Table of Contents

Examples

MACRO PROGRAMMING

The Model One's macro programming facility allows you to define a group of graphic commands to be executed together by issuing a single command whenever desired.

Up to 256 macros may be simultaneously defined and stored at any one time; a COLDstart or power-up will erase all macros.

Four commands are available for defining and executing macro commands. MACDEF and MACEND define a macro command; MACRO executes a macro; and MACERA erases a macro.

1.0 Defining a Macro

Macros are referred to by number, from 0 to 255. The command MACDEF num starts a macro definition. The command MACEND ends the macro definition. In between these two commands, you can include any number of graphics commands, including a MACRO command to execute another macro. The only graphics command which cannot be included in a macro is ASCII.

The QUIT command can be included in a macro

however, when you re-enter graphics, the Model One/80 remains in macro definition. Consequently, QUIT should be used in macros with caution.

After you issue the MACDEF command from the local terminal, you will receive the special macro definition prompt $. Graphics commands may be entered without being executed; they are included in the macro definition and executed when the macro is executed. Macro definition ends when you type the command MACEND.

For example:

```
! MACDEF 11          Start definition of macro 11:
                     user command is underlined.
$ CMOVE SCRORG DIGTZR
                     Make digitizer location (CREG 2) the
                     screen origin (CREG 4)
$ MACEND             End macro definition
```

The commands above define Macro 11. Macro 11 provides for interactive panning of the display. Every time Macro 11 is executed, the screen origin will be placed at the current digitizer location. If you want to update the screen origin constantly to reflect the digitizer origin, you can use the button table (section 5, example 4).

If you make an error--a typo or an invalid parameter-- while entering a macro definition, you will be given an error message, and the line will be erased. Macros cannot be changed or edited, however. (The only way to modify a macro is to erase it and redefine it.) You can then continue entering the macro command.

2.0 Executing a Macro

Once you've defined a macro, you can then execute it. To execute macro 11 once, type

MACRO 11

The more general form of the command is MACRO num, where num is the same number you used in defining the macro command.

Calling a macro that has not been defined does nothing. No error message is given.

3.0 Erasing a Macro

To erase any macro, you can type the command MACERA num. To erase all macros, you can use the COLDstart command. COLDstart, however, does reset a large number of other functions.

For example, you can erase Macro 11 with the command MACERA 11.

To redefine a macro, simply type MACDEF num. This will erase the old definition of the macro and provide you with a clean slate. No error is given if the macro already exists.

4.0 Suggestions for Writing Macros

As a programming practice, you should be careful to restore conditions to what they were before the macro was executed. For example, in defining macros which draw objects using MOVE and DRAW commands, you should restore the current point to its original value before the end of the macro. Relative MOVE and DRAW commands allow you to execute the macro with the current point positioned anywhere in the image memory by assigning the current point appropriately.

The examples below show several macros.

```
     MACDEF 25          Begin Macro 25
     ZOOMIN             Zoom in by factor of two
     VALUE 0,255,0      Set color to green (24-bit system)
or   VAL1K 12           Set color to green (8-bit system)
     MACEND             End Macro 25
     VAL8 255           Set color to white (255,255,255) (24-bit system)
or   VAL1K 63           Set color to white (8-bit system)
     CIRCLE 50          Draw circle of radius 50 around current point
     MACRO 25           Execute Macro 25: zoomin and change the color
                        to green
     CIRCLE 40          Draw green circle of radius 40 around current
                        point
     ZOOM 1             Restore scale factor
```

Example 1 Defining A Macro

Example 1 shows a simple macro; all it does is change the color to green and zoomin by a factor of 2.

```
     MACDEF 5              Start definition of Macro 5
     DRWREL 10,0           Draw four line segments: these define a box
     DRWREL 0,10           with the lower-left corner at the current
     DRWREL -10,0          point
     DRWREL 0,-10
     MACEND                End Macro 5
     VALUE 255,255,255     Change color to white (24-bit system)
or   VALlK 63              Change color to white (8-bit system)
     MOVABS 50,50          Move to the point (50,50)
     MACRO 5               Draw the box defined by Macro 5
     MOVABS 70,70          Move to (70,70)
     MACRO 5               Draw the box defined by Macro 5
     MACDEF 6              Define Macro 6
     MACRO 5               Macro 5 will be executed as part of
                           Macro 6
     MOVREL 20,0           Move right 20
     MACRO 5               Execute Macro 5 again
     MOVREL 0,20           Move up 20
     MACRO 5               Execute Macro 5 again
     MOVREL -20,0          Move left 20
     MACRO 5               Execute Macro 5 one more time
     MOVREL 0,-20          Move back to starting point
     MACEND                End definition of Macro 6
     MOVABS 0,0            Move to (0,0)
     MACRO 6               Execute Macro 6
                           --This will draw four boxes.
     MOVABS -50,-50        Move to (-50,-50)
     MACRO 6               Draw four more boxes.
```

Example 2 Lots of Little Boxes

Example 2 defines a macro, which in turn is executed by another macro to draw a collection of boxes. You can nest macros up to 16 deep. For example, you could set up macro 6 to be executed by another macro.

```
MACDEF 10           Define macro 10
VADD 0 10           Add contents of VREG 10 to current value
CADD 0 21           Add contents of CREG 21 to current point
MACRO 6             Execute macro 6
MACEND              End macro 10
VALUE 0,0,0         Set initial current value to black (24-bit system)
VALlK 0             Set value to black (8-bit system)
MOVABS -256,-256    Move current point to lower-left corner
VLOAD 10 1,0,4      Load VREG 10 (in 8-bit system, the
                    second two bytes will be ignored)
CLOAD 21 1,1        Load CREG 21
BUTTBL 0 10         Execute macro 10 every 1/30th second
                    (see section 5)
```

Example 3 A Macro Using Changing Colors

Example 3 takes advantage of the button table to execute macro 10 every 1/30th of a second. The net result is a set of square tubes; the shading varies according to value register 10. CADD and VADD are used to change the location and the color. Issue the command BUTTBL 0 0 to terminate execution of the macro.

5.0 Using the Button Table

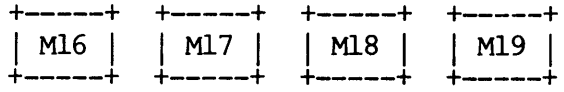
Many interactive applications require the display controller to perform graphics operations in response to user input. The digitizing tablet's cursor supplies a set of function buttons; macro commands can be set to be executed in response to these function buttons.

When a function button is pressed by the user, the button number is sent to the Model One over the TABLETSIO interface. The Model One then uses the button number to specify the macro to be executed in response to the function button. The button table keeps track of which macro should be executed in response to which button. The BUTTBL index,nmac command is used to load the button table. index gives the button number; nmac gives the number of the macro to be executed in response to that button.

The 13 or 16 buttons on the digitizing tablet cursor correspond to entries 1 through 13 (for the 13 button cursor) or entries 1 through 15 (for the 16 button cursor, where button 0 is ignored by the Model One). For example, if button 5 were pressed, entry 5 of the button table would indicate which macro should be executed in response.

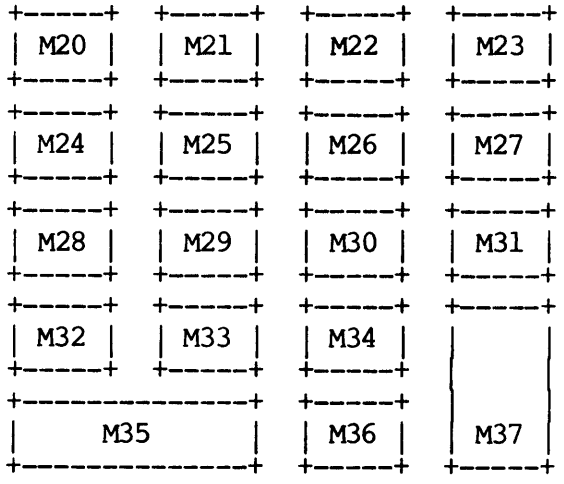
For the Raster Technologies keyboard, the numeric keypad and arrow keys can be used to execute certain macros in correspondence to the button table, as follows:

Arrow keys:



MXX=Macro Number

Numeric keypad and function keys:



Every 1/30th of a second, the Model One checks to see if any function buttons have been pressed. If a function button has been pressed, the button table is used to indicate which macro to execute; if no button has been pressed, the macro indicated by button table location zero is executed.

Button table location zero provides a "background macro"-- the macro specified by location zero is executed every 1/30th of a second when no buttons are pressed. For example, button table location zero can be used to provide interactive cursor tracking, by setting the cursor location equal to the data tablet's current coordinate every 1/30th second.

```
XHAIR 0 1          Turn on crosshair 0
MACDEF 15          Define Macro 15
CMOVE 5 2          Copy current cursor location into current
                   crosshair location
MACEND             End Macro 15
BUTTBL 0 15        Execute Macro 15 if no button is pushed
```

Example 4 Interactive Cursor Tracking

The macro running from location 0 cannot be interrupted until the final MACEND is executed. However, the macro at location 0 can interrupt other executing macros. Note that the FLUSH command (see Example 7) can be used to empty the function button event queue.

6.0 Advanced Macro Programming

This section gives several examples of macro programming, such as rubber-banding, panning, double-buffering, and movie-loop animation.

This macro continuously pans the display in steps set by the contents of CREG 20, one step every 1/30th second:

```
MOVABS 0,0
CIRCLE 50

MACDEF 10
CADD 4 20          Add the contents of CREG 20 to the
                   screen origin (CREG 4)
MACEND
CLOAD 20 10,15
BUTTBL 0 10        Execute Macro 10 every 1/30th second
```

Example 5 Display Pans Continuously

In Example 5, the panning of the display depends on the contents of CREG 20; to change the step, change CREG 20. To stop the panning entirely, change the button table entry at location 0 (BUTTBL 0 0, for example) or erase macro 10 (MACERA 10).

The digitizing tablet cursor can also be used for panning, as shown in the next example, where the cursor location controls the center of the screen.

```
MACDEF 11
CMOVE 4 2          Make the cursor location the screen origin
MACEND
BUTTBL 0 11        Execute Macro 11 every 1/30th second
```

Example 6 Cursor-controlled Panning

You can also define a macro to zoom in on the display whenever a function button is pressed:

```
MACDEF 12
ZOOMIN
FLUSH           Empty the function button event queue
MACEND
BUTTBL 2 12     Execute Macro 12 when button 2 is pressed
```

Example 7 Button-controlled Zooming

The FLUSH command in Example 7 empties the function button event queue. (The event queue is described in more detail in the manual Data Read-back and Image Transmission, with the READBU command.) The FLUSH command should be used whenever you want to make sure that no extraneous buttons have been pushed. Note that once the function button event queue has been filled, further button depressions are ignored. The function button event queue holds up to eight events.

The READBU flag,cflg command, described in detail in Data Read-Back and Image Transmission, takes an entry from the event queue and sends it to the host; the flag indicates whether or not the Model One should wait for a button to be pressed or send a button number of zero if the event queue is empty. clfg indicates whether the digitizer or joystick coordinates should be sent.

The next set of macros allow you to perform rubber-banding and confirmation of lines. To execute the set of macros, type in the three macros, enter the last three commands, and start entering lines. Button 1 confirms the endpoint of each line (and the start point of the next line).

```
MACDEF 40       Macro to draw and un-draw lines
MOVI 22         CREG 22 gives the starting point of the line
DRWI 21         CREG 21 gives the current endpoint
WAIT 1          Leave line on screen at least one frame
MOVI 22
CMOVE 21 2      Make cursor location the current endpoint
DRWI 21
MACEND
WAIT 1        Again, leave the line on the screen one frame

MACDEF 41       Macro to start rubberbanding of lines
XHAIR 0 0       Disable crosshair 0
PIXFUN 4        XOR lines with image memory
BUTTBL 0 40     Execute Macro 40 every 1/30th second
FLUSH
MACEND

MACDEF 42       Macro to confirm a given line and
MOVI 22          restart rubberbanding
DRWI 21
PIXFUN 0        Add the line to image memory
MOVI 22
DRWI 21
PIXFUN 4
```

```
CMOVE 22 21     Make line endpoint the new starting point
BUTTBL 0 40     Execute Macro 40 every 1/30th second
FLUSH
MACEND

VALUE 0,0,255   Set value to blue
MACRO 41        Execute setup macro
BUTTBL 1 42     Press button 1 to confirm the endpoint
```

Example 8 Rubberbanding Lines Create a Pattern

Example 8 sets up two states for the Model One: macro 40, which executes every 1/30th second, repeatedly draws a line (using the pixel function XOR to keep from destroying image memory), while macro 42 confirms the line and draws it into image memory. (Macro 41 sets things up.)

This sort of multi-state macro programming can be extended to support movie-loop animation and double-buffering. In movie-loop animation the bit planes of image memory are used to store a series of sequential frames from an animation sequence; the bit planes are then played back.

To perform this sort of animation, the various frames of the sequence are first loaded into the proper bit planes. The WRMASK command helps in making sure the planes are loaded correctly.

Then, to perform the playback of the images, a series of macro commands is used. Each macro displays one frame of the animation, perhaps changing the read-enable masks, look-up-tables, and screen origin. The last command of each macro changes the button table to display the next frame in the sequence. Thus, you can create a linked list of macros to display the desired animation sequence.

RASTER TECHNOLOGIES
MODEL ONE/80
APPLICATION DEVELOPMENT FEATURES

Revision 1.0 February 27, 1984

MODEL ONE APPLICATION DEVELOPMENT FEATURES
February 27, 1984

NOTICE:

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

RASTER TECHNOLOGIES DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS MATERIAL (INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE), EITHER EXPRESS OR IMPLIED. RASTER TECHNOLOGIES SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM ANY ERROR CONTAINED HEREIN, INCLUDING, BUT NOT LIMITED TO, FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS MATERIAL.

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 regard 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.

Table of Contents

Tables and Examples

APPLICATION DEVELOPMENT FEATURES

The Model One includes special features to help the applications programmer to identify and correct problems with host-based applications programs and Model One macro commands.

The local debugger allows the user to step through the command sequence being sent by the host, display a listing of currently defined macros, and issue graphics commands to the Model One from the local keyboard.

The command stream translator disassembles commands that are being executed by the Model One and displays them at the local alphanumeric terminal in mnemonic form.

The REPLAY comand allows the user to examine the last 32 characters that were sent from the host over the serial interface.

The ALPHAO strlen, string command allows ASCII data to be sent to the local terminal while the Model One is in GRAPHICS mode. This allows an applications program to issue prompt messages at the local workstation.

1.0 The Local Debugger

To enter the local debugger, type a [CTRL-X] at the terminal. See the manual Modes of Operation for more information about the special characters; in addition, the Command Reference includes complete information on the SPCHAR command. If you are in GRAPHICS mode when you type the [CTRL-X], type a carriage return after the [CTRL-X]. Once the current graphics command has been completed, the Model One responds with the debugger prompt DEBUG>. This prompt indicates that the Model One has suspended command execution and is ready for debugger commands. You can now enter any valid graphics command, as if you were in local GRAPHICS mode, or any of the special debugger commands (/Step, /List, /Macros, /V, /Continue, and /Help). Each debugger command is entered with a slash (/) and its first letter.

The /S (Step) command allows you to step through the host command stream or a local macro. /S may include a number to indicate the number of commands to execute before returning to the debugger. Once you specify a number, this number becomes the new default: /S 3 changes the default number of commands to three, instead of one. For example:

```
DEBUG> /S          Execute one command
DEBUG> /S 1000     Execute 1000 commands
DEBUG> /S 5        Execute 5 commands
DEBUG> /S          Execute 5 commands
```

The /M (Macro) command lists the numbers of all currently-defined macros. For example:

```
DEBUG> /M
0 25 100 101 102 200        Lists all non-empty macros
DEBUG>
```

The /L nmac (List macro) command lists the macro whose number is given. For example:

```
DEBUG> /M          List numbers of defined macros
0 25 100 101 102
DEBUG> /L 100      List contents of macro number 100
MOVABS 105 230
DRWREL 10 20
CIRCLE 50
DRWREL -10 20
CIRCLE 20
DEBUG>             After listing, returns to debugger
```

The /V flag enables or disables execution of the macro associated with button table entry 0. During normal execution, the macro indicated by button table entry 0 is executed every 1/30th of a second; when in the debugger, execution is automatically suspended. The command /V 1 restarts execution (every 1/30th second) of the macro indicated by button table entry 0. Note that, like all commands, the macro will not be executed unless you type /S. (/V 0 disables execution, if desired, once execution has been restarted.)

The /C command exits the local debugger and returns to normal operation of the Model One. Command execution continues from the point where it was interrupted.

Table 1 summarizes the local debugger commands.

Command	Use
/Step number	Step through number commands
/Macros	List all defined macros
/List nmac	List contents of macro number nmac
/V flag	flag=1 enable macro at button table entry 0; flag=0 disable macro at button table entry 0.
/Continue	Exit debugger and continue normal command execution
/Help	Provides information about debugger commands
All graphics commands	Normal execution of graphics commands directly

Table 1 Summary of Local Debugger Commands

2.0 Command Stream Translator

The Model One graphics command set includes the command DEBUG flag, which is used to turn the command stream translator on and off. The command stream translator should not be confused with the local debugger. The command stream translator disassembles the commands which are being executed by the Model One and displays them on the local alphanumeric terminal in mnemonic form.

The command stream translator will work with both DMA and serial interfaces; however, the Model One should be configured with XON/XOFF communications protocol enabled, to avoid host input buffer overflow problems.

Each command opcode is translated into the command name (0E (hex) becomes CIRCLE, for example); the parameters are converted to ASCII decimal.

DEBUG 1 starts the command stream translator; DEBUG 0 disables the command stream translator.

The command stream translator can be used from within an applications program to help diagnose problems. For example, if a particular section of code is causing problems, you can call the DEBUG command before entering that section of code:

```
    :
    :
    :
CALL PROC1          Call user procedure PROC1
CALL DEBUG (1)      Enable the command stream translator
CALL PROC2          Call user procedure PROC2 and
                    use the command stream translator to
                    follow it in detail
CALL DEBUG (0)      Disable the command stream translator
CALL PROC3          Call user procedure PROC3
    :
    :
```

Example 1 Using the DEBUG Command

You can also use the command stream translator in conjunction with the local debugger (see section 1). For example, you can halt normal command execution with the [CTRL-X] entry into the local debugger, type DEBUG 1 to turn on the command stream translator, and inspect the commands being executed by using the /S command of the local debugger to step through the commands. Alternatively, you can /Continue. The next example shows what such a command sequence would look like:

```
CTRL-X              Enter local debugger with [CTRL-X]
DEBUG> DEBUG 1      Enable command stream translator
DEBUG> /S 5         Execute 5 commands
MOVABS 0,0          Display disassembled command stream
```

```
DRWABS 50,100
CIRCLE 20
DRWABS 10,10
DRWABS 10,20
DEBUG> DEBUG 0        Disable command stream translator
DEBUG> /C             Exit local debugger
```

Example 2 Using the Local Debugger and the DEBUG Command

3.0 Instant Replay

The Model One command REPLAY displays the last 32 characters which were sent over the host serial port to the Model One. The characters are displayed in ASCII hexadecimal form. For example, the command REPLAY executed from the local alphanumeric terminal outputs an array of ASCII hex characters:

```
00 FF FD F0 A0 AA AB BF
FF F0 FC 00 00 00 00 00
F0 FF 3F 3C 80 89 8A 76
7F FF FF FF 30 3F 55 F5
```

The command stream starts at the upper-left corner, goes across the top row, and continues until it reaches the lower-right corner.

The REPLAY command may also be useful in debugging the HOSTSIO interface to identify stray characters from a modem or bad transmission line.

RASTER TECHNOLOGIES
MODEL ONE/80
PIXEL MOVER

Revision 1.0 February 27, 1984

MODEL ONE PIXEL MOVER
February 27, 1984

NOTICE:

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

RASTER TECHNOLOGIES DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS MATERIAL (INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE), EITHER EXPRESS OR IMPLIED. RASTER TECHNOLOGIES SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM ANY ERROR CONTAINED HEREIN, INCLUDING, BUT NOT LIMITED TO, FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS MATERIAL.

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 regard 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

Figures

PIXEL MOVER

The Option Card pixel mover allows a window of pixel data to be moved between banks of image memory and to a different location within any bank. The pixel mover uses coordinate registers 11, 12, 13, and 14 to specify the location of the pixel data which is to be moved. You must set the CREGs which control the pixel mover before the pixel move is initiated.

1.0 Source and Destination Windows

The pixel mover moves data from a rectangular source window to a destination window of the same size. Coordinate register 11 contains the location of one corner of the source window; CREG 12 contains the diagonal corner of the window.

The pixel mover begins scanning the source window by moving the pixel at the location specified by CREG 11 first. Subsequent pixels in the window are moved in rows, proceeding towards CREG 12. Thus, the selection of the corners of the source window, as placed in CREGs 11 and 12, give you control over the order in which pixels are moved.

The destination window is specified by CREGs 13 and 14. CREG 13 contains the location of one corner of the destination window. 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 as shown in Figure 1.

After loading CREGs 11 through 14, the PIXMOV command is executed to perform the actual move.

For example:

To move a 20 by 50 pixel array defined by the corner points (-100,100) and (-81,51) to the rectangle defined by the corner points (100,100) and (119,51), this command sequence would be used:

```
CLOAD 11 -100,100     Load source corner
CLOAD 12 -81,51       Load diagonal source corner
CLOAD 13 100,100      Load destination corner
CLOAD 14 119,51       Load destination scanning direction
PIXMOV                Perform the actual pixel move
```

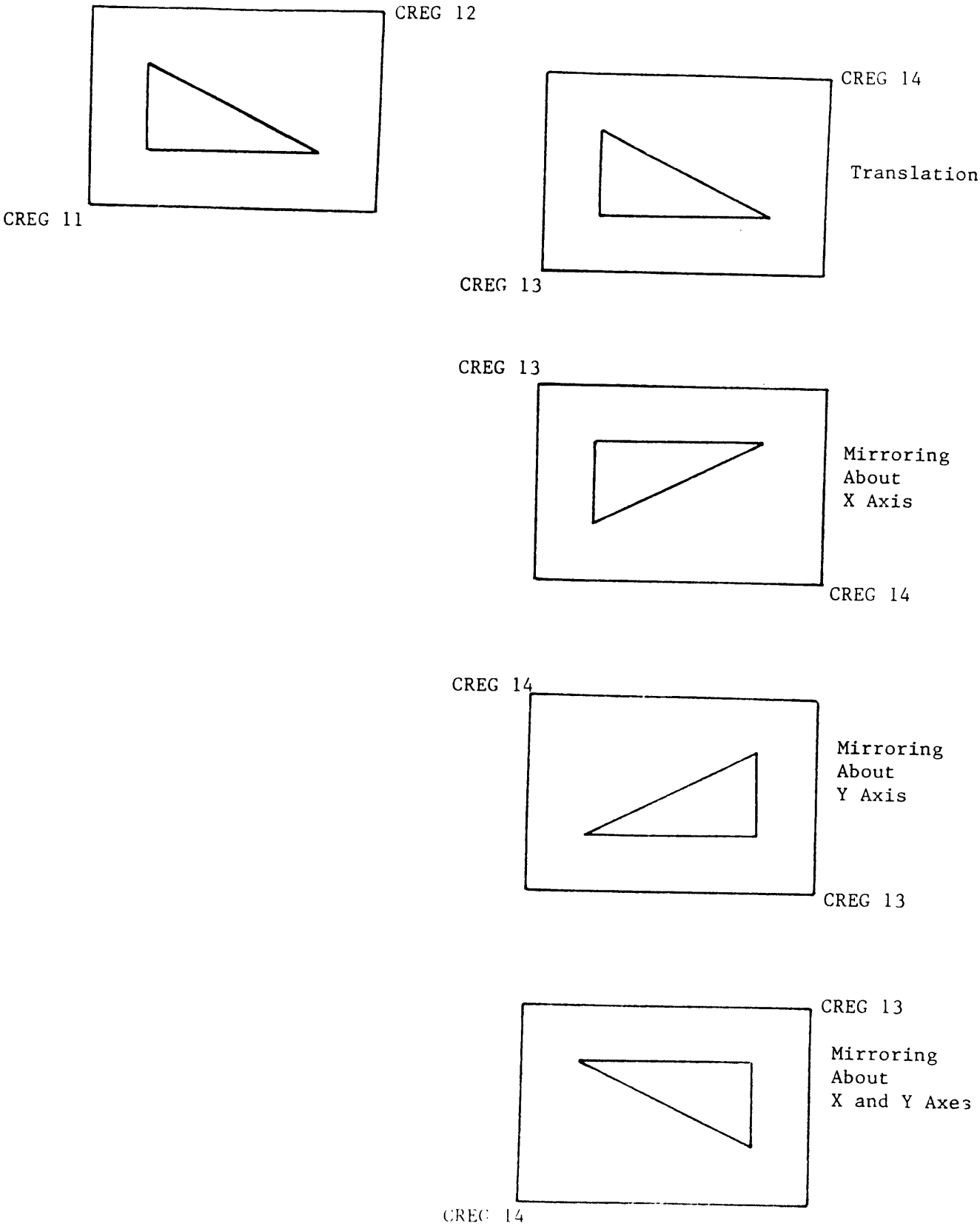


Figure 7.1 Settings of CREGs 11 through 14 Allow Mirroring

2.0 The Pixel Processor (PIXFUN) and the Pixel Mover

The PIXFUN command controls the pixel processor mode--whether new data is inserted, ANDed, ORed, and so on—during a pixel move.

Thus, you can use the pixel mover to pick up a window--perhaps containing a commonly-used area of the image—and transfer it to any desired area of the screen. This image section can then be inserted, ADDed, XORed, and so on. XOR can be used to drag the window until the correct positioning is determined, when the pixel function would be switched to INSERT.

Raster Technologies

Model One

ERROR MESSAGES REFERENCE GUIDE

Revision 1.0

June 20, 1985

Part # 552-00033-001

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

RASTER TECHNOLOGIES
MODEL ONE

NOTICE:

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

RASTER TECHNOLOGIES DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS MATERIAL (INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE), EITHER EXPRESS OR IMPLIED. RASTER TECHNOLOGIES SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM ANY ERROR CONTAINED HEREIN, INCLUDING, BUT NOT LIMITED TO, FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS MATERIAL.

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.

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	-63 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 15	ZOOM
Frames	0 to 255	BLINKR, WAIT
Index (button)	-63 to 63	BUTTBL, BUTTON
Index (LUT)	0 to 255 (Model One/10 allows 0 to 1024 for LUT16)	BLINKD, BLINKE, LUT8, LUT16, LUTA, LUTB, LUTG, LUTR, LUTRMP,
Look-up table index	0 to 255 (Model One/10 allows 0 to 1024 for LUT16)	BLINKD, BLINKE, LUT8, LUT16, LUTA, LUTB, LUTG, LUTR, LUTRMP,
Look-up table	1, 2, 4, 7	BLINKD, BLINKE
Lut (look-up table value)	1, 2, 4, 7	BLINKD, BLINKE
Macnum	0 to 255	BUTTBL, MACERA
Macro number	0 to 255	BUTTBL, MACERA
ncols	1 to 1280	READW, READWE, RUNLEN, RUNLN8
Nrows	1 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 15	ZOOM