 3 0             ; Wait until there is a button hit or release.
                           Return raw coordinate data in CREG 1.
  004 00010 -00015         (response from Model One.)
! READBU 3 0             ; Re-issue previous command again.  There is no
                           response until a button is pressed or released.
  -004 00010 -00015        (response from Model One indicating that button
                            4 was released).
```

READCR

SYNTAX

ASCII	READCR creg	
FORTRAN Call	CALL READCR (ICREG, IX, IY)	
Binary	[152] [creg]	(2 bytes)
	152 decimal = 230 octal = 98 hex	

FUNCTION

The READCR command sends the data in the coordinate register specified by creg to the port in Graphics mode. If the RDMODE is 0 (which it should be), the address is sent as two ASCII decimal numbers representing the x and y coordinates of the address. The numbers are sent in FORTRAN 2I6 format, followed by a carriage return.

If the command was issued by the host over the HOST interface, the Model One will wait for an acknowledge character (06 hex or 86 hex) from the host before command interpretation continues. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

ASCII PARAMETERS

creg	Coordinate register; range is 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

Input Parameter:	INTEGER*2	ICREG
Output Parameters:	INTEGER*2	IX, IY

IX and IY are returned from the subroutine with the x and y coordinates contained in the coordinate register specified by ICREG.

EXAMPLE

```
! CLOAD 23 110 200       ; Load CREG 23 with 110,200.
! READCR 23              ; Read contents of CREG 23.
  00110 00200              (Response from Model One.)
```

SYNTAX

ASCII	READER
FORTRAN Call	CALL READER (IERROR)
Binary	[56] (1 byte)
	56 decimal = 070 octal = 38 hex

FUNCTION

The READER command can be used to determine whether an error has occurred. The READER command returns a one-byte value giving the code of the first error which occurred since the last READER or COLDstart command.

If no error has occurred, the READER command returns zero. If the READER command returns "000", then you should assume that no error exists. However, there is an error message #000, "Illegal Call to routine ERROR" (which results from a firmware/hardware error), which is very unlikely to be generated. If an application fails and no other error messages besides "000" are generated, then the "000" may indicate an illegal call to ERROR.

The READER command clears the buffer which stores the error code.

Reference: The Model One Error Messages Reference Guide, included in the Model One/10 Manual, discusses each Model One/10 error message.

FORTRAN PARAMETER

LOGICAL*1 IERROR

IERROR is an output parameter which contains a one-byte error code.

SYNTAX

ASCII	READF func
FORTRAN Call	CALL READF (IFUNC)
Binary	[39] [func] (2 bytes)
	39 decimal = 047 octal = 27 hex

FUNCTION

The READF command controls the format and meaning of the data sent by the Model One when a READW and READWE command is executed. The parameter func specifies the format. The default setting for func is 0. The possible values for func are listed in the table below.

Important: You should use READF 1; the other format options are included for compatibility with other Model Ones. Use of other formats on the Model One/10 may cause unpredictable results.

Function	Data	ASCII decimal Format (RDMODE 0)	Binary Format (RDMODE 1)
0	Full 24-bit data	FORTRAN 3I3	3 binary bytes
1	Red component only	FORTRAN I3	1 binary byte
2	Green component only	FORTRAN I3	1 binary byte
3	Blue component only	FORTRAN I3	1 binary byte
4*	Packed r,g,b	FORTRAN I3	1 binary byte

Note: When func = 4, the data packing depends on whether RGBTRU is ON or OFF. With RGBTRU OFF, READF 4 returns only the first byte of the pixel value; the format is the same as READF 1. With RGBTRU ON, READF 4 packs the high 2 bits of each byte (r,g,b) into a single byte. This is the same RGB format used to send data to the Model One in the VAL1K command.

ASCII PARAMETER

func	Specifies the format; range is 0 to 4. You should use func = 1.

FORTRAN PARAMETER

INTEGER*2 IFUNC

EXAMPLE

```
! RUNLN8  10 20 255 99 48 99

                                  ; Draw a rectangle 10 rows high and 20 rows wide.
                                    The top 5 rows are white (LUT index 255) and the
                                    bottom 5 rows are red (LUT index 48).
! READF 1                         ; Set READWE format for 8-bit data (red component
                                    only).
! READWE 10 20 1                  ; Read a 10 by 20 window in run-length encoded form.
  255  099
  048  099                          (Response of Model One.)
```

READP READP

SYNTAX

ASCII	READP
FORTRAN Call	CALL READP (IRED, IGRN, IBLU)
Binary	[149] (1 byte)
	149 decimal = 225 octal = 95 hex

FUNCTION

The READP command returns the pixel value of the current point (CREG 0) to the port in Graphics mode. If RDMODE is set to 0 (which it should), the pixel value is returned as three ASCII decimal numbers representing the red, green, and blue components of the pixel value. The numbers are sent in FORTRAN 3I3 format, followed by a carriage return.

If the READP command was issued by the host, the Model One will wait for an ACK (06 hex or 86 hex) character from the host before continuing command interpretation. The acknowledge character must be sent from the host as a single control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

FORTRAN PARAMETERS

INTEGER*2 IRED, IGRN, IBLU

IRED, IGRN, AND IBLU are output parameters which contain the red, blue and green values of the pixel at the current point. The FORTRAN library handles transmission of the ACK character.

EXAMPLE

```
! LUT8 51 255,0,255      ; Set the color out for LUT index 51 to magenta.
! VAL8 51               ; Set current pixel value to 51 (magenta).
! FLOOD                 ; Flood displayed image memory to magenta.
! READP                 ; Read pixel value at current point.
  051 000 000             (Response from Model One.)
```

SYNTAX

ASCII	READVR vreg
FORTRAN Call	CALL READVR (IVREG, IRED, IGRN, IBLU)
Binary	[153] (1 byte)
	153 decimal = 231 octal = 99 hex

FUNCTION

The READVR command returns the pixel value in value register vreg to the port in Graphics mode. If RDMODE is set to 0 (which it should be), the pixel value is returned as three ASCII decimal numbers representing the red, green, and blue components of the pixel value. The numbers are sent in FORTRAN 3I3 format and are followed by a carriage return.

If the READVR command was issued by the host, the Model One will wait for an ACK (06 hex or 86 hex) character from the host before continuing command interpretation. The acknowledge character must be sent from the host as a single control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

ASCII PARAMETER

vreg	Value register; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

Input parameter:	INTEGER*2	IVREG
Output parameters:	INTEGER*2	IRED, IGRN, IBLU

IRED, IGRN, AND IBLU contain the red, blue and green components of the pixel value contained in the value register specified by IVREG.

The FORTRAN library handles transmission of the ACK character.

EXAMPLE

```
! VLOAD 3 35 0 0        ; Load VREG 3 with 35,0,0.
! READVR 3              ; Read contents of VREG 3.
  035 000 000             (Response from Model One.)
```

READW READW

SYNTAX

ASCII	READW nrows, ncols, bf
FORTRAN Call	CALL READW (NROWS, NCOLS, IRED, IGRN, IBLU)
Binary	[150] [highnrows][lownrows] [highncols][lowncols] [bf] (6 bytes)
	150 decimal = 226 octal = 96 hex

FUNCTION

The READW command instructs the Model One to read back a rectangular array of pixels to the port in Graphics mode. Nrows and ncols specify the number of rows and columns to read. The current point is used as the upper left-hand corner of the window. The window is scanned left to right and top to bottom.

The pixel values are sent to the host in the format set by the READF command. The RDMODE command determines whether the data is returned in binary or ASCII decimal format (you should use RDMODE 0 (ASCII format)).

The bf parameter (blocking factor) tells the Model One how many pixel values to send before inserting a carriage return into the output stream. If the end of the window is reached before the block is filled, the block is padded with zeros and sent. After sending each block, the Model One then waits for an ACK (06 hex or 86 hex) character from the host before sending out another block of data. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

Note: The READW command sends data to the host in the same format that the PIXELS and PIXEL8 commands use to send data to the Model One.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns; the range is 1 to 512 for nrows and 1 to 1024 for ncols.
bf	8-bit integer specifying the blocking factor.

FORTRAN PARAMETERS

CALL READW (NROWS, NCOLS, IRED, IGRN, IBLU)

Input parameters:	INTEGER*2	NROWS, NCOLS

Output parameters:	INTEGER*2	IRED(NROWS*NCOLS)
		IGRN(NROWS*NCOLS)
		IBLU(NROWS*NCOLS)

IRED, IGRN, and IBLU are byte (8-bit) arrays which contain the pixel values returned by the subroutine call. The arrays should be declared in the main program to a dimension at least as large as NROWS * NCOLS.

The FORTRAN library handles transmission of the ACK character.

Additional Subroutines

The READW subroutine is a shell which calls a series of lower level routines to do its work. The lower level routines are functionally independent and may be called themselves. READW calls one of the following routines, depending on the value of the readback format (IFMT) in the RASTEK common block. The readback format is changed by calling the READF subroutine.

CALL READW0 (NROWS, NCOLS, IRED, IGRN, IBLU)	Full 24-bit data (IFMT = 0)
CALL READWR (NROWS, NCOLS, IRED)	Red component only (IFMT = 1)
CALL READWG (NROWS, NCOLS, IGRN)	Green component only (IFMT = 2)
CALL READWB (NROWS, NCOLS, IBLU)	Blue component only (IFMT = 3)
CALL READW4 (NROWS, NCOLS, IVAL)	Packed r,g,b data (IFMT = 4)

Note: For READW4, IVAL is declared LOGICAL*1. The format of the data in IVAL depends on whether RGBTRU is ON or OFF. Refer to the READF command for information on the format.

SYNTAX

ASCII	READWE nrows, ncols, bf
FORTRAN Call	CALL READWE (NROWS, NCOLS, IDATA, NBYTES)
Binary	[151] [highnrows][lownrows] [highncols][lowncols] [bf] (6 bytes)
	151 decimal = 227 octal = 97 hex

FUNCTION

The READWE command instructs the Model One to read back a rectangular array of pixels to the port in Graphics mode. The Model One sends the data in a run-length encoded format. This format consists of a pixel value followed by a count of the horizontal pixels in a row which are set to that value. The count is set to one less than the number of pixels set to the value.

Nrows and ncols specify the number of rows and columns to read. The current point is used as the upper left corner of the window. The window is scanned left to right and top to bottom.

The pixel values are sent to the host in the format set by the READF command. The RDMODE command determines whether the data is returned in binary or ASCII decimal format (you should use RDMODE 0 (ASCII format)).

For serial communications, the bf parameter (blocking factor) tells the Model One how many pixel and count pairs to send before inserting a carriage return into the output stream. If the end of the window is reached before the block is filled, the block is padded with zeros and sent. After sending each block, the Model One then waits for an ACK (06 hex or 86 hex) character from the host before sending out another block of data. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

Note: The READWE command sends data to the host in the same format that the RUNLEN and RUNLN8 commands use to send data to the Model One.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns; range is 1 to 512 for nrows and 1 to 1024 for ncols.
bf	8-bit integer specifying the blocking factor.

FORTRAN PARAMETERS

CALL READWE (NROWS, NCOLS, IDATA, NBYTES)

Input parameters:	INTEGER*2	NROWS, NCOLS
Output parameters:	LOGICAL*1	IDATA(1)
	INTEGER*4	NBYTES

IDATA is a byte (8-bit) array which contains the run-length encoded data. The format of the data in IDATA varies depending on the value of the readback format (IFMT) in the RASTEK common block. The readback format is changed by calling the READF subroutine. If IFMT = 0 (indicating full color 24-bit data), then IDATA should be dimensioned in the main program to be at least NROWS * NCOLS * 4. For all other values of IFMT, IDATA should be dimensioned to be at least NROWS * NCOLS * 2.

NBYTES returns the number of elements in IDATA.

Additional Subroutines

READWE is a shell which calls a series of lower level routines to do its work. The lower level routines are functionally independent and may be called by themselves. READWE calls one of the following routines, depending on the value of the readback format (IFMT) in the RASTEK common block. The readback format is changed by calling the READF subroutine.

CALL RDWE0 (NROWS, NCOLS, IDATA, NBYTES)	Full 24-bit data (IFMT = 0)
CALL RDWER (NROWS, NCOLS, IRED, NBYTES)	Red component only (IFMT = 1)
CALL RDWEG (NROWS, NCOLS, IGRN, NBYTES)	Green component only (IFMT = 2)
CALL RDWEB (NROWS, NCOLS, IBLU, NBYTES)	Blue component only (IFMT = 3)

FORTRAN SUBROUTINES AND PARAMETERS, continued

Additional Subroutines, continued

CALL RDWE4 (NROWS, NCOLS, IVAL, NBYTES)	Packed r,g,b data (IFMT = 4)

Note: For RDWE4, IVAL is declared LOGICAL*1. The format of the data in IVAL depends on whether RGBTRU is ON or OFF. Refer to the READF command for information on the format.

EXAMPLE

```
! RUNLN8  10 20 3 99 48 99      ; Draw a rectangle.  The top 5 rows are blue
                                  and the bottom 5 rows are red.
! READWE 10 20 1                ; Read a 10 by 20 window in run-length
                                  encoded form.
  003 000 000 099                 (Response from Model One.)
  048 000 000 099

! READF 1                       ; Set READWE format for 8-bit data (red
                                  component only).
! READWE 10 20 1                ; Read same window.
  003 099                         (Response from Model One.)
  048 099
```

SYNTAX

ASCII	RECREL dx, dy
FORTRAN Call	CALL RECREL (IDX, IDY)
Binary	[137] [highdx][lowdx] [highdy][lowdy] (5 bytes)
	137 decimal = 211 octal = 89 hex

FUNCTION

The RECREL command draws a rectangle in image memory with one corner at the WCS current point (CREG 0) and the diagonally opposite corner displaced from the current point by dx,dy. The rectangle is drawn in the current pixel value (VREG 0). The current point is unchanged.

Note: A vector with a length that exceeds 32,767 units will not be drawn correctly.

ASCII PARAMETERS

dx, dy	16-bit integers specifying the displacement from the current point of the opposite corner of the rectangle; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2	IDX, IDY

EXAMPLE

```
! MOVABS 100 150        ; Move the current point to 100,150.
! RECREL 10 10          ; Draw rectangle with diagonally opposite corner
                          displaced from current point by 10,10 (at 110,160).
! RECREL -20 -30        ; Draw rectangle with diagonally opposite corner
                          displaced from current point by -20,-30
                          (at 80,120). The two rectangles intersect at
                          the current point, which has not changed.
```

SYNTAX

ASCII	RECTAN x,y
FORTRAN Call	CALL RECTAN (IX, IY)
Binary	[142] [highx][lowx] [highy][lowy] (5 bytes)
	142 decimal = 216 octal = 8E hex

FUNCTION

The RECTAN command draws a rectangle in image memory with one corner at the WCS current point (CREG 0) and the diagonally opposite corner at the point specified by x,y. The current point is unchanged.

Note: A vector with a length that exceeds 32,767 units will not be drawn correctly.

ASCII PARAMETERS

x, y	16-bit integers specifying the coordinates for the diagonally opposite corner of the rectangle; range is from -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2 IX, IY

EXAMPLE

```
! MOVABS 30 50         ; Move current point to 30,50.
! RECTAN 70 100        ; Draw rectangle with opposite corners located at
                         30,50 and 70,100.
! RECTAN -70 -100      ; Draw rectangle with opposite corners located at
                         30,50 and -70 -100.  The two rectangles intersect
                         at the current point, which has not changed.
```

RECTI

SYNTAX

ASCII	RECTI creg	
FORTRAN Call	CALL RECTI (ICREG)	
Binary	[143] [creg]	(2 bytes)
	143 decimal = 217 octal = 8F hex	

FUNCTION

The RECTI command draws a rectangle with one corner at the current point (CREG 0) and the diagonally opposite corner at the point specified by coordinate register creg. The current point is unchanged.

Note: A vector with a length that exceeds 32,767 units will not be drawn correctly.

ASCII PARAMETERS

creg	Coordinate register; range is 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

INTEGER*2	ICREG

EXAMPLE

```
! MOVABS 0 0            ; Move current point to 0,0.
! CLOAD 17 50 50        ; Load CREG 17 with 50,50.
! RECTI 17              ; Draw rectangle with opposite corners at
                          0,0 (current point) and 50,50 (CREG 17).

! MOVABS 200,100        ; Move current point to 200,100.
! RECTI 17              ; Draw rectangle with opposite corners at
                          200,100 (current point) and 50,50 (CREG 17).
```

SYNTAX

ASCII	REPLAY
FORTRAN Call	CALL REPLAY
Binary	[188] (1 byte)
	188 decimal = 274 octal = BC hex

FUNCTION

The REPLAY command sends a dump of the last 32 characters sent by the host over the HOST interface to the local alphanumeric display screen. The characters are displayed in ASCII hexadecimal format. The last character output is the last character that was sent by the host.

EXAMPLE

```
! REPLAY                  ; Dump last 32 characters of HOST input buffer
                            to the screen.

00 FF F0 FD E0 E2 20 40     (Possible response from Model One.)
30 3F E3 20 21 31 00 00
33 53 E5 25 20 32 37 70
7F FF FF FF 30 3F 55 F5
```

SYNTAX

ASCII	RGBTRU flag
FORTRAN Call	CALL RGBTRU (FLAG)
Binary	[78] [flag] (2 bytes)
	78 decimal = 116 octal = 4E hex

FUNCTION

The RGBTRU ON command can be used on the Model One/10 to allow existing 24-bit images (e.g., from the Model One/25, Model One/80, or Model One/380) to be loaded into the Model One/10's image memory. The value that is loaded is parsed; the top two bits of each byte of the value are packed into the single byte used to determine the LUT entry to display. The resulting default is equivalent to the VAL1K 0 to 63 entries in the LUT. The 24-bit images lose detail and shading and become more striped in appearance. However, the colors are similar.

RGBTRU OFF is the default.

ASCII PARAMETER

flag	Flag = 1 or ON, enable RGBTRU mode; flag = 0 or OFF, disable RGBTRU mode.

FORTRAN PARAMETER

INTEGER*2	FLAG

RMSK16 RMSK16

SYNTAX

ASCII	RMSK16 mask	
FORTRAN Calls	CALL RMSK16 (MASK)	
	CALL RMK16S (MASK)	Swaps high and low bytes
Binary	[67] [highmask] [lowmask]	(3 bytes)
	67 decimal = 103 octal = 43 hex	

FUNCTION

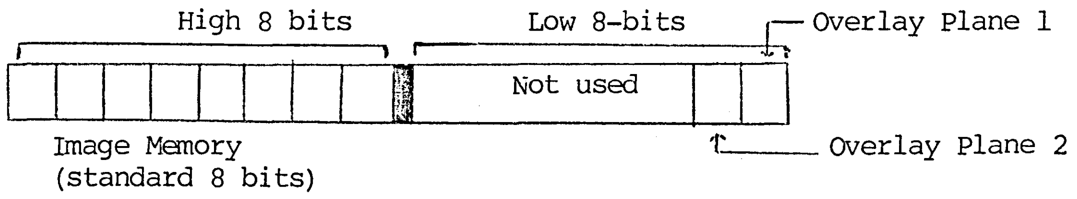
The RMSK16 command specifies a 16-bit read mask for the Model One's image memory planes. Only the high byte of the read mask is used. The high eight bits correspond to the eight bit planes of image memory. The low byte of the mask is used to control the overlays.

The read mask is ANDed with the data from image memory immediately before the data enters the look-up table. If a bit is set (1), the corresponding bit plane is read-enabled; if the bit is cleared (0), the corresponding bit plane is not displayed.

When used in conjunction with the WMSK16 command (write mask), the RMSK16 command can be used for double buffering and animation by writing into those bit planes not displayed and displaying those bit planes not being written into.

The read and write masks can also be used to store multiple images in image memory and select them for display.

The format for the mask is shown below:



FUNCTION, continued

The RMSK16 command swaps the high (left) and low (right) bytes that make up the 16-bit word used for the mask. In other words, 52 in hexadecimal must be entered as #5200. You should use a hexadecimal number for the mask.

Note: The FORTRAN subroutine RMK16S performs the swapping for you, so you can specify the mask without swapping the bytes.

ASCII PARAMETER

mask	The 16-bit read mask; range is 0 to 65,535 (#0000 to #FFFF).

FORTRAN PARAMETER

INTEGER*2	MASK

EXAMPLE

```
! MOVABS 0 0          ; Move current point to 0,0.
! PRMFIL ON           ; Select filled primitives.
! WMSK16 #0100        ; Write enable only bit plane 0.
! LUT8 1 255 0 0      ; Change the color out for LUT index 1 to red.
! VAL8 1              ; Set current pixel value to 1 (255,0,0: red).
! RECTAN 100 100      ; Draw a filled red square.

! MOVABS 200 200      ; Move current point to 200,200.
! WMSK16 #1000        ; Write enable only bit plane 4.
! LUT8 16 0 255 0     ; Change the color out for LUT table index 16 to
                        green.
! VAL8 16             ; Set current pixel value to 16 (0,255,0: green).
! CIRCLE 50           ; Draw green circle.

! RMSK16 #0100        ; Read enable only bit plane 0; only the red
                        square is displayed.
! RMSK16 #1000        ; Read enable only bit plane 4; only the green
                        circle is displayed.
! RMSK16 #1100        ; Read enable bit planes 0 and 4; both the circle
                        and the square are displayed.
```

SYNTAX

ASCII	RUNLEN nrows, ncols, r, g, b, cnt, ...
FORTRAN Call	CALL RUNLEN (NROWS, NCOLS, IDATA)
Binary	[42] [highnrows][lownrows] [highncols][lowncols] ([r] [g] [b] [cnt]) ... (5 + 4 * number of runs bytes)
	42 decimal = 052 octal = 2A hex

FUNCTION

The RUNLEN command transmits a run-length encoded image to the Model One/10. The array of transmitted data is nrows high and ncols wide. The location of the upper left corner of the array is defined by the current point (CREG 0). Pixels in image memory are filled left to right and top to bottom. Each pixel value is sent as a full color 24-bit quantity, one byte each of red, green, and blue.

Each pixel value is followed by a one byte count parameter, cnt, which specifies the number of horizontally contiguous pixels which are to be set the given r,g,b value. The count is one less that the number of pixels to be set. If cnt = 0, one pixel is set, if cnt = 1, two pixels are set; the range is up to cnt = 255, where 256 pixels are set.

Only the red byte is used to determine the location in the look-up table. For this reason, the RUNLEN command is very inefficient and you should only use it if you require compatibility with other Model Ones. Otherwise, use the RUNLN8 command.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is from 0 to 32,767
r, g, b	8-bit values specifying the red, green, and blue components of the pixel value.
cnt	8-bit value specifying the number of horizontally contiguous pixels to be set to the r,g,b value; range is 0 to 255.

FORTRAN PARAMETERS

```
INTEGER*2     NROWS, NCOLS
LOCICAL*1     IDATA(NROWS*NCOLS)
```

IDATA is a byte (8-bit) array which contains the run-length encoded data. The array must contain enough data to fill a window NROWS by NCOLS.

EXAMPLE

```
! MOVABS 0 0                ; Move the current point to 0,0.

! RUNLEN 10 20 48,0,0 99 3,0,0 99

                            ; Draw a rectangle 10 rows high and 20 rows wide.
                              The top 5 rows are red and the bottom 5 rows
                              are blue.
```

RUNLN8 RUNLN8

SYNTAX

ASCII	RUNLN8 nrows, ncols, val, cnt, ...
FORTRAN Call	CALL RUNLN8 (NROWS, NCOLS, IDATA)
Binary	[43] [highnrows][lownrows] [highncols][lowncols] ([val] [cnt]) ... (5 + 2 * number of runs bytes)
	43 decimal = 053 octal = 2B hex

FUNCTION

The RUNLN8 command transmits a run-length encoded image to the Model One/10. The array of transmitted data is nrows high and ncols wide. The location of the upper left corner of the array is defined by the current point (CREG 0). Pixels in image memory are filled left to right and top to bottom. Each pixel value is sent as an 8-bit quantity which is used as an index into the look-up table.

Each pixel value is followed by a one byte count parameter, cnt, which specifies the number of horizontally contiguous pixels which are to be set the value given by val. The count should be one less than the number of pixels to be set. If cnt = 0, one pixel is set, if cnt = 1, two pixels are set; the range is up to cnt = 255, where 256 pixels are set.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is from 0 to 32,767
val	8-bit parameter specifying the pixel value.
cnt	8-bit value specifying the number of horizontally contiguous pixels to be set to the pixel value; range is 0 to 255.

FORTRAN PARAMETERS

Input Parameter

```
INTEGER*2     NROWS, NCOLS
```

Output Parameter

```
LOCICAL*1     IDATA(NROWS*NCOLS)
```

IDATA is a byte (8-bit) array which contains the run-length encoded data. The array must contain enough data to fill a window NROWS by NCOLS.

EXAMPLE

```
! MOVABS 0 0             ; Move the current point to 0,0.

! RUNLN8 10 20 3,99 48,99

                        ; Draw a rectangle 10 rows high and 20 rows wide.
                          The top 5 rows are blue (LUT index 3) and the
                          bottom 5 rows are red (LUT index 48).
```

RUNLN16 RUNLN16

SYNTAX

ASCII	RUNLN16 nrows, ncols, val, cnt, ...
FORTRAN Call	CALL RUNLN16 (NROWS, NCOLS, IDATA)
Binary	[71] [highnrows][lownrows] [highncols][lowncols] ([val] [cnt]) ... (5 + 2 * number of runs bytes)
	71 decimal = 107 octal = 47 hex

FUNCTION

The RUNLN16 command transmits a run-length encoded image to the Model One/10. The array of transmitted data is nrows high and ncols wide. The location of the upper left corner of the array is defined by the current point (CREG 0). Pixels in image memory are filled left to right and top to bottom. Each pixel value is sent as a 16-bit quantity which is used as an index into the look-up table, as with the VAL16 command.

Each pixel value is followed by a one byte count parameter, cnt, which specifies the number of horizontally contiguous pixels which are to be set the value given by val. The count should be one less than the number of pixels to be set. If cnt = 0, one pixel is set, if cnt = 1, two pixels are set; the range is up to cnt = 255, where 256 pixels are set.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is from 0 to 32,767
val	16-bit parameter specifying the pixel value.
cnt	8-bit value specifying the number of horizontally contiguous pixels to be set to the pixel value; range is 0 to 255.

FORTRAN PARAMETERS

Input Parameters:	INTEGER*2	NROWS, NCOLS
Output Parameter:	INTEGER*1	IDATA(NROWS*NCOLS)

IDATA is a byte (8-bit) array which contains the run-length encoded data. The array must contain enough data to fill a window NROWS by NCOLS.

EXAMPLE

```
! MOVABS 0 0              ; Move the current point to 0,0.

! RUNLN16 10 20 3,99 48,99

                          ; Draw a rectangle 10 rows high and 20 rows wide.
                            The top 5 rows are blue (LUT index 3) and the
                            bottom 5 rows are red (LUT index 48).
```

SAVCFG

ASCII SYNTAX

```
SAVCFG
```

FUNCTION

The SAVCFG command saves the Model One/10 port configurations defined with the SYSCFG command. SAVCFG saves all port configurations; any port configurations that were not changed by SYSCFG, however, are not changed by SAVCFG. In addition, SAVCFG saves definitions of special characters, and the current settings of the ALPHEM and RGBTRU commands.

The SAVCFG command stores the current configuration in NVRAM (non-volatile RAM). The configuration saved with the SAVCFG command will be in effect when

- the system is powered on or off
- the RESET button is pressed, or
- the COLD (COLDstart) command is executed.

You can use the DFTCFG command to restore all ports to the default configuration, stored in PROM (programmable read-only memory).

To display the current configuration, use the DISCFG command.

Notes: The SAVCFG command can be executed only from the local alphanumeric terminal, and cannot be included in a macro.

You may prefer to use the Setup mode, entered by pressing the SETUP key on the keyboard. Setup mode is fully documented in Section 13 of the Model One/10 Introduction and Installation Guide, Rev. 2.0.

EXAMPLE

```
! SYSCFG SERIAL HOST PARITY N     ; Configures the serial port HOST.
Are you sure??                    ; Prompt from the Model One asking for
                                    confirmation of the configuration.
y                                 ; User response to prompt, confirming
                                    the new configuration is as desired.
! SAVCFG                          ; Saves the configuration defined
                                    by the SYSCFG command.
```

SCRORG SCRORG

SYNTAX

ASCII	SCRORG x,y	
FORTRAN Call	SCRORG (IX, IY)	
Binary	[54] [highx][lowx] [highy][lowy]	(5 bytes)
	54 decimal = 066 octal = 36 hex	

FUNCTION

The SCRORG command sets the screen origin register (CREG 4) to the point specified by x,y. The screen origin specifies the coordinate in image memory that will be displayed at the center of the screen. This command is used to pan the displayed image.

Notes:

Changes to the screen origin will not affect the hardware cursor for the scrolling text.

Use a WAIT 0 when SCRORG immediately follows a MODDIS, if the commands are issued from the host.

ASCII PARAMETERS

x, y	16-bit parameters specifying the screen origin coordinate; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2	IX, IY

EXAMPLE

```
! MOVABS 0 0             ; Move current point to 0,0.
! CIRCLE 50              ; Draw a circle in the center of the screen.
! SCRORG 300 300         ; Set screen origin to 300,300.  Circle is
                           displayed in lower left corner of screen.
! SCRORG -300 -300       ; Set screen origin to -300,-300.  Circle is
                           displayed in upper right corner of screen.
! SCRORG 0 0             ; Restore screen origin to 0,0.  Circle is
                           displayed in center of screen.
```

SEGAPP SEGAPP

SYNTAX

ASCII	SEGAPP segment
FORTRAN Call	CALL SEGAPP (SEGID)
Binary	[219] [segment3] [segment2] [segment1] [segment0] (5 bytes)

219 decimal = 333 octal = DB hex

FUNCTION

The SEGAPP command reopens an existing segment definition for modification. The commands you input after SEGAPP are appended to the specified segment definition. After you have input all the commands you wish to add, use the SEGEND command to close the segment.

ASCII PARAMETER

segment	The number of the segment to which you want to append commands (32 bits).

FORTRAN PARAMETER

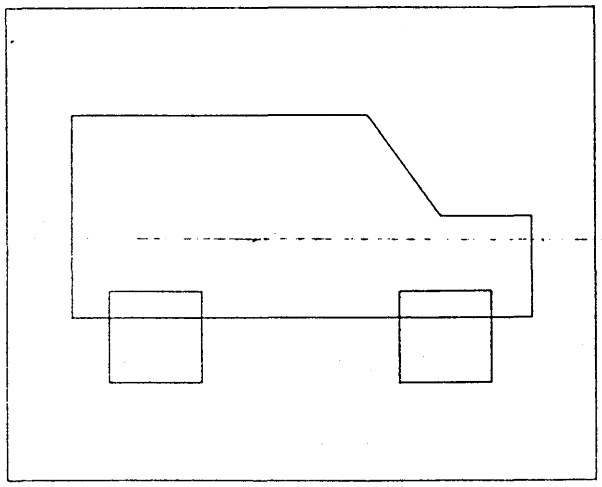
INTEGER*4	SEGID

EXAMPLE

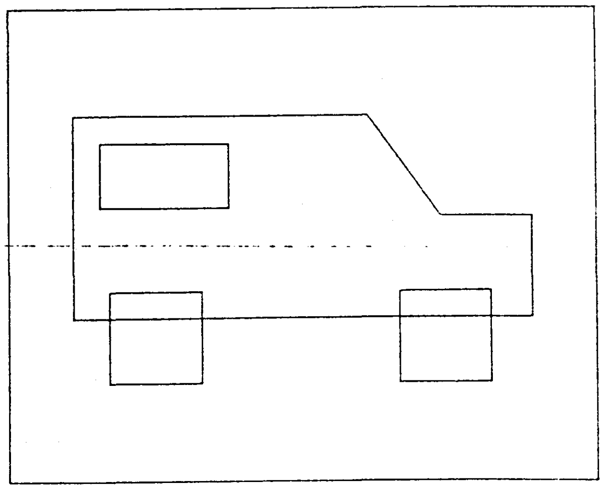
In this example, Segment 1 is reopened and a window is added to the truck. Refer to the example for the SEGDEF command to see the definition of Segment 1.

```
! SEGREF 1               ; Execute (draw) Segment 1, a truck with 2 wheels.
! SEGAPP 1               ; Reopen Segment 1 to append commands.
$ PUSH CREG CURPNT       ; Push WCS current point back onto the stack.
$ PICKID 40              ; Assign pick ID 40 to the window.
$ MOVABS -220,40         ; Move current point to lower left corner of
                           window.
$ RECREL 140,70          ; Draw window.
$ POP CREG CURPNT        ; Pop WCS current point from the stack.
$ SEGEND                 ; Close Segment 1.
! VAL8 0                 ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1               ; Execute (draw) appended Segment 1, now a truck
                           with 2 wheels and a window.
```

BEFORE THE APPEND



AFTER THE APPEND



SYNTAX

ASCII	SEGCOP segment2, segment1
FORTRAN Call	CALL SEGCOP (SEGDST, SEGSRC)
Binary	[231] [segment23] [segment22] [segment21] [segment20] [segment13] [segment12] [segment11] [segment10] (9 bytes)

231 decimal = 347 octal = E7 hex

FUNCTION

The SEGCOP command copies segment1 into segment2. If segment2 already exists, it is overwritten. In either case, segment1 remains unchanged.

ASCII PARAMETERS

segment1	The segment number of the source segment (32 bits).
segment2	The segment number of the destination segment (32 bits).

FORTRAN PARAMETERS

INTEGER*4 SEGDST, SEGSRC

EXAMPLE

```
! SEGCOP 3 2          ; Create Segment 3; copy Segment 2 into Segment 3.
```

SEGDEF SEGDEF

SYNTAX

ASCII	SEGDEF segment
FORTRAN Call	CALL SEGDEF (SEGID)
Binary	[220] [segment3] [segment2] [segment1] [segment0] (5 bytes)
	220 decimal = 334 octal = DC hex

FUNCTION

The SEGDEF command opens a segment definition. Segment, a 32-bit number, specifies the new segment number. Segments can contain a variety of different commands, including

- commands that draw 2-D graphics primitives
- commands that move the current point
- commands that change the current color
- commands that change primitive-generation attributes, such as VECPAT and PRMFIL
- the SEGREF command to nest (reference) child segments, and
- the PICKID command.

Do not put readback commands (e.g., READBU, READP, etc.) in segments. Readback commands in a display list segment will hang the system when the segment is referenced.

You must close every segment definition with the SEGEND command. By default, segments are both visible and pickable upon creation. If you want to change either of these attributes, use the SETATR command. To draw and pick segments after defining them, use the SEGREF command.

ASCII PARAMETER

segment	The segment number; range is 00000000 to FBFFFFFF hex (32 bits). Segment numbers in the range FC000000 to FFFFFFFF (hex) are reserved.

FORTRAN PARAMETER

INTEGER*4 SEGID

EXAMPLE

This example illustrates the definition of two segments, Segment 1 and Segment 2. Segment 1 is the top-level segment, and Segment 2 is nested (or referenced) within Segment 1.

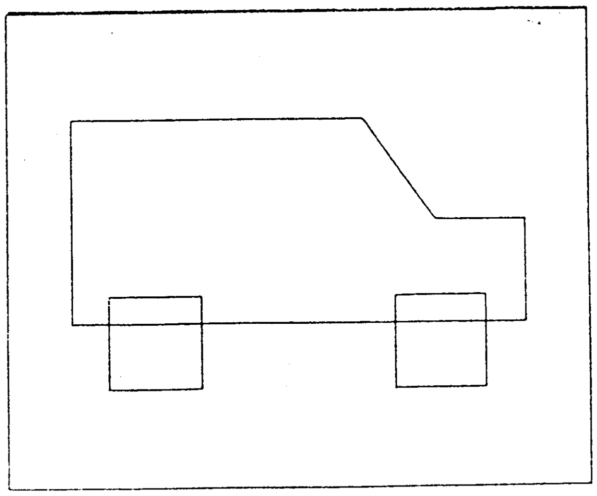
Note that the SEGDEF command changes the Model One prompt to "$" which indicates segment definition mode.

```
! SEGINI 0 128          ; Initialize display list structures with a
                          block size of 128 bytes.
! SEGDEF 1              ; Begin definition of Segment 1, which defines
                          the truck and the position of the 2 wheels.
$ PICKID 10             ; Assign pick ID 10 label to the truck body.
$ PUSH CREG CURPNT      ; Push WCS current point onto the stack.
$ MOVABS -250,-80       ; Move current point to lower left corner of truck.
$ DRWREL 0,220          ; Draw 5 vectors to define truck body.
$ DRWREL 320,0
$ DRWREL 80,-110
$ DRWREL 100,0
$ DRWREL 0,-110
$ PICKID 20             ; End pick ID 10 (body); assign pick 20 label to
                          the chassis (bottom line) of truck.
$ DRWREL -500,0         ; Draw chassis of truck.
$ MOVABS -160,-100      ; Move current point to position rear wheel.
$ SEGREF 2              ; Nest Segment 2, the wheel.
$ MOVABS 155,-100       ; Move current point to position front wheel.
$ SEGREF 2              ; Nest Segment 2, the wheel.
$ POP CREG CURPNT       ; Pop WCS current point from the stack.
$ SEGEND                ; End pick ID 20 (chassis); end definition of
                          Segment 1, the truck body and position of wheels.

! SEGDEF 2              ; Begin definition of Segment 2, which defines the
                          shape of the wheel.
$ PICKID 30             ; Assign pick ID 30 label to the wheel.
$ PUSH CREG CURPNT      ; Save current point (center of wheel).
$ MOVREL -50,-50        ; Move current point to left corner of wheel.
$ RECREL 100,100        ; Draw wheel.
$ POP CREG CURPNT       ; Restore current point (center of wheel)
$ SEGEND                ; End pick ID 30 (wheel); end definition of
                          Segment 2, the wheel.
!                       ; Model One returns to normal command mode.
```

EXAMPLE, continued

Note that defining a segment does not draw it. To draw a segment, you execute the SEGREF command.

```
! EXMODE NORMAL DRAW     ; Set execution mode to normal draw.
! SEGREF 1               ; Execute (draw) Segment 1.
```



SYNTAX

ASCII	SEGDEL segment
FORTRAN Call	CALL SEGDEL (SEGID)
Binary	[222] [segment3] [segment2] [segment1] [segment0] (5 bytes)
	222 decimal = 336 octal = DE hex

FUNCTION

The SEGDEL command deletes the specified segment.

ASCII PARAMETER

segment	The number of the segment to be deleted (32 bits).

FORTRAN PARAMETER

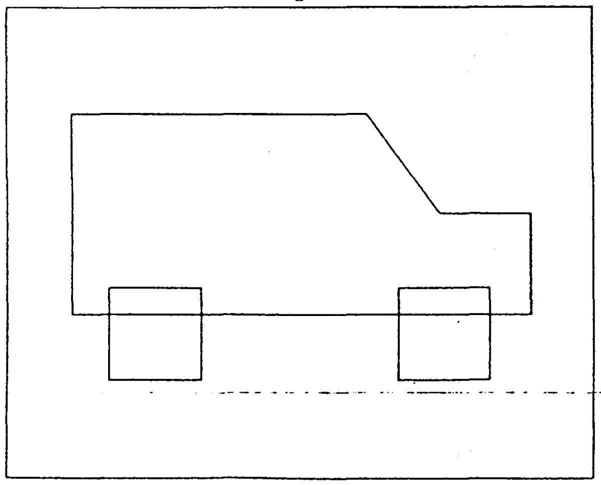
INTEGER*4	SEGID

EXAMPLE

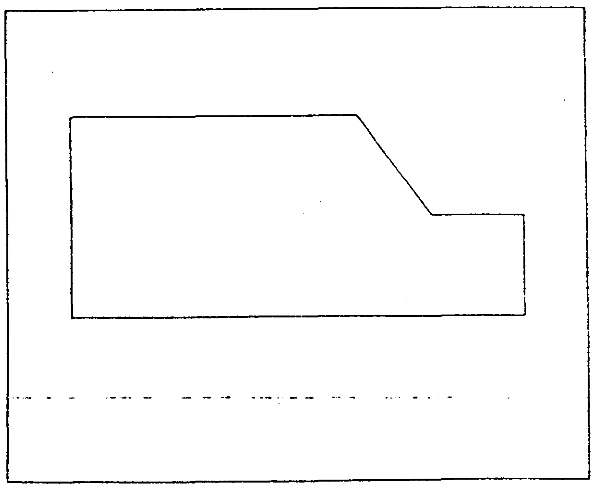
This example refers to the Segment 1 and Segment 2 which were defined in the example for the SEGDEF command. Segment 1 defines the car body, chassis, and position of the two wheels. Segment 2 defines the shape of the wheel. Segment 2 is nested within Segment 1. Note that when Segment 2 is deleted, Segment 1 is still drawn, even though it includes a reference to Segment 2, which no longer exists. However, the Model One issues error message # 083 ('SEGMENT NOT FOUND').

```
! SEGREF 1              ; Execute (draw) Segment 1.
! SEGDEL 2              ; Delete Segment 2, which is nested within Segment 1.
! VAL8 0                ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1              ; Execute (draw) Segment 1, now a truck without
                          wheels.
```

BEFORE DELETION



AFTER DELETION



SEGEND

SYNTAX

ASCII	SEGEND
FORTRAN Call	CALL SEGEND
Binary	[221] (1 byte)
	221 decimal = 335 octal = DD hex

FUNCTION

The SEGEND command ends the definition of an open segment by closing the segment. You must also use the SEGEND command to close a segment that you append with the SEGAPP command.

EXAMPLE

```
! SEGDEF 25            ; Begin definition of Segment 25.
$ RECREL 40 40         ; Draw rectangle.
$ SEGEND               ; End definition of Segment 25.
!                      ; The Model One returns to normal command mode.
```

Note that the Model One prompt changes to "$" during the definition of a segment. SEGEND, which ends the definition, changes the prompt back to "!".

SEGINI

SYNTAX

ASCII	SEGINI reserved, blocksize
FORTRAN Call	CALL SEGINI (BLKSIZ)
Binary	[225] [reserved = 0] [highblocksize][lowblocksize] (4 bytes)
	225 decimal = 341 octal = E1 hex

FUNCTION

The SEGINI command initializes the Display List firmware. You should execute SEGINI before using any of the Display List commands.

The SEGINI command performs the following five functions:

- deletes all existing segments
- initializes the display list structures
- initializes EXMODE to NORMAL DRAW
- initializes the size of the display list blocks to the size specified by the blocksize parameter, and
- initializes the pick aperture to the default size: 8 x 8 pixels for the Model One/10.

ASCII PARAMETERS

reserved	An 8-bit reserved quantity, must be 0.
blocksize	The size (in bytes) of the display list data blocks (16 bits); range is 32 to 16384. Note that an even-numbered range avoids the wasting of memory since odd block sizes are rounded down to the nearest even integer.

FORTRAN PARAMETER

INTEGER*2	BLKSIZ

EXAMPLE

```
! SEGINI 0 128          ; Initialize display list structures with a block
                          size of 128 bytes, delete existing segments,
                          set default pick aperture size, initialize
                          EXMODE to NORMAL.
```

SEGINQ   SEGINQ

SYNTAX

ASCII	SEGINQ segment
FORTRAN Call	CALL SEGINQ (SEGNUM, ISTAT)
Binary	[229] [segment3] [segment2] [segment1] [segment0] (5 bytes)

229 decimal = 345 octal = E5 hex

FUNCTION

The SEGINQ command reads back the attributes of and verifies the existence of the specified segment. One 16-bit word is returned; Bit 0 (the LSB) is set if the segment is visible, and Bit 1 is set if the segment is pickable; all other bits are clear. If the segment does not exist, then the Model One returns negative one.

ASCII PARAMETER

segment	The number of the segment whose attributes are being queried.

FORTRAN PARAMETER

Input Parameter:	INTEGER*4	SEGNUM
Output Parameter:	INTEGER*2	ISTAT

EXAMPLE

```
! SEGINQ 1              ; Query the attributes of Segment 1.
  00003                 ; The Model One returns:
                          Segment 1 exists, visibility is ON, pickability
                          is ON.
! SETATR 1 PICK OFF     ; Set pickability OFF for Segment 1.
! SEGINQ 1              ; Query the attributes of Segment 1.
  00001                 ; The Model One returns:
                          Segment 1 exists, visibility is ON, pickability
                          is OFF.
! SETATR 1 VIS OFF      ; Set visibility OFF for Segment 1.
! SEGINQ 1              ; Query the attributes of Segment 1.
  00000                 ; The Model One returns:
                          Segment 1 exists, visibility is OFF, pickability
                          is OFF.
! SEGDEL 1              ; Delete Segment 1.
! SEGINQ 1              ; Query the attributes of Segment 1.
  -00001                ; The Model One returns:
                          Segment 1 does not exist.
```

SEGREF

SYNTAX

ASCII	SEGREF segment
FORTRAN Call	CALL SEGREF (SEGID)
Binary	[216] [segment3][segment2][segment1][segment0] (5 bytes)
	216 decimal = 330 octal = D8 hex

FUNCTION

The SEGREF command has two functions: to execute a top-level segment or to reference a segment within the current segment definition.

SEGREF executes a top-level segment according to the current execution mode, as set by the EXMODE command.

SEGREF can also be used to nest segments by referencing the number of a child segment from within the definition of a parent. Segments can be nested to a level of 16. Lower-level segments must be defined if they are to be executed when their parent is executed. However, they need not be defined when referenced.

Note: The SEGREF command only updates the SEGREG and PIDREG registers if you are in PICK mode and there is a pick hit.

ASCII PARAMETER

segment	The number of the segment (32 bits).

FORTRAN PARAMETER

INTEGER*4	SEGID

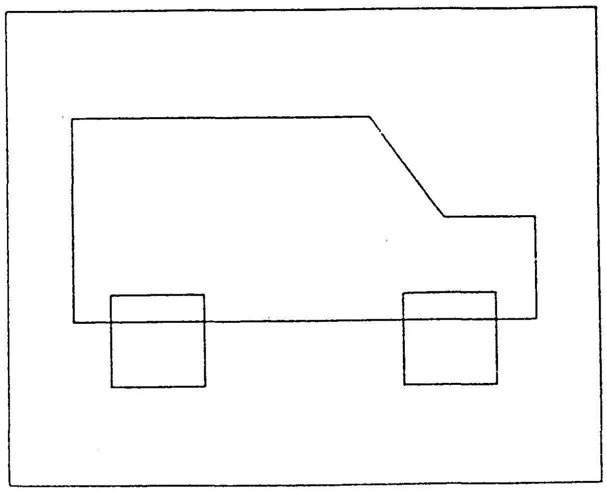
EXAMPLE 1: Nesting a Segment

This example illustrates the use of the SEGREF command to nest one segment within another. In this example, Segment 2 (which defines the truck wheels) is nested two times within Segment 1 (which defines the truck body, chassis and position of the wheels). This is the same example that was used to illustrate the SEGDEF command.

```
! SEGINI 0 128            ; Initialize display list structures with a
                            block size of 128 bytes.
! SEGDEF 1                ; Begin definition of Segment 1, which defines
                            the truck and the position of the 2 wheels.
$ PICKID 10               ; Assign pick ID 10 label to the truck body.
$ PUSH CREG CURPNT        ; Push WCS current point onto the stack.
$ MOVABS -250,-80         ; Move current point to lower left corner of truck.
$ DRWREL 0,220            ; Draw 5 vectors to define truck body.
$ DRWREL 320,0
$ DRWREL 80,-110
$ DRWREL 100,0
$ DRWREL 0,-110
$ PICKID 20               ; End pick ID 10 (body); assign pick 20 label to
                            the chassis (bottom line) of truck.
$ DRWREL -500,0           ; Draw chassis of truck.
$ MOVABS -160,-100        ; Move current point to position rear wheel.
$ SEGREF 2                ; Nest Segment 2, the wheel.
$ MOVABS 155,-100         ; Move current point to position front wheel.
$ SEGREF 2                ; Nest Segment 2, the wheel.
$ POP CREG CURPNT         ; Pop WCS current point from the stack.
$ SEGEND                  ; End pick ID 20 (chassis); end definition of
                            Segment 1, the truck body and position of wheels.

! SEGDEF 2                ; Begin definition of Segment 2, which defines the
                            shape of the wheel.
$ PICKID 30               ; Assign pick ID 30 label to the wheel.
$ PUSH CREG CURPNT        ; Save current point (center of wheel).
$ MOVREL -50,-50          ; Move current point to left corner of wheel.
$ RECREL 100,100          ; Draw wheel.
$ POP CREG CURPNT         ; Restore current point (center of wheel)
$ SEGEND                  ; End pick ID 30 (wheel); end definition of
                            Segment 2, the wheel.
!                         ; Model One returns to normal command mode.
```

EXAMPLE 2: Executing a Segment - Drawing

```
! EXMODE NORMAL DRAW     ; Set execution mode to normal draw (the default).
! SEGREF 1               ; Execute (draw) Segment 1.
```



Please refer to the EXMODE command for examples of executing segments in pick mode, highlight mode, and unhighlight mode.

SEGREN

SYNTAX

ASCII	SEGREN segment2, segment1
FORTRAN Call	CALL SEGREN (SEGNEW, SEGOLD)
Binary	[218] [segment23] [segment22] [segment21] [segment20] [segment13] [segment12] [segment11] [segment10] (9 bytes)
	218 decimal = 332 octal = DA hex

FUNCTION

The SEGREN command renames a specified segment. Segment1 is the segment you want to rename; segment2 is the new name. If segment2 already exists, it will be overwritten.

Caution: Any references to an old segment name in SEGDEF and SEGREF commands are not automatically updated when that segment is renamed.

ASCII PARAMETERS

segment1	The number of the segment to be renamed (32 bits).
segment2	The new number for the segment (32 bits).

FORTRAN PARAMETERS

INTEGER*4	SEGNEW, SEGOLD

EXAMPLE

```
! SEGREN 21 20          ;Rename Segment 20 to Segment 21.
```

SYNTAX

ASCII	SETATR segment, attrib, flag
FORTRAN Call	CALL SETATR (SEGID, ATTRIB, FLAG)
Binary	[230] [segment3] [segment2] [segment1] [segment0] [attrib] [flag] (7 bytes)
	230 decimal = 346 octal = E6 hex

FUNCTION

The SETATR command enables or disables the visibility or pickability attributes of the specified segment. Both of these attributes are automatically set to ON upon segment definition.

Note: When you change a segment's attributes, the new attributes are not visibly noticed until that segment is re-referenced.

ASCII PARAMETERS

segment	The number of the segment for which you want to change an attribute (32 bits).
attrib	The attribute to be set (8 bits).
flag	Flag = 1 or ON, enable attribute; flag = 0 or OFF, disable attribute.

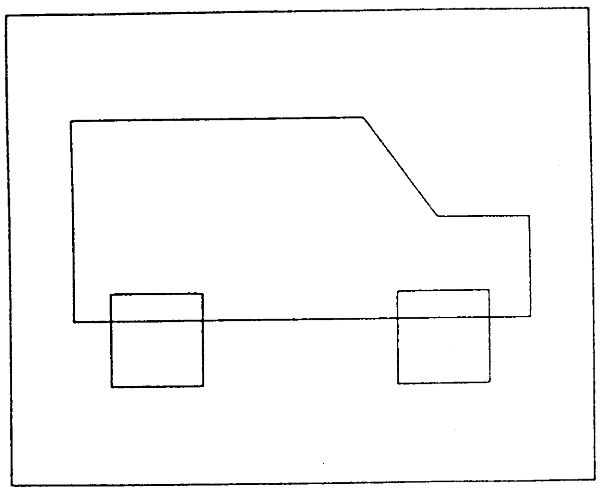
FORTRAN PARAMETERS

```
INTEGER*4    SEGID
INTEGER*2    ATTRIB, FLAG
```

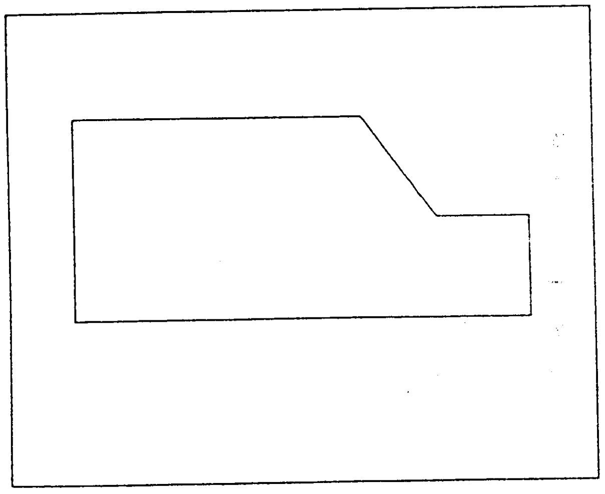
EXAMPLE: Setting the Visibility Attribute to OFF

```
! SEGREF 1              ; Execute (draw) Segment 1.
! SETATR 2 VIS OFF      ; Set visibility OFF for Segment 2, the wheel.
! VAL8 0                ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1              ; Execute (draw) Segment 1; nested Segment 2 is
                          not visible.
```

SEG 1: EVERYTHING VISIBLE



SEG 1: NESTED SEG 2 NOT VISIBLE



ASCII SYNTAX

```
SETGL  global, value
```

FORTRAN SUBROUTINE CALLS

```
CALL SETGL  (GLOBAL,XAPSZ,YAPSZ,PID,SID,IGNORE,PICBUF)
```

Note: The following are provided for convenience.

```
CALL SETGL0 (XAPSZ,YAPSZ)      (Set value for pick aperture)
CALL SETGL1 (PID)              (Set value for pick ID register)
CALL SETGL2 (SID)              (Set value for segment ID register)
CALL SETGL3 (IGNORE)           (Set number of pick hits to ignore)
CALL SETGL4 (PICBUF)           (Set information to store in pick buffer)
```

HOST BINARY COMMAND STREAMS

```
[70] [global = 0] [highxap][lowxap] [highyap][lowyap]   (6 bytes)
[70] [global = 1] [pid3][pid2][pid1][pid0]              (6 bytes)
[70] [global = 2] [sid3][sid2][sid1][sid0]              (6 bytes)
[70] [global = 3] [highignore][lowignore]               (4 bytes)
[70] [global = 4] [picbuf]                              (3 bytes)
```

70 decimal = 106 octal = 46 hex

FUNCTION

The SETGL command sets the values of the following globally defined display list parameters:

- the size of the pick aperture
- the current pick ID register
- the current segment ID register
- the number of picks to ignore during picking, and
- the type of information to store in the pick buffer during picking.

The global parameter specifies which global parameter to set.

The value parameter varies depending on which parameter is being set.

ASCII PARAMETERS

global — The global parameter to set, as follows.

0 = PICKAP	Size of pick aperture.
1 = PIDREG	Pick ID number.
2 = SEGREG	Segment ID number.
3 = IGNORE	Pick hits to ignore during picking.
4 = PICKBUF	Information to store in the pick buffer during picking.

value — The new value for the specified parameter.

global = 0	Two 16-bit parameters which represent half of the size of the pick aperture in the x and y dimensions. On the Model One/10, the default pick aperture size is 8 x 8 (SETGL PICKAP 4 4).
global = 1	The pick ID number (32 bits).
global = 2	The segment ID number (32 bits).
global = 3	The number of picks to be ignored during picking (32 bits).
global = 4	The information type (8 bits), as follows.

0 = TREE	Store entire pick trees.
1 = SEGID	Store only segment IDs.
2 = PICKID	Store only pick IDs.
3 = SEGPID	Store segment ID/pick ID pairs. (Default)

FORTRAN PARAMETERS

```
INTEGER*2     GLOBAL, XAPSZ, YAPSZ, IGNORE, PICBUF

INTEGER*4     PID, SID
```

EXAMPLE

```
! SETGL PICKAP 8 8         ; Set the pick aperture size to 16 x 16.
! SETGL PICKBUF TREE       ; Specify to save entire trees in pick buffer
                             during picking.
```

SYNTAX

ASCII	SETPAT pattern
FORTRAN Call	CALL SETPAT (IPAT)
Binary	[121] [patternhi] [patternlo]

121 decimal = 171 octal = 79 hex

FUNCTION

The SETPAT command sets the area fill pattern to be used when area fills ar performed.

The pattern is a 16 bit pattern organized as follows.

```
0    1    2    3
4    5    6    7
8    9    10   11
12   13   14   15
```

15 is the most significant bit and 0 is the least significant bit.

If a bit is 1 (set), it is drawn in the foreground color (VREG 0). If a bi is 0 (cleared), it is drawn in the background color (VREG 6).

The default is solid fill (#FFFF).

ASCII Parameter

pattern The 16-bit fill pattern. (See FUNCTION above for details.)

FORTRAN Parameter

INTEGER*2 IPAT

SYNTAX

ASCII	SPCHAR char, flag, code
FORTRAN Call	CALL SPCHAR (ICHAR, IFLAG, ICODE)
Binary	[178] [char] [flag] [code] (4 bytes)
	178 decimal = 262 octal = B2 hex

FUNCTION

The SPCHAR command can be used to redefine or disable any of the special characters used by the Model One, thereby circumventing problems with certain host computers and operating systems. The parameter char specifies which of the special characters is to be defined or disabled.

The default values for special characters are listed below.

Char	ASCII Code	Hex equivalent	Purpose
0	CTRL D	04 or 84	Enter Graphics mode
1	CTRL Y	19 or 99 (CTRL-Y)	Send break to host
2	CTRL V	16 or 96 (CTRL-V)	WARMstart
3	@	40 or B0	Line kill
4	CTRL H	08 or 88	Backspace
5	CTRL F	06 or 86	ACK (Acknowledge)
6	CTRL U	15 or 95	NACK (Negative Acknowledge) (abort)
7	CTRL X	18 or 98	Invoke Debug mode
8	CTRL Q	11 or 91	Resume communications (XON)
9	CTRL S	13 or 93	Suspend communications (XOFF)

Flag = 1 or ON indicates that the special character is to be redefined. The third parameter, code, specifies the new ASCII code of the special character. Flag = 0 or OFF indicates that the special character is to be disabled. The code parameter is ignored but must be present.

FUNCTION, continued

Once you have made changes to the special characters, you may save the changes with the SAVCFG command. Then the new special characters will be initialized with every COLDstart command.

Note: Disabling the WARMstart character does not disable the WARMstart command.

ASCII PARAMETERS

char	Special character number; range is 0 to 9.
flag	Flag = 1 or ON, redefine special character; flag = 0 or OFF, disable special character.
code	8-bit parameter specifying the new ASCII code.

FORTRAN PARAMETERS

```
INTEGER*2     ICHAR, IFLAG, ICODE
```

EXAMPLE

```
! SPCHAR 0 1 #05         ; Change the Enter Graphics mode control code to
                           05 hex or 85 hex (CTRL-E).
! SPCHAR 2 0 0           ; Disable the WARMstart character.
```

SYNTAX

```
SYSCFG ERROR  port_mnemonic

SYSCFG HOST   port_mnemonic, ASCII or BINARY

SYSCFG SERIAL port_mnemonic [RTS on/off] [CTS on/off] [STOP 1/2]
              [BITS 7/8] [PARITY e/o/l/h/n] [BAUD rate] [CTRL on/off]
              [XIN on/off] [XOUT on/off]

SYSCFG SERIAL TABLET type
```

FUNCTION

The SYSCFG command has several different forms, which are each described in more detail on the following pages. The table below outlines the forms of the SYSCFG command.

Command	Function
SYSCFG ERROR	Configures the Model One's error port.
SYSCFG HOST	Configures the host port to accept either 8-bit binary characters or ASCII hexadecimal characters from the host.
SYSCFG SERIAL	Configures the Model One's host port (e.g., baud, parity, XIN, etc.).
SYSCFG SERIAL TABLET	Sets the tablet port for the graphics input device that is being used.

Notes: Each of the SYSCFG commands can be executed only from the local alphanumeric terminal, and cannot be included in a macro.

See the following pages for a more detailed description of each of the SYSCFG commands.

You may prefer to use the Setup mode, entered by pressing the SETUP key on the keyboard. Setup mode is fully documented in Section 13 of the Model One/10 Introduction and Installation Guide, Rev. 2.0.

ASCII SYNTAX

```
SYSCFG ERROR  port_mnemonic
```

FUNCTION

The SYSCFG ERROR command specifies which port is to be used as the error port. This command can only be executed from the local alphanumeric terminal, and cannot be included in a macro.

ASCII PARAMETER

port_mnemonic The port mnemonic or port number. The ports are:

1	TABLET	Tablet or mouse port
0	HOST	Host serial port

ASCII SYNTAX

```
    SYSCFG HOST  port_mnemonic,  ASCII or BINARY
```

FUNCTION

The SYSCFG HOST command configures the host port to accept either ASCII hexadecimal or 8-bit binary characters from the host. The command does not change the default host port.

Note: When you are installing the serial interface to the host, you should configure the HOST port to match the characteristics of the terminal that was connected to the line previously.

ASCII PARAMETER

port_mnemonic The port mnemonic or port number. The ports are:

1	TABLET	Tablet or mouse port
0	HOST	Host serial port

EXAMPLE

```
! SYSCFG HOST HOST ASCII        ; Configure the host port to accept ASCII
                                  hexadecimal characters.
```

SYSCFG SERIAL SYSCFG SERIAL

ASCII SYNTAX

```
    SYSCFG SERIAL  port_mnemonic [RTS on/off] [CTS on/off] [STOP 1/2]
                   [BITS 7/8] [PARITY e/o/l/h/n] [BAUD rate] [CTRL on/off]
                   [XIN on/off] [XOUT on/off]
```

FUNCTION

The SYSCFG SERIAL command configures the Model One's host port. This command can only be executed from the local alphanumeric terminal, and cannot be included in a macro.

The tablet port configuration is established for each type of XY digtizer. The tablet port is configured with the SYSCFG SERIAL TABLET <type> command.

Displaying the Current Configuration: To display the current configuration, use the DISCFG command. The following is an example of what you might see after entering the DISCFG command. In this example, the default configuration is displayed.

```
PORT          RTS   CTS   STOP BITS   XIN XOUT   CTRL   PARITY    BAUD
_______________________________________________________________________

TABLET        OFF   OFF   2     8     OFF  OFF   OFF    NONE      9600
HOST          OFF   OFF   1     8     OFF  ON    OFF    NONE      9600
```

Changing the Configuration: To change the configuration of any serial port, use the SYSCFG SERIAL command, followed by the port to be configured, and the keywords and values for any parameter you want to change. Omit any parameter you want to leave unchanged.

Saving the New Configuration: After you have changed the configuration, you need to save the new configuration with the SAVCFG command. The SAVCFG command copies the new configuration into NVRAM (non-volatile RAM).

Restoring the Default Configuration: To reset the configuration of serial ports to the default values, use the DFTCFG command. The default values are stored in PROM (programmable read-only memory) are are not affected by the SYSCFG or SAVCFG commands.

ASCII PARAMETERS

port_mnemonic	Supplies the mnemonic of the serial port to be configured. For the Model One/10, the only port configured with the SYSCFG SERIAL command is the host. 0 or HOST
RTS	Not currently applicable to the Model One/10.
CTS	Not currently applicable to the Model One/10.
STOP	Specifies whether 1 or 2 stop bits should be used.
BITS	Specifies whether 7 or 8 bits are sent per byte.
XIN	Indicates whether XON/XOFF is to be accepted at input. ON specifies that output will be enabled or disabled according to the XON/XOFF signals received by the port. OFF specifies that XON/XOFF signals should be ignored.
XOUT	Indicates whether XON/XOFF should be sent when the port's queue is near full (ON) or simply not used (OFF).
CTRL	Indicates whether the Model One/10 should accept control characters (including [CTRL-S] and [CTRL-Q]) from the port (ON) or ignore them (OFF).
PARITY	Specifies input/output parity. Parity may be even (E), odd (O), high (H), low (L), or not used (N).
BAUD	Specifies baud rate, which may be 75, 100, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, or 19200.

EXAMPLE

```
! SYSCFG SERIAL HOST BAUD 4800          ; Changes the baud rate for the
                                          host port to 4800;  leaves
                                          other parameters unchanged.

! SYSCFG SERIAL HOST PARITY 0 CTRL ON   ; Changes, for the host port,
                                          parity to odd and indicates
                                          control characters should be
                                          accepted.
```

ASCII SYNTAX

```
    SYSCFG SERIAL TABLET  type
```

FUNCTION

The SYSCFG SERIAL TABLET command sets the tablet port for the XY digitizer that is being used.

You can only enter this command from the local terminal. You cannot include the command in a macro.

ASCII PARAMETER

type	GTCO	GTCO tablet.
	SUMMA	Summagraphics Bit-Pad or mouse.

EXAMPLE

```
! SYSCFG SERIAL TABLET GTCO         ; Sets the tablet port for the GTCO
                                      tablet.
```

ASCII SYNTAX

```
SYSTAT  infotype   or
SYSTAT  infotype, segment
```

FORTRAN Calls

```
CALL SYST80 (FUNC,ARRSIZ,SID,FREBLK,NUM,SEGS,NUMBLK)    (general form)

CALL SYSTA0 (FREBLK)                 (Return number of free memory blocks
                                      available for segment definition.)
CALL SYSTA1 (ARRSIZ, NUM, SEGS)      (Return array of defined segments.)
CALL SYSTA2 (SID, NUMBLK)            (Return number of memory blocks used
                                      by specified segment.)
```

Binary

```
[228] [infotype = 0,1]          (2 bytes)
[228] [infotype = 2] [segment3][segment2][segment1][segment0]  (6 bytes)

228 decimal = 344 octal = E4 hex
```

FUNCTION

The SYSTAT command returns information on display list memory usage and availability. The value of the infotype parameter (0, 1, or 2) determines the type of information that the SYSTAT command returns.

If infotype = 0, the SYSTAT command returns a 32-bit word indicating the number of remaining free memory blocks available. These blocks are the size specified with the SEGINI command.

If infotype = 1, the SYSTAT command returns an array of defined segment numbers.

If infotype = 2, the SYSTAT command returns the number of memory blocks used by the specified segment.

ASCII PARAMETERS

infotype	The type of information to return.	
	0 = FREEMEM	Return the number of free memory blocks available for segment definition.
	1 = SEGS	Return array of defined segment numbers. Format is (n,array), where n is the number of defined segments (16 bits), and array is an array of n segment IDs (32 bits). The returned array is unsorted.
	2 = SIZE	Return the number of memory blocks used by the specified segment (32 bits).
segment	The number of the segment for which to return memory block information (32 bits). Use this parameter only when infotype = 2.	

FORTRAN PARAMETERS

CALL SYST80 (FUNC, ARRSIZ, SID, FREBLK, NUM, SEGS, NUMBLK)

Input Parameters:	INTEGER*2	FUNC, ARRSIZ
	INTEGER*4	SID
Output Parameters:	INTEGER*2	NUM
	INTEGER*4	FREBLK, SEG(1), NUMBLK

FUNC	The type of information to return.
ARRSIZ	Size of the segments array in longwords.
SID	The segment ID used for FUNC = 2.
FREBLK	The number of free memory blocks available for segment definition.
NUM	The number of defined segments.
SEGS	Array of NUM segments IDs.
NUMBLK	The number of blocks used in SID.

EXAMPLE

In this example, the blocksize is 256 and Segment 1 has been defined as specified in the SEGDEF command example.

```
! SYSTAT FREEMEM        ; Query system memory availability.
 0000001150             ; Model One returns: 1150 free memory blocks
                          available for segment definition.
! SYSTAT SEGS           ; Ask for the number of defined segments, and a
                          list of segment IDs.
  00002                 ; Model One returns: two segments have been
 0000000001               defined, Segment 1 and Segment 2.
 0000000002
! SYSTAT SIZE 1         ; Ask for the number of memory blocks used by
                          Segment 1.
 0000000001             ; Model One returns: 1 memory block used by
                          Segment 1.
```

SYNTAX

ASCII	TEKEM flag
FORTRAN Call	CALL TEKEM (FLAG)
Binary	[57] [flag] (2 bytes)
	57 decimal = 071 octal = 39 hex

FUNCTION

The TEKEM command invokes the Tektronix 4014 emulator. For complete details of the command, see the manual, Tektronix Emulator Technical Note.

ASCII PARAMETERS

flag	See the Tektronix Emulator Technical Note for details.

FORTRAN PARAMETERS

```
INTEGER*2     FLAG
```

Note: The FORTRAN call, CALL TEKEM (FLAG), enters the Tektronix Emulator; it is then up to the user to ensure that the correct data is sent. After the emulator is exited, Model One FORTRAN library commands may again be used.

SYNTAX

ASCII	TEXT1 string
FORTRAN Call	CALL TEXT1 (STRLEN, STRING)
Binary	[144] [strlen] ([char1][char2]...[charn])
	144 decimal = 220 octal = 90 hex

FUNCTION

The TEXT1 command draws a horizontal text string into image memory with Font 1. If the text size is the default 16, each character uses 7 x 9 pixels. Descenders take an additional two pixels.

The parameter string specifies the text to be drawn. If the command is entered in ASCII mode from the local alphanumeric terminal or keyboard, then the string to be drawn is the set of ASCII characters which follow the TEXT1 command on the command line. If the TEXT1 command is sent from the host, then the length of the string must be specified. Strlen specifies the number of characters in the string, and is followed by strlen bytes containing the ASCII characters to be drawn.

The current point (CREG 0) specifies the starting point for the text string and remains unchanged by the command.

You can specify the size of the text and the baseline angle of the text using the TEXTC or TEXTN commands.

Transformations affect the rotation, scaling, and translation of text that is drawn with any of the text drawing commands.

ASCII PARAMETER

string	The text to be drawn.

FORTRAN PARAMETERS

```
INTEGER*2     STRLEN, STRING(1)
```

STRLEN is an integer specifying the number of characters that are to be drawn. STRING is an integer array with two characters packed per 16-bit word, as in FORTRAN A2 format.

EXAMPLE

```
! MOVABS 0 0             ; Move current point to 0,0.
! TEXT1 Horizontal text ; Draw text string in default size 16 and angle 0.
! MOVABS 0 20            ; Move current point to 0,20.
! TEXTC 32 45            ; Set double-sized text at a 45 degree angle.
! TEXT1 Double-sized text at an angle
```

TEXT2 TEXT2

SYNTAX

ASCII	TEXT2 string
FORTRAN Call	CALL TEXT2 (STRLEN, STRING)
Binary	[145] [strlen] ([char1][char2]...[charn])
	145 decimal = 221 octal = 91 hex

FUNCTION

The TEXT2 command draws a horizontal text string into image memory with Font 2. Font 2 is a user defined character set which is downloaded using the TEXTDN command.

At power-on or COLDstart, each character in Font 2 defaults to the same character in Font 1 (Font 2 will appear twice as large as Font 1). When Font 2 is downloaded, each character replaces the power-on default definition.

The TEXT2 command is issued in the same manner as the TEXT1 command. The current point (CREG 0) specifies the starting point for the text string and remains unchanged by the command. Strlen specifies the length of the string when text is not sent in ASCII mode.

ASCII PARAMETER

string	The text to be drawn.

FORTRAN PARAMETERS

INTEGER*2 STRLEN, STRING(1)

STRLEN is an integer specifying the number of characters that are to be drawn. STRING is an integer array with two characters packed per 16-bit word, as in FORTRAN A2 format.

TEXTC

SYNTAX

ASCII	TEXTC size, ang
FORTRAN Call	CALL TEXTC (ISIZE, IANG)
Binary	[146] [size] [highang][lowang] (4 bytes)
	146 decimal = 222 octal = 92 hex

FUNCTION

The TEXTC command sets the size of text and the baseline angle for subsequent text drawing commands: TEXT1, TEXT2, VTEXT1, and VTEXT2.

The size parameter specifies the scale factor, which may range from 0 to 255. Size = 16 is the default. A size of 32 doubles the size of the text. The parameter ang specifies the angle in which the text will be drawn. The angle is measured in one-degree increments counter-clockwise. For the TEXT1 and TEXT2 commands (horizontal text), zero degrees is the positive x axis. An angle of zero degrees causes the TEXT1 and TEXT2 commands to draw normally oriented text from left to right. For the VTEXT1 and VTEXT2 commands (vertical text), an angle of zero degrees causes the commands to draw normally oriented text from top to bottom.

Note: The TEXTC and TEXTN commands work independently of one another. Whichever command was executed last takes precedence.

ASCII PARAMETERS

size	Text size; range is 0 to 255. (Default 16)
ang	Text angle; range is -32,768 to 32,767. (Default 0)

FORTRAN PARAMETERS

INTEGER*2	ISIZE, IANG

EXAMPLE

```
! TEXT1 Horizontal text       ; Draw text string in default size 16 and
                                default angle 0.
! TEXTC 16 30                 ; Set text size to 16, angle to 30 degrees.
! TEXT1 Angled text           ; Draw text at a 30 degree angle.
! MOVABS 0 50                 ; Move current point to 0,50.
! TEXTC 32 0                  ; Set text size to 32, angle to 0 degrees.
! TEXT1 Larger text           ; Draw double sized text.
```

TEXTDN

SYNTAX

ASCII	TEXTDN char, numvec, veclst
FORTRAN Call	CALL TEXTDN (ICHAR, NUMPTS, IPOINT)
Binary	[38] [char] [highnumvec][lownumvec] [highv1][lowv1] ... [highvn][lowvn] (4 + 2 * numvec bytes)
	38 decimal = 046 octal = 26 hex

FUNCTION

The TEXTDN command defines a second font, Font 2, for text drawing. Font 2 is used by the TEXT2 and VTEXT2 commands. The parameter char specifies the character to be defined. Veclst defines the series of relative moves and draws that specify the character. Numvec specifies the number of vectors that will be drawn.

The TEXTDN command uses a 64 x 64 matrix for the definition of each character. Each move and draw of one unit in the matrix is equivalent to a pixel. With normal text size and angle (size = 16, ang = 0), the vectors will be drawn exactly as specified. To achieve varying spacing between user-defined characters, you should make the last part of the character definition a move to the beginning of the next character.

When transmitting data from the host, each relative move or draw requires 2 bytes of information. Each vector2 bytes of information. Each vector is specified as follows:

Bits 0-6:	7-bit two's complement relative y change; range -64 to 63.
Bit 7:	May be 0 or 1.
Bits 8-14:	7-bit two's complement relative x change; range -64 to 63.
Bit 15:	1 for DRAW, 0 for MOVE.

You may define 128 different characters. Font 2 is initialized to the same character set as Font 1. The font for a particular character only changes after the new character is defined with the TEXTDN command. This allows you to change just a portion of the character set.

The Font 1 characters are listed at the end of this command description for reference.

TEXTDN

SYNTAX

ASCII	TEXTDN char, numvec, veclst
FORTRAN Call	CALL TEXTDN (ICHAR, NUMPTS, IPOINT)
Binary	[38] [char] [highnumvec] [lownumvec] [highv1] [lowv1] ... [highvn] [lowvn] (4 + 2 * numvec bytes)
	38 decimal = 046 octal = 26 hex

FUNCTION

The TEXTDN command defines a second font, Font 2, for text drawing. Font 2 is used by the TEXT2 and VTEXT2 commands. The parameter char specifies the character to be defined. Veclst defines the series of relative moves and draws that specify the character. Numvec specifies the number of vectors that will be drawn.

The TEXTDN command uses a 64 x 64 matrix for the definition of each character. Each move and draw of one unit in the matrix is equivalent to a pixel. With normal text size and angle (size = 16, ang = 0), the vectors will be drawn exactly as specified. To achieve varying spacing between user-defined characters, you should make the last part of the character definition a move to the beginning of the next character.

Each relative move or draw requires 2 bytes of information. Each vector is specified as follows:

Bits 0-6:	7-bit two's complement relative y change; range -64 to 63.
Bit 7:	May be 0 or 1.
Bits 8-14:	7-bit two's complement relative x change; range -64 to 63.
Bit 15:	1 for DRAW, 0 for MOVE.

You may define 128 different characters. Font 2 is initialized to the same character set as Font 1. The font for a particular character only changes after the new character is defined with the TEXTDN command. This allows you to change just a portion of the character set.

The Font 1 characters are listed at the end of this command description for reference.

ASCII PARAMETERS

char	8-bit integer specifying the number of the character to be defined.
numvec	16-bit integer specifying the number of vectors to be drawn.
veclst	16-bit integer specifying the vectors that define the character.

FORTRAN PARAMETERS

INTEGER*2 ICHAR, NUMPTS, IPOINT(1)

IPOINT is an integer array containing the list of moves and draws required to draw the character; the format described above is used.

FORTRAN EXAMPLE

The following FORTRAN program defines a special character, which is a box with an "x" in it. The call to the TEXTDN command replaces ASCII A with this new character. The subsequent call to the TEXT2 command draws the new character on the screen. If the host computer does not accept numbers in hex, you should convert them to octal.

```
program textdn_example
integer*2   text(8)           /* Dimension array for character definition.
call RTINIT ('-serial',0)     /*Initialize library for serial output.
call ENTGRA                   /* Enter graphics mode.

text(1) = #8A00          /* Draw in positive x 10 pixels.
text(2) = #800A          /* Draw in positive y 10 pixels.
text(3) = #F600          /* Draw in negative x 10 pixels.
text(4) = #8076          /* Draw in negative y 10 pixels.
text(5) = #8A0A          /* Draw in positive x 10 pixels and y 10 pixels.
text(6) = #7600          /* Move in negative x 10 pixels.
text(7) = #8A76          /* Draw in positive x 10 pixels and negative y 10
                         /* pixels.
text(8) = #1400          /* Move in positive x 20 pixels, putting the starting
                         /* point of the next character 20 pixels to the
                         /* right of the character being defined.

call TEXTDN (65,8,text) /* Replace ASCII A with new definition.
call TEXT2 (1,A)         /* Draw [x] on the screen.
call QUIT                /* Exit Graphics mode.
call EXIT                /* Exit from FORTRAN.
end                      /* End from FORTRAN.
```

Decimal	Hex	Character	Decimal	Hex	Character	Decimal	Hex	Character
0	00	NUL	46	2E	. (period)	92	5C	\
1	01	SOH	47	2F	/	93	5D	]
2	02	STX	48	30	0	94	5E	^
3	03	ETX	49	31	1	95	5F	_ (line)
4	04	EOT	50	32	2	96	60	` (quote)
5	05	ENQ	51	33	3	97	61	a
6	06	ACK	52	34	4	98	62	b
7	07	BEL	53	35	5	99	63	c
8	08	BS	54	36	6	100	64	d
9	09	HT	55	37	7	101	65	e
10	0A	LF	56	38	8	102	66	f
11	0B	VT	57	39	9	103	67	g
12	0C	FF	58	3A	:	104	68	h
13	0D	CR	59	3B	;	105	69	i
14	0E	SO	60	3C	<	106	6A	j
15	0F	SI	61	3D	=	107	6B	k
16	10	DLE	62	3E	>	108	6C	l
17	11	DC1	63	3F	?	109	6D	m
18	12	DC2	64	40	@	110	6E	n
19	13	DC3	65	41	A	111	6F	o
20	14	DC4	66	42	B	112	70	p
21	15	NAK	67	43	C	113	71	q
22	16	SYN	68	44	D	114	72	r
23	17	ETB	69	45	E	115	73	s
24	18	CAN	70	46	F	116	74	t
25	19	EM	71	47	G	117	75	u
26	1A	SUB	72	48	H	118	76	v
27	1B	ESC	73	49	I	119	77	w
28	1C	FS	74	4A	J	120	78	x
29	1D	GS	75	4B	K	121	79	y
30	1E	RS	76	4C	L	122	7A	z
31	1F	US	77	4D	M	123	7B	{
32	20	SP	78	4E	N	124	7C	|
33	21	!	79	4F	O	125	7D	}
34	22	"	80	50	P	126	7E	~
35	23	#	81	51	Q	127	7F	DEL
36	24	$	82	52	R			
37	25	%	83	53	S			
38	26		84	54	T			
39	27	'	85	55	U			
40	28	(	86	56	V			
41	29	)	87	57	W			
42	2A	*	88	58	X			
43	2B	+	89	59	Y			
44	2C	, (comma)	90	5A	Z			
45	2D	-	91	5B	[			

TEXTN

SYNTAX

ASCII	TEXTN xsize, ysize, xangle, yangle
FORTRAN Call	CALL TEXTN (XSIZE, YSIZE, XANGLE, YANGLE)
Binary	[169] [xsize] [ysize] [hixangle] [loxangle] [hiyangle] [loyangle] (7 bytes)
	169 decimal = 251 octal = A9 hex

FUNCTION

The TEXTN command, like the TEXTC command, sets the size of text and the baseline angle for subsequent text drawing commands. The TEXTN command allows independent control of the x and y components of the text size and angle. Note that if xsize = ysize and xangle = yangle, the TEXTN command is identical to the TEXTC command.

The following are some examples of the kinds of text you can create using the TEXTN command.

Tall skinny letters	xsize < ysize
Short fat letters	xsize > ysize
Forward italics	xangle = 0, yangle < 0
Backward italics	xangle = 0, yangle > 0
Forward italics at an angle	xangle > 0, yangle = 0
Backward italics at an angle	xangle < 0, yangle = 0

Note: The TEXTC and TEXTN commands work independently of one another. Whichever command was executed last takes precedence.

ASCII PARAMETERS

xsize	The x component size; range is 0 to 255. (Default 16)
ysize	The y component size; range is 0 to 255. (Default 16)
xangle	The x angle; range is -32,768 to 32,767. (Default 0)
yangle	The y angle; range is -32,768 to 32,767. (Default 0)

FORTRAN PARAMETERS

INTEGER*2	XSIZE, YSIZE, XANGLE, YANGLE

EXAMPLE

```
! MOVABS 0 0             ; Move current point to 0,0.
! TEXTN   32 16 0 0      ; Set up for short, fat text.
! TEXT1 Short fat text
! MOVABS 0 -25           ; Move current point to 0,-25.
! TEXTN 32 32 0 -45      ; Set up for double sized text, forward italics.
! TEXT1 Forward italics
! MOVABS 0 -50           ; Move current point to 0,-50.
! TEXTN 16 16 0 45       ; Set up for normal sized text, backward italics.
! TEXT1 Backward italics
```

TEXTRE TEXTRE

SYNTAX

ASCII	TEXTRE
FORTRAN Call	CALL TEXTRE
Binary	[177] (1 byte)
	177 decimal = 261 octal = B1 hex

FUNCTION

The TEXTRE command erases the definition of any user defined characters sent via the TEXTDN command and resets Font 2 equal to Font 1 (Font 2 will appear twice as large as Font 1). The TEXTRE command frees all of the space used by previously defined characters.

VADD

SYNTAX

ASCII	VADD vsum, vreg
FORTRAN Call	CALL VADD (IVSUM, IVREG)
Binary	[166] [vsum] [vreg] (3 bytes)
	166 decimal = 246 octal = A6 hex

FUNCTION

The VADD command adds the contents of value register vreg to the contents of value register vsum and places the result into value register vsum.

vsum = vsum + vreg

Only the first byte of the value register is used as an index into the look-up table. As a result, any information in the second and third bytes is ignored.

ASCII PARAMETERS

vsum, vreg	Value registers; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

INTEGER*2	IVSUM, IVREG

EXAMPLE

```
! VLOAD 10 37 0 0       ; Load VREG 10 with 37,0,0.
! VLOAD 11 100 0 0      ; Load VREG 11 with 100,0,0.
! VADD 10 11            ; Add contents of VREG 10 and VREG 11 and place
                          result in VREG 10.
! READVR 10             ; Read contents of VREG 10.
  137 000 000             (Response from Model One.)
```

VAL1K VAL1K

SYNTAX

ASCII	VAL1K val	
FORTRAN Call	CALL VAL1K (IRGBV)	
Binary	[176] [val]	(2 bytes)
	176 decimal = 260 octal = B0 hex	

FUNCTION

The VAL1K command is provided for compatibility with the Model One/40. Unless you require this compatibility, use the VAL8 or VALUE commands instead.

The VAL1K command sets the current pixel value (VREG 0) to the value val. All operations which write into image memory use this value.

Val is used as an index into the look-up table. The value val is treated modulo 64. Therefore, VAL1K only allows addressing of look-up table indices 0 to 63.

ASCII PARAMETER

val	Current pixel value; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2	IRGBV

VAL8 VAL8

SYNTAX

ASCII	VAL8 val
FORTRAN Call	CALL VAL8 (IVAL)
Binary	[134] [val] (2 bytes)
	134 decimal = 206 octal = 86 hex

FUNCTION

The VAL8 command sets the current pixel value, look-up table index, (VREG 0) to the value val. All operations which write into image memory use this value.

Only the red component of VREG 0 is set by the VAL8 command; the green and blue components are set to zeros.

ASCII PARAMETER

val	The current pixel value; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 IVAL

EXAMPLE

```
! LUT8 1 255,0,0         ; Set color out for LUT index 1 to red.
! VAL8 1                 ; Set current pixel value to 1 (red).
! CIRCLE 50              ; Draw a red circle.
! LUT8 100 255 0 255     ; Change the color out for LUT index 100 to
                           255,0,255 (magenta).
! VAL8 100               ; Set current pixel value to 100 (magenta).
! CIRCLE 100             ; Draw a magenta circle.
```

SYNTAX

ASCII	VAL16 value
FORTRAN Call	CALL VAL16 (VALUE) CALL VAL16S (VALUE) Swaps high and low bytes
Binary	[69] [hivalue] [lowvalue] (3 bytes)
	69 decimal = 105 octal = 45 hex

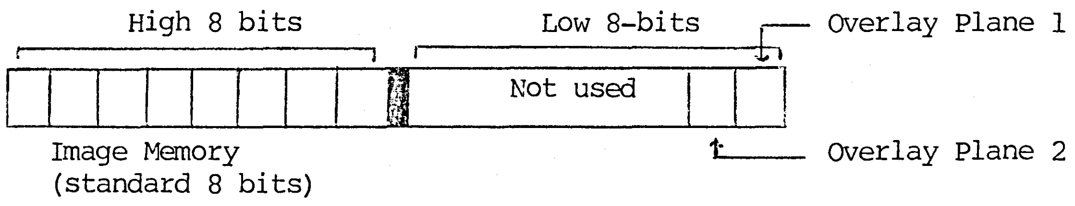
FUNCTION

The VAL16 command changes the current pixel value (VREG 0) by loading the high and low bytes directly into the value register.

The VAL16 command swaps the high (left) and low (right) bytes that make up the 16-bit word used for the value. In other words, 52 in hexadecimal must be entered as #5200. You should use a hexadecimal number for the value. The FORTRAN subroutine RMSK16S performs the swapping for you, so you can specify the value without swapping the bytes.

The VAL16 or VAL8 commands are more efficient on the Model One/10 than the VALUE command, and should be used unless compatibility with another Model One is a requirement.

The format for the value is indicated below.



ASCII PARAMETER

value	The 16-bit value to be loaded directly into VREG 0. Range is 0 to 65,535.

FORTRAN PARAMETER

INTEGER*2 VALUE

EXAMPLE

```
! WMSK16 #FF03          ; Set the write and read masks to write into and
                          read from all 10 bit planes:
                          FF is 8 bits for graphics
                          03 is for the overlay planes.
! LUT16 350 0 255 0     ; Load look-up table index 350 with all green.
! VAL16 #5401           ; Set current pixel value to value of look-up
                          table index 350 (decimal). Note that 350 deci-
                          mal = 154 hexadecimal, which for VAL16 has its
                          bytes swapped, so that the "54" is in the high
                          (left) byte, and the "01" is in the low byte.
! DRWREL 100 0          ; Draw a green vector.
```

VALUE VALUE

SYNTAX

ASCII	VALUE r, g, b
FORTRAN Call	CALL VALUE (IRED, IGRN, IBLU)
Binary	[6] [r] [g] [b] (4 bytes)

FUNCTION

The VALUE command changes the current pixel value (VREG 0) to the value specified by _r_, _g_, _b_. All operations which write into image memory use VREG 0, the current pixel value.

Only the first byte (red) is used as an index into the look-up table with RGBTRU OFF. With RGBTRU ON, the high 2 bits are used from the red, green, and blue values.

ASCII PARAMETERS

r, g, b	8-bit parameters specifying the red, green, and blue components of the current pixel value; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2	IRED, IGRN, IBLU

VECPAT VECPAT

SYNTAX

ASCII	VECPAT mask	
FORTRAN Call	CALL VECPAT (MASK)	
Binary	[46] [mask]	(2 bytes)
	46 decimal = 056 octal = 2E hex	

FUNCTION

The VECPAT command specifies the pattern of pixels to be repeated along every subsequent line that is drawn. The lines can be solid, dotted, dashed, or any other repetitive pattern that you specify.

The VECPAT command sets the vector generator pattern register to a 16-bit value specified by mask. Every time a pixel is generated by the vector generator, the mask is rotated by one bit. If the least significant bit of the mask is set to 1, the pixel will be written into image memory. If the bit is reset to 0, the pixel will be skipped.

The VECPAT command affects all line drawing commands and all graphics primitive commands, such as RECTAN, CIRCLE, and so on.

The VECPAT command also affects the filling of graphics primitives when PRMFIL is on, as well as the fill pattern used when the CLEAR command is executed.

The default mask for VECPAT is #FFFF, so that every pixel along the line is drawn.

ASCII PARAMETER

mask	16-bit pattern register mask; range is 0 to 65,535.

FORTRAN PARAMETER

INTEGER*2	MASK

EXAMPLE

```
! MOVABS 0 0            ; Move current point to 0,0.
! VECPAT #F0F0          ; Set vector pattern mask to draw lines with dashes.
! DRWABS 100 100        ; Draw dashed line.
! RECTAN 0              ; Draw a rectangle with dashed lines.
! VECPAT #AAAA          ; Set vector pattern mask to draw dotted lines.
! DRWABS 200 200        ; Draw dotted line.
! VECPAT #00FF          ; Set vector pattern mask to draw lines with long
                          dashes.
! DRWABS 0,200          ; Draw dashed line.
! MOVABS -200 -200      ; Move current point to -200,-200.
! PRMFIL ON             ; Select filled primitives.
! CIRCLE 50             ; Draw a circle filled with the pattern set with
                          the vector pattern mask.
```

SYNTAX

ASCII	VLOAD vreg, r, g, b
FORTRAN Call	CALL VLOAD (IVREG, IRED, IGRN, IBLU)
Binary	[164] [vreg] [r] [g] [b] (5 bytes)
	164 decimal = 244 octal = A4 hex

FUNCTION

The VLOAD command loads the value register specified by vreg with the pixel value specified by r,g,b. The Model One/10 stores three bytes of information, although it only uses the first 8 bits of value information (i.e., the red component).

ASCII PARAMETERS

vreg	Value register; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.
r, g, b	Red, green, and blue components of the pixel value; range is 0 to 255. Use only the red component.

FORTRAN PARAMETERS

INTEGER*2	IVREG, IRED, IGRN, IBLU

EXAMPLE

```
! VLOAD 13 50 0 0        ; Load VREG 13 with 50,0,0.
! READVR 13              ; Read contents of VREG 13.
  050 000 000              (Response from Model One.)
```

SYNTAX

ASCII	VMOVE vdst, vsrc
FORTRAN Call	CALL VMOVE (IVDST, IVSRC)
Binary	[165] [vdst] [vsrc] (3 bytes)
	165 decimal = 245 octal = A5 hex

FUNCTION

The VMOVE command copies the value in the value register specified by vsrc to the value register specified by vdst. All three bytes are copied.

On the Model One/10, only the first byte of the value register is used as an index into the look-up table. As a result, any information in the second and third bytes is ignored.

ASCII PARAMETERS

vdst	Destination value register; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.
vsrc	Source value register; range is 0 to 63.

FORTRAN PARAMETERS

INTEGER*2 IVDST, IVSRC

EXAMPLE

```
! VLOAD 10 25 0 50      ; Load VREG 10 with 25,0,50.
! VMOVE 11 10           ; Move contents of VREG 10 into VREG 11.
! READVR 11             ; Read contents of VREG 11.
  025 000 050             (Response from Model One.)
```

VSUB

SYNTAX

ASCII	VSUB vdif, vreg
FORTRAN Call	CALL VSUB (IVDIF, IVREG)
Binary	[167] [vdif] [vreg] (3 bytes)
	167 decimal = 247 octal = A7 hex

FUNCTION

The VSUB command subtracts the contents of value register vreg from the contents of value register vdif and places the result into value register vdif.

vdif = vdif - vreg

Only the first byte of the value register is used as an index into the look-up table.

ASCII PARAMETERS

vdif, vreg	Value registers; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

```
INTEGER*2     IVDIF, IVREG
```

EXAMPLE

```
! VLOAD 10 25 0 0       ; Load VREG 10 with 25,0,0.
! VLOAD 11 100 0 0      ; Load VREG 11 with 100,0,0.
! VSUB 11 10            ; Subtract contents of VREG 10 from contents of
                          VREG 11 and place result in VREG 11.
! READVR 11             ; Read contents of VREG 11.
  075 000 000             (Response from Model One.)
```

SYNTAX

ASCII	VTEXT1 string
FORTRAN Call	CALL VTEXT1 (STRLEN, STRING)
Binary	[147] [strlen] ([char1][char2]...[charn])
	147 decimal = 223 octal = 93 hex

FUNCTION

The VTEXT1 command draws a vertical text string into image memory with Font 1. If the text size is the default 16, each character uses 7 x 9 pixels. Descenders take an additional two pixels.

The parameter string specifies the text to be drawn. If the command is entered in ASCII mode from the local alphanumeric terminal or keyboard, then the string to be drawn is the set of ASCII characters which follow the VTEXT1 command on the command line. If the VTEXT1 command is sent from the host, then the length of the string must be specified. Strlen specifies the number of characters in the string, and is followed by strlen bytes containing the ASCII characters to be drawn.

The current point (CREG 0) specifies the starting point for the text string and remains unchanged by the command.

You can specify the size of the text and the baseline angle of the text using the TEXTC or TEXTN commands.

Transformations affect the rotation, scaling, and translation of text that is drawn with any of the text drawing commands.

ASCII PARAMETER

string	The text to be drawn.

FORTRAN PARAMETERS

```
INTEGER*2     STRLEN, STRING(1)
```

STRLEN is an integer specifying the number of characters that are to be drawn. STRING is an integer array with two characters packed per 16-bit word, as in FORTRAN A2 format.

EXAMPLE

```
! MOVABS 0 0            ; Move current point to 0,0.
! VTEXT1 Vertical text  ; Draw text string in default size 16 and angle 0.
! MOVABS 20 0           ; Move current point to 20,0.
! TEXTC 32 0            ; Change text size to 32.
! VTEXT1 Double-sized text.
! TEXTC 16 90           ; Change text size to 16, angle to 90.
! VTEXT1 Text at a right angle.
```

VTEXT2

SYNTAX

ASCII	VTEXT2 string
FORTRAN Call	CALL VTEXT2 (STRLEN, STRING)
Binary	[148] [strlen] ([char1][char2]...[charn])
	148 decimal = 224 octal = 94 hex

FUNCTION

The VTEXT2 command draws a vertical text string into image memory with Font 2. Font 2 is a user defined character set which is downloaded using the TEXTDN command.

At power-on or COLDstart, each character in Font 2 defaults to the same character in Font 1 (Font 2 will appear twice as large as Font 1). When Font 2 is downloaded, each character replaces the power-on default definition.

The VTEXT2 command is issued in the same manner as the VTEXT1 command. The current point (CREG 0) specifies the starting point for the text string and remains unchanged by the command. Strlen specifies the length of the string when text is not sent in ASCII mode.

ASCII PARAMETER

string	The text to be drawn.

FORTRAN PARAMETERS

INTEGER*2	STRLEN, STRING(1)

STRLEN is an integer specifying the number of characters that are to be drawn. STRING is an integer array with two characters packed per 16-bit word, as in FORTRAN A2 format.

WAIT

SYNTAX

ASCII	WAIT frames	
FORTRAN Call	CALL WAIT (FRAMES)	
Binary	[61] [highframes][lowframes]	(3 bytes)
	61 decimal = 075 octal = 3D hex	

FUNCTION

The WAIT command waits for the number of frame times specified by frames before continuing command execution. The WAIT command is often used for choreographing graphic displays and synchronizing updates with vertical blanking.

The delay in command execution in seconds is frames divided by 60. If the frames parameter is zero, command execution will stop until the next vertical blanking interval.

ASCII PARAMETER

frames	16-bit parameter specifying the number of frame times to wait; range is 0 to 65,535.

FORTRAN PARAMETER

INTEGER*2	FRAMES

WARM WARM

SYNTAX

ASCII	WARM
FORTRAN Call	CALL WARM
Binary	[254] (1 byte)
	254 decimal = 376 octal = FE hex

FUNCTION

The WARM command reintializes the Model One firmware. This clears the serial input and output queues, reinitializes the functions set with DIP switches or the SYSCFG command, and returns the system to Alpha mode.

You can also execute the WARM command by pressing the WARMstart special character (default CTRL V or CTRL Y). Note that disabling the WARMstart special character does not disable the WARM command.

SYNTAX

ASCII	WINDOW x1, y1, x2, y2
FORTRAN Call	CALL WINDOW (IX1, IY1, IX2, IY2)
Binary	[58] [highx1][lowx1] [highy1][lowy1] [highx2][lowx2] [highy2][lowy2] (9 bytes)
	58 decimal = 072 octal = 3A hex

FUNCTION

The WINDOW command sets the current clipping window to the rectangle specified by x1,y1,x2,y2. The lower left corner of the window (CREG 9) is specified by x1 and y1. The upper right corner of the window (CREG 10) is specified by x2 and y2. X1 must be less than or equal to x2, and y1 must be less than or equal to y2.

All vectors and graphics primitives are clipped to the current clipping window.

The default clipping window is equal to the physical memory size.

ASCII PARAMETERS

x1, y1	16-bit parameters specifying the lower left corner of clipping window; range is -32,768 to 32,767.
x2, y2	16-bit parameters specifying the upper right corner of clipping window; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2	IX1, IY1, IX2, IY2

EXAMPLE

```
! WINDOW -20 -20 30 30       ; Set current clipping window to rectangle
                               with opposite corners at -20,-20 and 30,30.
! MOVABS 0 0                 ; Move current point to 0,0.
! CIRCLE 25                  ; Draw circle of radius 25; only part of the
                               circle is displayed.
! TEXT1 Clipped text         ; Draw text string; only part of the text is
                               displayed.
! CLEAR                      ; Clear current window to current pixel value.
```

SYNTAX

ASCII	WMSK16 mask
FORTRAN Calls	CALL WMSK16 (MASK) CALL WMK16S (MASK)
Binary	[68] [highmask] [lowmask] (3 bytes)
	68 decimal = 104 octal = 44 hex

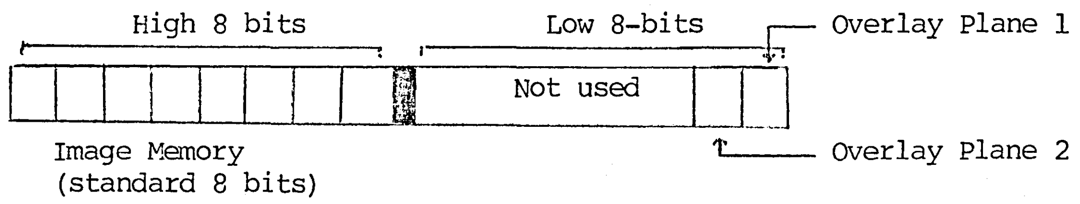
FUNCTION

The WMSK16 command specifies a 16-bit write mask for the Model One's image memory planes. Only the high byte of the write mask is used. The high eight bits correspond to the 8 bit planes of image memory. The low byte of the mask is used to control the overlay planes. If a bit is set (1), the corresponding bit plane is write-enabled; if the bit is cleared (0), the corresponding bit plane may not be written into.

When used in conjunction with the RMSK16 command (read mask), the WMSK16 command can be used for double buffering and animation by writing into those bit planes not displayed and displaying those bit planes not being written into.

The write and read masks can also be used to store multiple images in image memory and select them for display.

The format for the mask is shown below:



The WMSK16 command swaps the high (left) and low (right) bytes that make up the 16-bit word used for the mask. In other words, 52 in hexadecimal must be entered as #5200. You should use a hexadecimal number for the mask. The FORTRAN subroutine WMK16S performs the swapping for you, so you can specify the mask without swapping the bytes.

ASCII PARAMETER

mask	The 16-bit read mask; range is 0 to 65,535 (#0000 to #FFFF).

FORTRAN PARAMETER

```
INTEGER*2     MASK
```

EXAMPLE

```
! MOVABS 0 0            ; Move current point to 0,0.
! PRMFIL ON             ; Select filled primitives.
! WMSK16 #0100          ; Write enable only bit plane 0.
! LUT8 1 255 0 0        ; Change the color out for LUT index 1 to red.
! VAL8 1                ; Set current pixel value to 1 (255,0,0: red).
! RECTAN 100 100        ; Draw a filled red square.

! MOVABS 200 200        ; Move current point to 200,200.
! WMSK16 #1000          ; Write enable only bit plane 4.
! LUT8 16 0 255 0       ; Change the color out for LUT table index 16 to
                          green.
! VAL8 16               ; Set current pixel value to 16 (0,255,0: green).
! CIRCLE 50             ; Draw green circle.

! RMSK16 #0100          ; Read enable only bit plane 0; only the red
                          square is displayed.
! RMSK16 #1000          ; Read enable only bit plane 4; only the green
                          circle is displayed.
! RMSK16 #1100          ; Read enable bit planes 0 and 4; both the circle
                          and the square are displayed.
```

SYNTAX

ASCII	WRMASK bitm, bankm
FORTRAN Call	CALL WRMASK (IBITM, IBANKM)
Binary	[157] [bitm] [bankm] (3 bytes)
	157 decimal = 235 octal = 9D hex

FUNCTION

We recommend the use of the WMSK16 command instead of the WRMASK command.

The WRMASK command enables or disables write operations into the 8 image memory planes of the Model One. The two parameters, bankm and bitm, are used together to specify the write mask. Bankm specifies which pairs of bit planes to write-enable. Bitm specifies whether the lower planes, upper planes, or both planes of each pair are to be write-enabled.

bankm

Bits 7-4	Unused: must be zeros.
Bit 3	1: Write-enable bit planes 6 and 7. 0: Do not write-enable bit planes 6 and 7.
Bit 2	1: Write-enable bit planes 4 and 5. 0: Do not write-enable bit planes 4 and 5.
Bit 1	1: Write-enable bit planes 2 and 3. 0: Do not write-enable bit planes 2 and 3.
Bit 0	1: Write-enable bit planes 0 and 1. 0: Do not write-enable bit planes 0 and 1.

bitm

bitm = 0	Do not write-enable either plane of the pair.
bitm = 64	Write-enable plane 0, 2, 4, or 6 (lower plane of the pair only).
bitm = 128	Write-enable plane 1, 3, 5, or 7 (upper plane of the pair only).
bitm = 192	Write-enable both planes of the pair.

ASCII PARAMETERS

bankm	Range is 0 to 15, as described above.
bitm	Values may be 0, 64, 128, 192, as described above.

FORTRAN PARAMETERS

INTEGER*2	IBITM, IBANKM

GENERAL SYNTAX

```
    XFORM2D  type, (reserved, arg)
```

FUNCTION

The XFORM2D command defines a linear transformation which maps 2-D coordinates in the 16-bit World Coordinate Space (WCS) into coordinates in the Display Coordinate Space (DCS).

Types of Transformations

You can define the following four types of transformations with the XFORM2D command.

XFORM2D REL	Specifies a 2 x 2 matrix which performs relative scaling and/or rotation centered about the current point; matrix elements are concatenated with the previous transformation. The new origin is at the old current point, and the new WCS currrent point is set to 0,0.
XFORM2D ABS	Specifies a 3 x 2 matrix which performs absolute scaling with translation, rotation centered about the current point with translation, or both with translation. The WCS current point is set to 0,0.
XFORM2D XLATE	The new WCS origin is set to the old current point, and the new WCS current point is set to 0,0. The DCS current point is updated.
XFORM2D RESET	Specifies a return to the default transformation (the identity). The WCS curent point is set to 0,0 and the DCS curent point is updated.

What You Have to Specify

The parameters which you specify for the XFORM2D command vary according to the type of transformation.

The RESET and XLATE types require only one parameter, type.

What You Have to Specify, continued

The REL and ABS types require a reserved parameter, which must equal zero.

For the relative transformation type, REL, you must then specify a 2 x 2 matrix, which is made up of the transformed x basis and y basis vectors.

For the absolute transformation type, ABS, you must specify a 3 x 2 matrix, which is made up of the transformed x basis and y basis vectors, as well as the displacement vector.

The transformation matrices are described in more detail below.

Default Transformation: The Identity

The default transformation is the identity, which specifies no scaling, no rotation, and no translation.

```
     x basis vector =  (1, 0)
     y basis vector =  (0, 1)
displacement vector =  (0, 0)
```

To return to the default identity, use the RESET type of the XFORM2D command.

Translation With No Scaling or Rotation

Use the XLATE type of the XFORM2D command to specify a translation with no scaling or rotation. Before executing the command, move the current point with the MOVABS or MOVREL command to reflect the translation you wish to effect (i.e. the displacement vector).

Absolute Transformation: XFORM2D ABS

For the absolute transformation type, ABS, you must specify a 3 x 2 matrix, consisting of the transformed x basis and y basis vectors, as well as the displacement vector.

Listed below are the sets of vectors (or 3 x 2 matrices) which describe scaling with translation, rotation with translation, and scaling followed by rotation with translation.

FUNCTION, continued

Absolute Transformation: XFORM2D ABS, continued

To specify no translation, use zeros in the last row.

To concatenate different transformations, apply the rules of matrix multiplication.

Scale by scale factors Sx and Sy
Translate by displacement vector D

```
     x basis vector =  (Sx, 0)
     y basis vector =  (0,  Sy)
displacement vector =  (Dx, Dy)
```

Rotate by A degrees
Translate by displacement vector D

```
     x basis vector =  (cos(A), sin(A))
     y basis vector =  (-sin(A), cos(A))
displacement vector =  (Dx, Dy)
```

Scale by scale factors Sx and Sy
Rotate by A degrees
Translate by displacement vector D

```
     x basis vector =  (Sx  cos(A),  Sy  sin(A))
     y basis vector =  (-Sx  sin(A), Sy  cos(A))
displacement vector =  (Dx, Dy)
```

Relative Transformation: XFORM2D REL

For the relative transformation type, REL, you must specify a 2 x 2 matrix. This matrix includes the x and y basis vectors as listed above, but does not include the displacement vector.

CONTENTS

The following pages describe these aspects of the XFORM2D command:

- ASCII Syntax and Parameters
- FORTRAN Subroutines and Parameters
- Host Binary Command Stream, and
- Example.

ASCII SYNTAX

Relative Transformation (type = 0 (REL))

```
XFORM2D  type, reserved, mat22
```

Absolute Transformation (type = 1 (ABS))

```
XFORM2D  type, reserved, mat32
```

Reset to Identity or Translate (type = 2 (XLATE) or 3 (RESET))

```
XFORM2D  type
```

ASCII PARAMETERS

type	The type of transformation (1 byte).
0 = REL	Specifies a 2 x 2 matrix which performs relative scaling and/or rotation centered about the current point; matrix elements are concatenated with the previous transformation. The new origin is at the old current point, and the new WCS current point is set to 0,0.
1 = ABS	Specifies a 3 x 2 matrix which performs absolute scaling with translation, rotation centered about the current point with translation, or both with translation. The WCS current point is set to 0,0.
2 = XLATE	The new WCS origin is set to the old current point, and the new WCS current point is set to 0,0. The DCS current point is updated.
3 = RESET	Specifies a return to the default transformation (the identity). The WCS current point is set to 0,0 and the DCS current point is updated.

ASCII PARAMETERS, continued

reserved	A reserved parameter which must always be zero.
mat22	A 2 x 2 transformation matrix. The order of the matrix elements is x basis vector and y basis vector. The elements must be specified in a 32-bit fixed-point format with a binary point between words; range is -128.0 to 127.0. The Model One ignores the high 8 bits.
mat32	A 3 x 2 transformation matrix. The order of the matrix elements is x basis vector, y basis vector, and displacement vector. The first 4 elements must be specified in a 32-bit fixed-point format, as for mat22. The translation elements must be 16-bit signed integers.

FORTRAN SUBROUTINES

General Form:	CALL XFM2D (TYPE, 0, MAT22, MAT32)
Relative, matrix form:	CALL XFM2D0 (0, MAT22)
Absolute, matrix form:	CALL XFM2D1 (0, MAT23)
Translate:	CALL XFM2D2
Reset to Identity:	CALL XRM2D3
Relative, vector form:	CALL XF2D0X (0,XFVEC)
Absolute, vector form:	CALL XF2D1X (0, XFVEC)

FORTRAN PARAMETERS

```
INTEGER*2     TYPE

REAL*4        MAT22(2,2), MAT32(3,2)

REAL*4        XFVEC(1)

    MAT22 is defined as
         MAT22(1,1) = x basis vector, x component
         MAT22(1,2) = x basis vector, y component
         MAT22(2,1) = y basis vector, x component
         MAT22(2,2) = y basis vector, y component

    MAT32 is defined as
         MAT32(1,1) = x basis vector, x component
         MAT32(1,2) = x basis vector, y component
         MAT32(2,1) = y basis vector, x component
         MAT32(2,2) = y basis vector, y component
         MAT32(3,1) = displacement vector, x component
         MAT32(3,2) = displacement vector, y component

    XFVEC is defined as
         XFVEC(1) = x basis vector, x component
         XFVEC(2) = x basis vector, y component
         XFVEC(3) = y basis vector, x component
         XFVEC(4) = y basis vector, y component
         XFVEC(5) = displacement vector, x component
         XFVEC(6) = displacement vector, y component
```

For subroutine XF2D1X, only XFVEC(1) through XFVEC(4) is used.

Relative Transformation

```
[125] [type = 0] [0] [xb03][xb02][xb01][xb00]   [xb13][xb12][xb11][xb10]
                     [yb03][yb02][yb01][yb00]   [yb13][yb12][yb11][yb10]
                                                               (19 bytes)
```

Absolute Transformation

```
[125] [type = 1] [0] [xb03][xb02][xb01][xb00]   [xb13][xb12][xb11][xb10]
                     [yb03][yb02][yb01][yb00]   [yb13][yb12][yb11][yb10]
                     [highxt][lowxt] [highyt][lowyt]           (23 bytes)
```

Reset to Identity

```
[125] [type = 2]             (2 bytes)
```

Translate

```
[125] [type = 3]             (2 bytes)
```

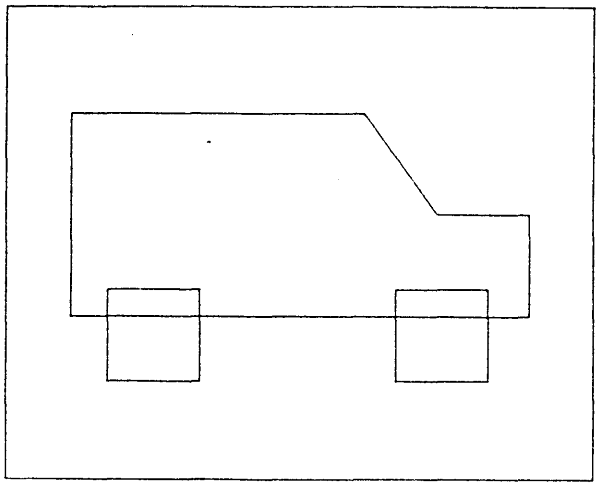
EXAMPLE

This example illustrates four different transformations: the identity, an absolute scaling, a relative rotation, and a translation. The Segment 1 which is executed in this example is defined in the example for the SEGDEF command.

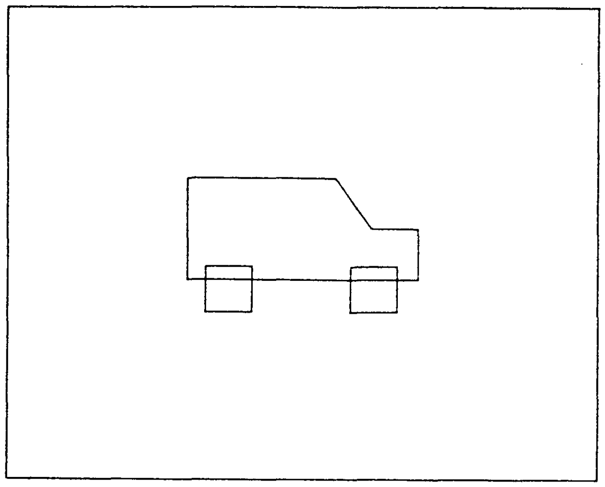
```
! SEGREF 1                ; Execute (draw) Segment 1. (See figure labelled
                            IDENTITY TRANSFORMATION).

! XFORM2D ABS 0 .5 0 0 .5 0 0
                          ; Define an absolute transformation which scales
                            by half.
! VAL8 0                  ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1                ; Draw Segment 1. (See figure labelled AFTER ABSOLUTE
                            SCALING.)
! XFORM2D REL 0 .866 .5 -.5 .866
                          ; Define a relative transformation of 30 degrees
                            rotation.  This transformation is concatenated
                            with the absolute scaling transformation.
! VAL8 0                  ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1                ; Draw Segment 1. (See figure labelled AFTER RELATIVE
                            ROTATION.)

! MOVABS 250,150          ; Move current point to 250,150.
! XFORM2D XLATE           ; Define a transformation which translates the WCS
                            origin to the old current point.  This translation
                            is concatenated with the current transformation
                            of 1/2 scaling and 30 degree rotation.
! VAL8 0                  ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1                ; Draw Segment 1. (See figure labelled AFTER
                            TRANSLATION.)

! XFORM2D RESET           ; Set the current transformation to the identity.
! VAL8 0                  ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1                ; Draw Segment 1.  (See the first figure labelled
                            IDENTITY TRANSFORMATION).
```

EXAMPLE, continued

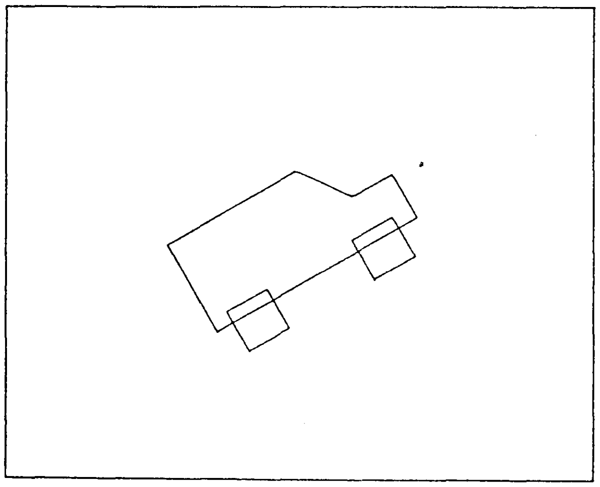
IDENTITY TRANSFORMATION



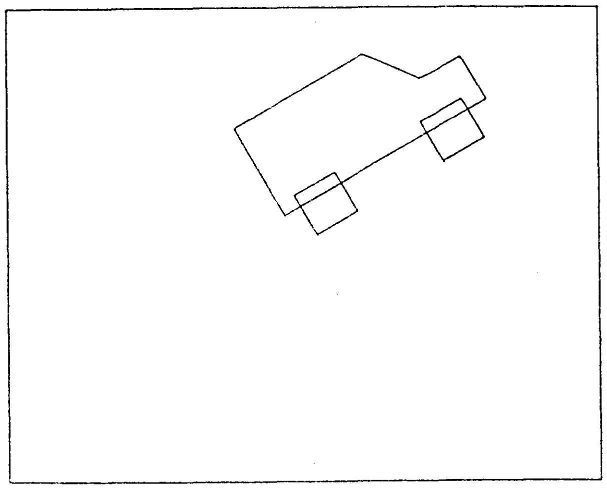
AFTER ABSOLUTE SCALING



AFTER RELATIVE ROTATION



AFTER TRANSLATION



XHAIR XHAIR

SYNTAX

ASCII	XHAIR num, flag
FORTRAN Call	CALL XHAIR (NUM, IFLAG)
Binary	[156] [num] [flag] (3 bytes)
	156 decimal = 234 octal = 9C hex

FUNCTION

The Model One/10 includes four types of crosshairs. These are in hardware and are non-destructive.

You can select from four types of crosshairs (only one crosshair at a time).

XHAIR 0 selects a 16 x 16 crosshair.
XHAIR 1 selects a full-screen cursor.
XHAIR 2 selects a blinking 16 x 16 crosshair.
XHAIR 3 selects a blinking full-screen crosshair.

Model One/10 crosshairs are XORed with the image memory.

Using the crosshairs slightly
increases the processing time for some commands.
If maximum performance is important, turn off the crosshairs
whenever possible.

ASCII PARAMETERS

num	Crosshair number; crosshairs 0, 1, 2, and 3 are available on the Model One/10.
flag	Flag = 1 or ON, enable crosshair; flag = 0 or OFF, disable crosshair.

FORTRAN PARAMETERS

INTEGER*2 NUM, IFLAG

EXAMPLE

```
! XHAIR 0 ON            ; Crosshair 0 (16 x 16 crosshair) is displayed.
! XHAIR 1 ON            ; Crosshair 1 (full-screen crosshair) is displayed.
! XHAIR 1 OFF           ; Crosshair 1 is no longer displayed.
```

SYNTAX

ASCII	XMOVE type, creg2, creg1	
FORTRAN Call	CALL XMOVE (TYPE, DSTREG, SRCREG)	
Binary	[127] [type] [creg2] [creg1]	(4 bytes)
	127 decimal = 177 octal = 7F hex	

FUNCTION

The XMOVE command performs a 2-D transformation on a specified x,y coordinate. The type parameter specifies which of the two XMOVE options is used:

- map from the World Coordinate Space (WCS) into the Device Coordinate Space (DCS), using the current 2-D transformation, or

- map from the DCS into the WCS, using the inverse of the current 2-D transformation.

The creg1 parameter contains the pre-transformed coordinate. Creg2 is automatically loaded with the transformed coordinate.

Note: If you set creg1 to 0 or CURPNT (the current point), you can see where geometry will be located with a new transformation before actually drawing the segments.

ASCII PARAMETERS

type The type of transformation.

0 = WCSDCS	Map from WCS to DCS.
1 = DCSWCS	Map from DCS to WCS.

creg1 The coordinate register containing the pre-transformed coordinate. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

creg2 The coordinate register in which to store the transformed coordinate.

FORTRAN PARAMETERS

```
INTEGER*2     TYPE, DSTREG, SRCREG
```

EXAMPLE

```
! XFORM2D ABS  0 2 0 0 2 0 0
                        ; Define an absolute transformation which scales
                          by 2.
! MOVABS 10 20          ; Move the current point to 10,20.

! XMOVE WSCDCS 20 CURPNT
                        ; Map the current point from WCS to DSC and store
                          in CREG 20.
! READCR 20             ; Read back the contents of CREG 20.
  00020 00040           ; The current point is scaled by 2.

! XMOVE DCSWCS 21 CURPNT
                        ; Map the current point from DSC to WSC and store
                          in CREG 21.
! READCR 21             ; Read back the contents of CREG 21.
  00005 00010           ; The current point is scaled by 1/2.
```

SYNTAX

ASCII	XYDIG flag,fpformat,x,y	(if flag = 0 or XYSCALE)
	XYDIG flag, state	(if flag = 1 or CLIPWRAP)
FORTRAN Call	CALL XYDIG (FLAG, FPFORM, X, Y, STATE)	
Binary	[172] [flag = 0] [fpformat] [x] [y]	(11 bytes)
	[172] [flag = 1] [state]	(3 bytes)
	172 decimal = 254 octal = AC hex	

Note: [x] and [y] are each 4 bytes, arranged as High/Low/High/Low, with the first two bytes representing whole units and the second two representing fractional units.

FUNCTION

The XYDIG command has two different functions, depending on the value of flag

When flag = 0 or XYSCALE, the XYDIG command allows you to define the scalin factor to be applied to the XY digitizer being used with the Model One Coordinate data in CREG 2 will be determined in terms of the scaling facto you establish with the XYDIG command.

When flag = 1 or CLIPWRAP, the XYDIG command allows you to cause CREG 2 to b clipped or wrapped to screen coordinates.

The options for state are:

0	No clipping or wrapping (original case)
1	Clip, relative mouse only
2	Wrap

State = 1 (clipping) causes CREG 2 to be clipped at screen bounds (-640 t 639, -511 to 512 for a 1024 x 1280 system). Zooming or changing the clippin window (i.e. CREGs 9 and 10) does not affect this.

State = 2 (wrapping) observes the same limits as clipping, but instead c clipping, the CREG value will wrap around within the range of the scree coordinates.

FUNCTION, continued

Notes: The clipping/wrapping feature executes at the time that CREG 2 is updated from the graphics input device. Changes to CREG 2 that occur at other times (such as through CLOAD or CADD commands or via the bipolar processor) are not clipped or wrapped unless or until an incoming packet is processed from a mouse or tablet.

Wrapping can be used with any graphics input device. Clipping can only be used with a relative mouse, the M2 mouse. If you attempt to select clipping with a tablet configured, you will see the message: ILLEGAL PARAMETER.

ASCII PARAMETERS

flag	Flag indicates type of scaling to perform. 0 = XYSCALE Define scaling factor. 1 = CLIPWRAP Specify clipping or wrapping for CREG 2. In binary format, flag must be 0 or 1.
fpformat	Specifies that the x and y parameters are in floating point format (16.16). Fpformat must be 3.
x,y	X and y scaling factors. In ASCII format, these may be a value from 0.00000 to 8.0. The binary format must be 16.16 format. The default scaling factor is 1,1.
state	0 No clipping or wrapping (default) 1 Clip, relative mouse only 2 Wrap

FORTRAN PARAMETERS

INTEGER*2	FLAG, FPFORM, STATE
REAL*4	X, Y

MNEMONICS FOR COORDINATE REGISTER AND VALUE REGISTER PARAMETERS

The table below lists the mnemonics which you can use for coordinate register parameters. For coordinate registers 11 through 63, there are no mnemonics.

CREG #	Mnemonic	Description
0	CURPNT	Current point. Starting point of graphics primitives commands. Updated by a MOVE or a DRAW.
1	JOYSTK	Unscaled XY digitizer location; current coordinates from digitizer. Updated automatically.
2	DIGTZR	Scaled XY digitizer location; current coordinates from digitizer. Updated automatically.
3	CORORG	Coordinate origin. Coordinates of the center of image memoroy.
4	SCRORG	Screen origin. Coordinates of the pixel in the center of the screen.
5	XHAIR0	Crosshair 0 location in image memory.
6	XHAIR1	Crosshair 1 location in image memory.
7-8		Reserved.
9	LWNORG	Clipping window origin. Lower-left corner of current clipping window. All vectors are clipped to this window.
10	UWNORG	Clipping window origin. Upper-right corner of current clipping window. All vectors are clipped to this window.

The table below lists the mnemonics which you can use for value register parameters. For value registers 4 through 63, there are no mnemonics.

VREG #	Mnemonic	Description
0	CURVAL	Current value; value used in all graphics primitives commands.
1	XR0VAL	Value used for Crosshair 0.
2	XR1VAL	Value used for Crosshair 1.
3	FILMSK	Fill mask used for area fills.

Raster Technologies

Model One

TEKTRONIX 4014 EMULATION

Revision 3.0

November 15, 1985

Part # 552-00039-001

Raster Technologies, Inc.
Two Robbins Road
Westford, MA 01886
(617) 692-7900

TEKTRONIX EMULATOR TECHNICAL NOTE
December 2, 1983

1.0 INTRODUCTION

The Model One TEKEM command enters the TEKTRONIX 401X emulator for the Raster Technologies Model One. This emulator emulates the Tektronix models 4006, 4010, 4012, 4013, 4014, 4015, 4016, and 4054. When the Tektronix emulator is invoked, the Model One command set is unavailable and the 401X emulator is entered. Any host program/graphics package combination (e.g., Plot 10, GCS, DIGRAF) that has been used to drive the 401x terminal will then be able to drive the Model One.

This technical note is divided into the following sections:

TEKEM Command Syntax: section 2.0
Standard Emulation: section 3.0
Extensions to 401x Emulation: section 4.0
Escape Sequences: section 5.0

Each of the latter three categories is subdivided into host-controlled functions and keyboard-controlled local functions.

It is assumed that the reader of this document is familiar with the 4014 command set and the Model One.

NOTE: in using the Tektronix emulator (TEKEM), the ESCAPE key is used to invoke command sequences. Thus, if you are using the alphanumeric terminal for ESCAPE sequences, he may have some problems. In this case, these commands will correct the problem:

```
SPCHAR WARM OFF
SPCHAR BREAK OFF
SPCHAR KILL ON #15
```

The kill character is now CTRL-U; BREAK and WARMstart keys are disabled.

To use ESCAPE sequences from the host computer, you can disable all CTRL characters with the command:

```
SYSCFG SERIAL HOSTSIO CTRL OFF
```

This configuration is the default.

2.0 COMMAND SYNTAX

The Tektronix emulator is invoked with the Model One command:

```
     TEKEM <bitflag>
```

where <bitflag> is an 8-bit parameter, the bits of which are set as follows:

```
   Bit number (0=LSB) :     7  6  5  4  3  2  1  0
                            -  -  -  -  -  -  -  -
                            0  0  0  P  S  D  G  G
```

Here,

S signals screen mode: it is 0 for normal screen mode and 1 if the terminal is to send an XOFF (^S) to the host upon a page-full condition. In the latter case, an XOFF (^Q) will be sent after the user clears the page.

D signals duplex mode: it is 0 for full duplex and 1 for half duplex.

GG signals graphics terminating character mode: it is 0 if no terminating character is to be sent after sending coordinates, it is 01 if a carriage return is to be sent, and it is 10 (=2 decimal) if a carriage return followed by EOT is to be sent.

P signals that different line types will be treated differently. If P=0, all line types will be INSerted (pixel function INS). This is the default.

If P=1, dotted and dashed lines use pixel function CONDitional. Solid lines may be INSerted, XORed, or ORed, as determined by the ESCAPE sequences in section 5.

Thus, for example, TEKEM 6 would signal normal screen mode, half duplex mode, and <CR><EOT> to follow coordinate data.

3.0 STANDARD EMULATION

3.1 General Description

The TEKEM emulator is designed to provide 401X functionality. However, some "gray areas" arise from physical differences between the Model One and the Tektronix terminals. These differences are described in the next section.

3.2 Host-Controlled Functions

3.2.1 Addressability

The TEKEM emulator will respond to either 10-bit (1024x1024) or 12-bit (4096x4096) coordinate addressing. In the 12-bit mode, the "extra" byte sent by the host is ignored; the emulator will expect

HIY, LOY [EXTRA (ignored)], HIX, LOX.

In other words, the emulator will accept 12-bit addressing but supports 10 bits of address.

3.2.2 Graphic Input Mode (GIN)

Upon receipt of the sequence <ESC> <SUB>, the emulator will respond by displaying the crosshair. GIN will be terminated by the user typing any character. This character will NOT be displayed on the screen. GIN termination will generate 5 bytes in the following sequence:

Byte 1	character (GIN terminator)
Byte 2	HIX
Byte 3	LOX
Byte 4	HIY
Byte 5	LOY

This sequence will be terminated by a CR, a CR EOT, or by nothing, depending on the two "GG" bits of the TEKEM parameter.

3.2.3 Alpha Cursor Address

With the emulator in alpha mode, sending ESC ENQ from the host will return the terminal status and lower-left coordinate of the alpha cursor to the host. The first byte returned is the terminal status; after that, the bytes are identical to GIN. The emulator stays in alpha mode after returning the information.

3.2.4 Graphic Beam Address

With the emulator in graphics mode, sending ESC ENQ from the host will return the terminal status and graph beam coordinates to the host. The byte sequence is identical to that of the alpha cursor location. After returning the information, the emulator stays in graphics mode.

3.2.5 Crosshair Address

When displaying the crosshair cursor, sending the ESC ENQ sequence from the host returns the intersection point of the crosshair to the host. A 15 msec delay must occur between ESC SUB (enabling the crosshair) and ESC ENQ. Four addresss bytes are returned. After sending the bytes, the emulator returns to alpha mode.

3.2.6 Alpha/Graphics Mode

The emulator will be in alpha mode when it is first invoked. Upon receipt of US or CR, the emulator will return to alpha mode from whatever mode it was in. In alpha mode, all bytes received are treated as alphanumeric characters. After receiving GS, all further bytes received are treated as graphic information (addresses). The first vector drawn after receipt of GS is a MOVE (invisible); subsequent vectors are DRAWs (visible).

3.2.7 Line Fonts

Sending ESC, followed by the ASCII Decimal Equivalent (ADE) numbers below, will generate all subsequent vectors in the selected line font.

ADE	Line Font
96	solid (default)
97	dotted
98	dot-dash
99	short dash
100	long dash

3.2.8 Character Size

Sending ESC followed by the appropriate ADE will select the character size.

ADE	Character Size
56	large
57	medium
58	small
59	very small

3.2.9 Point Plot Mode

Sending FS will enter Point Plot Mode. In this mode, points are addressed normally; the endpoint (the addressed point) will be drawn.

3.3 Keyboard-Controlled Functions

3.3.1 Page Key

To emulate the PAGE/RESET key of the 401x, press CTRL-W followed by P (for page). The screen is erased and the alpha cursor moves to HOME (the upper left-hand corner).

3.3.2 Other Keyboard Functions

With the exception of margin (3-way toggle) and hard-copy, all other keyboard functions are supported.

4.0 EXTENSIONS TO THE 40XX COMMAND SET

4.1 Overview

In addition to providing all the features of the 401x graphic tarminals, the emulator provides access to the powerful command set of the Model One. Some of the more popular commands such as changing the background color, changing the vector color, XHAIR color, areafill, and so forth, are made directly available. Most ot these functions may be invoked either from the host or by keyboard entry. On the host, the commands may be sent through a routine that outputs ASCII strings (TOUTST for example in PLOT 10), or the user may write a set of routines that output specific command strings. Each of these extensions will be discussed.

4.2 Host-Controlled Functions

Throughout this section, the user must be aware that all parameters are ASCII numbers. For example, num in sections 4.2.1 and 4.2.2 is the ASCII numbers '0' through '4'.

4.2.1 Selected-Screen Quadrant

ESC DC4 (CTRL-T) 0 num selects the quadrant on the screen based on num. The range of num is 0 to 4. Number 0 selects the whole screen, while numbers 1-4 select the corresponding screen quadrants. Quadrant 1 is upper left, 2 is upper right, 3 is lower left, and 4 is lower right. When a quadrant is selected, all incoming data is scaled appropriately to fit in that quadrant. Any display manipulation command, such as erase, received from the host affects only that quadrant, thereby providing selective quadrant erasing.

The host binary command stream is:

```
[1BH][14H][30H][num]        (4 bytes)
```

This function is invoked by sending:

```
[ESC][DC4 (CTRL-T)][0][num]
```

The range for num is 0 to 4.

4.2.2 View Screen Quadrant

ESC DC4 1 num selects the quadrant for viewing on full screen; each quadrant will be multiplied by a factor of 2 to fill the entire screen (2X zoom).

The host binary command stream is:

```
[1BH][14H][31H][num]        (4 bytes)
```

This function is invoked by sending:

```
[ESC][DC4][1][num]
```

The range of num is 0 to 4.

4.2.3 Monitor Toggle

ESC DC4 2 toggles between the monitor and alpha terminal for display of text.

The host binary command stream is:

```
[1BH][14H][32H]       (3 bytes)
```

This function is invoked by sending:

```
[ESC][DC4][2]
```

4.2.4 Termination

ESC DC4 3 terminates Tektronix emulation and returns to the Model One command environment.

The host binary command stream is:

```
[1BH][14H][03H]     (3 bytes)
```

This function is invoked by sending:

```
[ESC][DC4][3]
```

4.2.5 Change Background Color

ESC DC4 4 N sets the background color to N. This color is used the next time the screen is cleared. The color is specified as:

```
MSB 7 6 5 4 3 2 1 0 LSB
    X X R R G G B B
```

The host binary command stream is:

```
[1BH][14H][34H][highN][lowN]     (5 bytes)
```

The function is invoked by sending:

```
[ESC][DC4][4][N]
```

Note that N must be sent as two HEXASCII digits.

4.2.6 Change Foreground Color

ESC DC4 5 N sets the foreground color to N immediately. (N is described in the previous section.)

The host binary command stream is:

[1BH][14H][35H][highN][lowN]   (5 bytes)

The function is invoked by sending:

[ESC][DC4][5][N]

4.2.7 Change Crosshair Color

ESC DC4 6 N sets the crosshair color to N. (N is described in section 4.2.5.)

The host binary command stream is:

[1BH][14H][36H][highN][lowN]   (5 bytes)

This function is invoked by sending:

[ESC][DC4][6][N]

4.2.8 Area Fill

ESC DC4 7 performs an area fill, using the current color, starting at the current cursor position and bounded by any pixel in a different color. (See the AREA1 command in the Model One documentation for further details.)

The host binary command stream is:

[1BH][14H][37H]   (3 bytes)

This function is invoked by sending:

[ESC][DC4][7]

4.2.9 Change Look-Up Table Entry

ESC DC4 8 Entry Value changes a single look-up table entry to a specific value. Entry ranges from 0 to 3FH; value ranges from 0 to 3FH. (See the LUT8 command in the Model One documentation for further details.)

The host binary command stream is:

[1BH][14H][38H][highE][lowE][highRV][lowRV]
[highGV][lowGV][highBV][lowBV]   (11 bytes)

This function is invoked by sending:

[ESC][DC4][8][Entry][Value]

4.2.10 Redraw and No Redraw: Stroke Terminal Simulation

ESC DC4 10 turns on redraw of vectors (simulation of a stroke terminal); ESC DC4 9 turns off redraw.

The host binary command stream is:

```
[1BH][14H][41H]    (3 bytes)     for REDRAW
[1BH][14H][39H]    (3 bytes)     for NO REDRAW
```

These commands are invoked by sending:

```
[ESC][DC4][A]       for REDRAW
[ESC][DC4][9]       for NO REDRAW
```

All vectors drawn after REDRAW are affected, until NO REDRAW is invoked.

4.2.11 Text Color

ESC DC4 11 N sets the text color to N. (N is described in section 4.2.5.)

The host binary command stream is:

```
[1BH][14H][42H][highN][lowN]      (5 bytes)
```

The command is invoked by sending:

```
[ESC][DC4][B][N]
```

4.2.12 Redraw Rate

This command sets the rate of redraw for the redraw command (see section 4.2.10, above). ESC DC4 12 N sets the redraw rate to N. N=0 is not permitted.

The host binary command stream is:

```
[1BH][14H][43H][highN][lowN]  (5 bytes)
```

The command is invoked by sending:

```
[ESC][DC4][C][N]
```

The default rate is N=20.

4.3 Locally Controlled Functions

All local commands must be entered from the alpha terminal. Each local command is preceded by CTRL-W and is followed by the command letter and any required parameters.

These commands are available:

S num	Set active sub-screen
V num	Set viewing area to num

M	Toggle text output between Model One and alphanumeric terminal
R	Terminal Tektronix emulator and return to Model One
P	Erase current subscreen (same as ESC FF)
B N	Set background color to N
F N	Set foreground color to N
C N	Set character size to N
H	Provide HELP (a list of available local commands)

5.0 ESCAPE SEQUENCES

This section compares the Model One Tektronix Emulator (TEKEM) and the actual Tektronix 4014 escape sequences.

Escape Sequence	TEKEM	TEK4014
ESC NUL		
ESC DC4 (^T)	Enter extension sequence	Stay in escape mode, interpret following characters as part of escape sequence
ESC ENQ (^E)	Same as TEK4014	Terminal status request
ESC BEL (^G)	Ring terminal bell	Ring terminal bell
ESC BS (^H)	Move left, no erase	Move left one space
ESC HT (^I)	Not applicable	Move one space right
ESC LF (^J)	Same as TEK4014	Set LCE
ESC VT (^L)	Not applicable	Move one line up
ESC FF (^L)	Same as TEK4014	Erase display; home; clear
ESC CR (^M)	Same as TEK4014	Set LCE; ignore further CRs
ESC SI (^O)	Not applicable	ASCII character set
ESC SO (^N)	Not applicable	Alternate character set
ESC ETB (^W)	Access local command set from A/N terminal	Select LOCAL commands
ESC CAN (^X)	Not applicable	Set bypass mode
ESC SUB (^Z)	Same as TEK4014	Set GIN, crosshair, and bypass modes
ESC ESC (csK)	Not applicable	Return to ANSI
ESC FS*(csL)	Same as TEK4014	Special point plot mode
ESC GS*(csM)	Same as TEK4014	Graph mode with dark vectors
ESC RS*(csN)	Not applicable	Incremental plot mode
ESC US*(cs0)	Same as TEK4014	Alpha mode
ESC 8	78 characters, 46 lines	74 characters, 35 lines
ESC 9	86 characters, 46 lines	81 characters, 38 lines
ESC :	129 characters, 70 lines	121 characters, 58 lines
ESC ;	140 characters, 76 lines	133 characters, 64 lines
ESC (96),(104),(112)	Solid line, PIXFUN INS	
ESC (97),(105),(113)	Dotted line, PIXFUN COND	
ESC (98),(106),(114)	Dot-dash line, PIXFUN COND	
ESC (99),(107),(115)	Short dash line, PIXFUN COND	
ESC (100),(108),(116)	Long dash line, PIXFUN COND	
ESC (101),(109),(117)	Solid line, PIXFUN XOR	
ESC (102),(110),(118)	Solid line, PIXFUN OR	
ESC (103),(111),(119)	Solid line, PIXFUN INS	

*FS, GS, US, and RS may be sent with or without a preceding ESC.

In addition, the Model One Tektronix emulator does not support page full conditions; it does not stop displaying alphanumeric information when the screen is full. It does not honor the "make hard copy" request. The VIEW/HOLD operation no longer exists.

102680752

Raster Technologies

551-00010-001

CUSTOMER FEEDBACK SHEET

ur feedback is very important to help Raster Technologies provide documentation that meets user needs. ›ase complete the following feedback sheet, fold in thirds, staple, and mail (postage is pre-paid). Thank you.

BACKGROUND

me ______________________

mpany ______________________

ldress ______________________

____

one ______________________

›w much experience do you have using Raster Technologies' odel One products? None ☐ Some ☐ Considerable ☐

›r what types of applications are you using the Model One?

____

____

____

RATINGS

Please rate the documentation in terms of:	Excellent	Good	Adequate	Poor
Organization	☐	☐	☐	☐
Completeness	☐	☐	☐	☐
Ease of finding information	☐	☐	☐	☐
Accuracy	☐	☐	☐	☐
Command descriptions	☐	☐	☐	☐
Examples	☐	☐	☐	☐
Illustrations	☐	☐	☐	☐

COMMENTS

'hat aspects of the documentation do you find **most** useful? ______________________

____

____

____

/hat aspects of the document do you find **least** useful? ______________________

____

____

____

√hat additions/deletions would you recommend? ______________________

____

____

____

\ny general comments? ______________________

____

____

____

Thank You!

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
First Class Permit No. 6 N. Billerica, MA

Raster Technologies

2 Robbins Road
Westford, MA 01886