Ω500 Display Controller
Reference Manual

(Order Number 004-02207-00)

April 1, 1984
DISCLAIMER

The information in this manual is subject to change without notice.

Metheus Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Metheus Corporation assumes no responsibility for any errors that may appear in this manual. Metheus Corporation makes no commitment to update nor to keep current the information contained in this document.

Metheus Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Metheus product. No other circuit patent licenses are implied.

Metheus software products are copyrighted by and shall remain the property of Metheus Corporation. Use, duplication, or disclosure is subject to restrictions stated in your Metheus software license.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Metheus Corporation.

METHEUS, AXIA, FLASH-fill, and PIXBLT are trademarks of Metheus Corporation.

April 1, 1984
<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Revision History</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>-00</td>
<td>First Issue. Describes the Q500 Display Controller Instructions for version 2.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4/84</td>
<td></td>
</tr>
<tr>
<td></td>
<td>microcode</td>
<td></td>
</tr>
</tbody>
</table>

This equipment complies with the requirements in part 15 of FCC rules for a class A computing device. Operation of this equipment in a residential area may cause unacceptable interference to radio and TV reception requiring the operator to take whatever steps are necessary to correct interference.

April 1, 1984

iii
PREFACE

This manual describes the instruction set for the Ω500 display controller. Primarily, this is a reference manual and requires a high degree of experience with interactive graphics to understand. The first two chapters, however, present some basic concepts about graphics systems in general and the Ω500 in particular.

The reference section, Chapters 5 through 11, divides the instructions into seven functional groups. Within each group, instructions are listed alphabetically. Chapter 4 introduces and describes the organization of this reference section.

Chapters at a Glance

Chapter 1 - Introduction to the Ω500
Briefly discusses some basic concepts about interactive graphics systems.

Chapter 2 - Ω500 Graphics
Provides system-specific details like addressing and memory size. Introduces several instructions and their function. Contains a list of all instructions and the chapter that describes them.

Chapter 3 - Installation Instructions
Provides step-by-step instructions for installing and using the GRAFIN board, and for cabling the Ω500 to the monitor and the host computer.

Chapter 4 - Using the Instruction Reference Chapters
Introduces and explains how to effectively use the chapters that describe each group of instructions.

Chapters 5 through 11
Describes seven groups of instructions that control the Ω500 Display Controller. Instructions in each group are alphabetically arranged by name.

Appendix A
Lists the conversions between ASCII, decimal, octal, and hexadecimal.

April 1, 1984
CONTENTS

CHAPTER 1
Introduction to the Ω500
The Ω510 ................................................................. 1-2
The Ω530 ................................................................. 1-2
Speed ................................................................. 1-2
Reliability ............................................................. 1-2
Software and Hardware Support .................................. 1-3

CHAPTER 2
Ω500 Graphics
Display Units .......................................................... 2-1
Coordinate Addresses ............................................... 2-2
Lines, Rectangles, Arcs, and Polygons ......................... 2-3
Pointers ............................................................... 2-4
Pan ................................................................. 2-5
Raster-Ops .......................................................... 2-5
Ω530 Folded Mode .................................................. 2-6

CHAPTER 3
Installation Instructions
Selecting GRAFIN Data Rates .................................... 3-1
Installing the Interface ............................................ 3-6
Replacing the Fuse ................................................ 3-6
Connecting to the Monitor ....................................... 3-7
Connecting to the Computer .................................... 3-7
Connecting to Power ............................................. 3-7

CHAPTER 4
Using the Instruction Reference Chapters
Group Descriptions ................................................ 4-1

April 1, 1984
CHAPTER 5
Display-Pointer-Move Instructions
MOVP1 .................................................. 5-2
MOVP2 .................................................. 5-4
POLYC .................................................. 5-6
POLYM .................................................. 5-9
POLYS .................................................. 5-12
POLYV .................................................. 5-14
RM0VP1 ................................................. 5-17
RM0VP2 ................................................. 5-20

CHAPTER 6
Drawing Instructions
AFILL1 ............................................... 6-2
AFILL2 ............................................... 6-4
AFILL3 ............................................... 6-6
ARC .................................................. 6-8
CHAR ................................................ 6-10
CLEAR ............................................. 6-12
COMPDRA .......................................... 6-14
DRAW ............................................... 6-16
FFILL ............................................... 6-18
POLYF ............................................... 6-20
POLYO ............................................... 6-23
RECT1 ............................................... 6-25
RECT2 ............................................... 6-27
RLFILL ............................................... 6-29
XDRAW ............................................... 6-31

CHAPTER 7
Drawing-Control Instructions
CSPACE ........................................... 7-2
FSIZE ............................................... 7-6
LDROC ............................................ 7-7
PATTERN .......................................... 7-11
RASTOP ........................................... 7-15
RDMASK ........................................... 7-20
SET COLOR ......................................... 7-22
SETCORN .......................................... 7-25
SETCSZ ............................................ 7-27
WRMASK ........................................... 7-28

viii

April 1, 1984
CHAPTER 8
Display-Control Instructions
BLANK................................................. 8-2
BLINK.................................................. 8-4
CMAP................................................... 8-6
CURS................................................... 8-8
PPAN.................................................. 8-10
SZCUR................................................ 8-12

CHAPTER 9
Data-Transfer Instructions
GRAFIN.................................................. 9-2
PIXBLT.................................................. 9-7
RDR..................................................... 9-9
RPixel.................................................. 9-11
WPIXEL............................................... 9-13
WRR..................................................... 9-15

CHAPTER 10
Utility Instructions
DFUALT............................................... 10-2
INIT.................................................. 10-4
INQ.................................................. 10-5
LDFONT............................................... 10-7
LDPAT............................................... 10-9
READ CONF........................................ 10-11
SIG READ.......................................... 10-12
SYNCH............................................... 10-13

CHAPTER 11
Folded-Mode Instructions
BANK................................................ 11-2
OVMAP............................................... 11-4
SELRES............................................. 11-7
XBARLD............................................. 11-9

APPENDIX A

April 1, 1984
<table>
<thead>
<tr>
<th>NUMBER</th>
<th>FIGURE TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>The Q500 Display Controller and Monitor</td>
<td>1-1</td>
</tr>
<tr>
<td>2-1</td>
<td>Creating Images With Pixel Combinations</td>
<td>2-2</td>
</tr>
<tr>
<td>2-2</td>
<td>Coordinate Addresses</td>
<td>2-3</td>
</tr>
<tr>
<td>3-1</td>
<td>Removing the Top Cover</td>
<td>3-2</td>
</tr>
<tr>
<td>3-2</td>
<td>Removing the Host and GRAFIN Interface Boards</td>
<td>3-3</td>
</tr>
<tr>
<td>3-3</td>
<td>Locating the Data Transfer Switch on the GRAFIN Board</td>
<td>3-4</td>
</tr>
<tr>
<td>3-4</td>
<td>Connecting the Monitor to the Back Panel</td>
<td>3-7</td>
</tr>
<tr>
<td>5-1</td>
<td>Moving the P1 Pointer</td>
<td>5-3</td>
</tr>
<tr>
<td>5-2</td>
<td>Moving the P2 Pointer</td>
<td>5-5</td>
</tr>
<tr>
<td>5-3</td>
<td>Nested Polygons Created with POLYV</td>
<td>5-8</td>
</tr>
<tr>
<td>5-4</td>
<td>Nested Polygons Created with POLYV and POLYM</td>
<td>5-8</td>
</tr>
<tr>
<td>5-5</td>
<td>Creating Two Open Polygons Using POLYM</td>
<td>5-11</td>
</tr>
<tr>
<td>5-6</td>
<td>Adding Vertices with POLYV</td>
<td>5-16</td>
</tr>
<tr>
<td>5-7</td>
<td>Moving P1 to a Relative Location</td>
<td>5-19</td>
</tr>
<tr>
<td>5-8</td>
<td>Moving P2 to a Relative Location</td>
<td>5-22</td>
</tr>
<tr>
<td>6-1</td>
<td>P1 Set to a Different Color than the Inner Boundary</td>
<td>6-3</td>
</tr>
<tr>
<td>6-2</td>
<td>P1 Set Between Boundaries of Two Different Colors</td>
<td>6-3</td>
</tr>
<tr>
<td>6-3</td>
<td>Filling to the Outer Boundary with AFILL2</td>
<td>6-5</td>
</tr>
<tr>
<td>6-4</td>
<td>Drawing an Arc</td>
<td>6-9</td>
</tr>
<tr>
<td>6-5</td>
<td>Typing Characters on the Screen</td>
<td>6-11</td>
</tr>
<tr>
<td>6-6</td>
<td>Drawing a Line in the Complement Color</td>
<td>6-15</td>
</tr>
<tr>
<td>6-7</td>
<td>Drawing a Line</td>
<td>6-17</td>
</tr>
<tr>
<td>6-8</td>
<td>FLASH-filling a Rectangle</td>
<td>6-19</td>
</tr>
<tr>
<td>6-9</td>
<td>Filling a Polygon</td>
<td>6-22</td>
</tr>
<tr>
<td>6-10</td>
<td>Outlining a Polygon</td>
<td>6-24</td>
</tr>
<tr>
<td>6-11</td>
<td>Drawing a Rectangle Outline</td>
<td>6-26</td>
</tr>
<tr>
<td>6-12</td>
<td>Filling a Rectangle</td>
<td>6-28</td>
</tr>
<tr>
<td>7-1</td>
<td>Loading the Raster-Op Comparator Array</td>
<td>7-9</td>
</tr>
<tr>
<td>7-2</td>
<td>Default Drawing and Filling Patterns</td>
<td>7-14</td>
</tr>
<tr>
<td>7-3</td>
<td>The SET COLOR Index to the Default Color Map</td>
<td>7-24</td>
</tr>
<tr>
<td>7-4</td>
<td>Selecting the Orientation of Text</td>
<td>7-26</td>
</tr>
<tr>
<td>8-1</td>
<td>Displaying the Cursor</td>
<td>8-9</td>
</tr>
<tr>
<td>8-2</td>
<td>Panning to a New Display-Memory Origin</td>
<td>8-11</td>
</tr>
</tbody>
</table>

April 1, 1984
### TABLES

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>TABLE TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>0500 Instructions Listed Alphabetically</td>
<td>2-7</td>
</tr>
<tr>
<td>3-1</td>
<td>Data Rate Selection</td>
<td>3-5</td>
</tr>
<tr>
<td>7-1</td>
<td>Drawing Modes</td>
<td>7-12</td>
</tr>
<tr>
<td>7-2</td>
<td>Data Flow Control</td>
<td>7-16</td>
</tr>
<tr>
<td>7-3</td>
<td>Op1 and Op2 Assignments</td>
<td>7-17</td>
</tr>
<tr>
<td>7-4</td>
<td>ALU Functions</td>
<td>7-17</td>
</tr>
<tr>
<td>7-5</td>
<td>Drawing Raster-Ops</td>
<td>7-19</td>
</tr>
<tr>
<td>7-6</td>
<td>PIXBLT Raster-Ops</td>
<td>7-19</td>
</tr>
<tr>
<td>7-7</td>
<td>Meaning of Bits in the Orientation Byte</td>
<td>7-25</td>
</tr>
<tr>
<td>9-1</td>
<td>0500 Status Message Format</td>
<td>9-3</td>
</tr>
<tr>
<td>10-1</td>
<td>INQ Parameters</td>
<td>10-6</td>
</tr>
<tr>
<td>A-1</td>
<td>Code Conversion Chart</td>
<td>A-1</td>
</tr>
</tbody>
</table>

April 1, 1984
CHAPTER 1

INTRODUCTION TO THE Ω500

Figure 1-1. The Ω500 Display Controller and Monitor

The Ω500 Display Controller, shown in Figure 1-1, is a high-resolution, high-performance color graphics device. It translates instructions from a host computer system into color and timing signals for a display monitor.

Two models of the Ω500 are available, the Ω510 and the Ω530. Each can be purchased as a graphics subsystem which includes the display controller and a high-resolution color monitor. Both models scan at 60 Hz scanning rate to ensure a flicker-free display. The difference between the two models is the amount of memory resolution they provide.
Introduction

The 0510

The 0510 Display Controller’s graphics memory provides a resolution of 1280 x 1024 points which, when coupled with a monitor capable of this same high resolution, provides a highly-detailed display.

The 0510 memory is organized into eight bit-planes. This organization places 256 colors in the display color map at one time. These colors are dynamically selectable from a palette of 16.7 million color choices.

The 0530

The 0530 Display Controller’s graphics memory can be switched between high and low resolution in software. High-resolution applications operate in the normal 1280 x 1024 configuration. For applications that require more color depth and lower resolution, the 0530 can be operated in folded mode. Folded mode provides 32 bit-planes, each with 640 x 512 resolution.

Speed

The 0500 writes four to ten times faster than other systems because it uses a bipolar bit-slice processor. Faster display speed means an easier and friendlier system to work with. No more long waits for graphics changes to appear.

Reliability

All display electronics in the 0500 are located on a single printed-circuit board. This increases system reliability by eliminating multiple edge connectors. Built-in self-test features and a unique on-board signature analyzer also improve system reliability and serviceability.
Software and Hardware Support

The AXIA Graphics Package option, available for the host computer, provides software that supports the Ω500 Systems. The AXIA Graphics Package is described in a separate manual.

Special instructions support graphics input devices, such as graphics tablets and bit-pads. These devices simply plug into the serial connector on the Ω500 back panel.
CHAPTER 2

Ω500 GRAPHICS

This chapter briefly introduces the terms used in this manual and the instructions found in the reference section, Chapters 5 through 11. Table 2-1, at the end of this chapter, lists all the instructions with the chapter number and title that describes them. This chapter describes several elements of computer graphics, listed below, that are specific to the Ω500 Display Controllers.

- Display units
- Coordinate addresses
- Lines, rectangles, arcs, and polygons
- Pointers
- Pan
- Raster-ops
- Ω530 folded mode

Display Units

In the Ω500 Systems, the pixel (picture element) is the basic element of graphic display. Pixels represent a single addressable point in graphic memory and a single displayable point on the monitor screen. Graphic memory resolution (the range of addressable pixels) is 1280 in the x axis by 1024 in the y axis; That is over one million addressable points. All display images are created by writing combinations of adjoining pixels. (See Figure 2-1.)

Any pixel can be displayed in any color available. The Ω500 uses eight bit-planes to provide 256 color-map addresses that can be chosen from the 16.7 million colors in the palette. Any one of those 256 colors can then be used as the current drawing color.
Figure 2-1. Creating Images With Pixel Combinations

Coordinate Addresses

The Q500 devices describe locations on the screen with coordinate addresses. Each point, or pixel, in graphic memory is represented by a unique coordinate address. Pixel addresses consist of an x value and a y value representing the point where the x and y axes meet.

Figure 2-2 shows these axes and their ranges in decimal. Normally, addresses range from 0 through 1279 (decimal) in the x axis and 0 through 1023 in the y axis. In folded mode, the Q530 coordinates range from 0 through 639 for x and 0 through 511 for y.
In the coordinate system, the upper left corner of the screen and in graphic memory represents (0x,0y). The lower right corner represents (1279x,1023y). Therefore, x address values increase from left to right; y address values increase from the top to bottom. Memory and screen coordinates are the same.

Although two 8-bit bytes are transmitted between the host and the display controller for each coordinate address, only 11 bits represent each x coordinate value and 10 represent each y coordinate value. A complete address specification consists of four 8-bit bytes: low-x, high-x, low-y, and high-y. The reference chapters in this manual abbreviate these address specifications to lox, hix, loy, and hiy and explain which bits are significant.

**Lines, Rectangles, Arcs, and Polygons**

Pixels combine in many ways to create graphic images. The basic display elements are the line (or vector), the rectangle, the arc, and the polygon.
A line, or vector, is a sequence of adjoining pixels extended between two points on the display. A line can be solid or dashed. A solid line contains pixels all written in the same color. A dashed line contains pixels written in the current drawing color interspersed with spaces.

A rectangle is any four-sided figure with 90-degree angles, including a square, where the sides are parallel to the x and y axes. Only two corner points are required to define a rectangle. Rectangles can be outlined, solidly filled, pattern-filled, or filled then outlined in another color. A number of drawing instructions, described in Chapter 6, create the various rectangle styles.

An arc is a curved line. Like any line, it can be solid or dashed. Continuing an arc until the end meets the starting point creates a circle.

A polygon is any multi-sided figure. Polygons are created by defining their vertex points using two drawing instructions, described in Chapter 6, called POLYY and POLYM. Like rectangles, polygons can be outlined and filled using two more drawing instructions, POLYO and POLYF.

Pointers

Two pointers, P1 and P2, mark the positions of figures in graphic memory and on the screen, but they are not displayed. Pointers mark, for example, the center point of an arc or the corners of a rectangle. The Display-Pointer-Move Instructions, described in Chapter 5, position these pointers either at absolute locations or at relative locations.
Pan

The PPAN instruction, described in Chapter 8, relocates the display on the screen vertically or horizontally, panning like a camera. Images pan in fixed blocks measuring 40 pixels horizontally and 16 pixels vertically. When the 0530 is operated in folded mode, images pan 20 pixels horizontally.

Raster-Ops

Raster operation instructions (see RASTOP in Chapter 7) conditionally modify pixels. Raster-ops affect lines and blocks of pixels logically or arithmetically. For example, a logical raster-op can exclusively OR (XOR) the pixels in a rectangle to complement their color. An arithmetic raster-op can add color values to pixels as they are transferred from their source to their destination. Chapter 9 contains more information on the PIXBLIT instruction (Pixel Block Transfer).

The RASTOP instruction specifies the type of raster operation (logical or arithmetical) and enables the Raster-Op Comparator (ROC). The ROC is an array of bits that conditionally prevent or allow data transfers for that pixel location. The LDR0C instruction, in Chapter 7, loads values into the ROC.

0530 Folded Mode

The 0530 Display Controller has two screen resolutions: high resolution (1280 * 1024 * 8) and folded mode (640 * 512 * 32). When high resolution is selected, the 8 bit-planes allow 256 colors. In folded mode, the memory is reconfigured to provide 32 bit-planes. These 32 planes are divided into four banks of 8 planes each: bank0, bank1, bank2, and bank3.

These four banks can be individually routed to any of the color lookup tables (the palette). For instance, routing one bank to the red lookup table, one bank to the green, and another bank to the blue creates a 24-bit-deep image. Another way to achieve this image is by storing four sets of 8-bit images in the banks (one image per bank). To display a particular image,
route the appropriate bank to all three color lookup tables.

Additionally, plane 0 of both bank2 and bank3 can be used as overlay planes. When the overlay lookup-table is enabled, bits set in an overlay plane take priority over colors in the memory banks. Either or both of the two overlay planes may be enabled.

In folded mode, operations such as setting the current drawing color or write mask, take place in a selected bank of 8 planes. Other banks can also perform these operations independently. Each bank remembers its last drawing color and write mask. Selecting a new bank restores its previous color and write mask.

A jumper on the display controller board selects the default screen resolution, normal or folded. However, screen resolution can dynamically change using (Since the horizontal scanning frequency changes between 65.7 KHz and 33KHz, the monitor may need adjustment. Few monitors can run at both frequencies.)

This chapter introduced several instructions that perform graphic operations. Many more exist. Table 2-1 alphabetically lists all the instructions that control the Ω500 systems and the chapter in this manual that describes them.
<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>CHAPTER TITLE</th>
<th>CHAPTER NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFILL1</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>AFILL2</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>AFILL3</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>ARC</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>BANK</td>
<td>Folded-Mode Instructions</td>
<td>11</td>
</tr>
<tr>
<td>BLANK</td>
<td>Display-Control Instructions</td>
<td>8</td>
</tr>
<tr>
<td>BLINK</td>
<td>Display-Control Instructions</td>
<td>8</td>
</tr>
<tr>
<td>CHAR</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>CLEAR</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>CMAP</td>
<td>Display-Control Instructions</td>
<td>8</td>
</tr>
<tr>
<td>CMPDR</td>
<td>Drawing Instructions</td>
<td>7</td>
</tr>
<tr>
<td>CSPACE</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>CURS</td>
<td>Display-Control Instructions</td>
<td>8</td>
</tr>
<tr>
<td>DRAW</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>DFAULT</td>
<td>Utility Instructions</td>
<td>10</td>
</tr>
<tr>
<td>FFILL</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>FSIZE</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>GRAFIN</td>
<td>Data-Transfer Instruction</td>
<td>9</td>
</tr>
<tr>
<td>INIT</td>
<td>Utility Instructions</td>
<td>10</td>
</tr>
<tr>
<td>INQ</td>
<td>Utility Instructions</td>
<td>10</td>
</tr>
<tr>
<td>LDFONT</td>
<td>Utility Instructions</td>
<td>10</td>
</tr>
<tr>
<td>LD PAT</td>
<td>Utility Instructions</td>
<td>10</td>
</tr>
<tr>
<td>LDROC</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>MOVPI</td>
<td>Display-Pointer-Move Instructions</td>
<td>5</td>
</tr>
<tr>
<td>MOVPP2</td>
<td>Display-Pointer-Move Instructions</td>
<td>5</td>
</tr>
<tr>
<td>OVMAP</td>
<td>Folded-Mode Instructions</td>
<td>11</td>
</tr>
<tr>
<td>PATTERN</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>PIXBLT</td>
<td>Data-Transfer Instruction</td>
<td>9</td>
</tr>
<tr>
<td>POLYC</td>
<td>Display-Pointer-Move Instructions</td>
<td>5</td>
</tr>
<tr>
<td>POLYM</td>
<td>Display-Pointer-Move Instructions</td>
<td>5</td>
</tr>
<tr>
<td>POLYF</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>POLYO</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>POLYS</td>
<td>Display-Pointer-Move Instructions</td>
<td>5</td>
</tr>
<tr>
<td>POLYV</td>
<td>Display-Pointer-Move Instructions</td>
<td>5</td>
</tr>
<tr>
<td>PPAN</td>
<td>Display-Control Instructions</td>
<td>8</td>
</tr>
</tbody>
</table>
Table 2-1. Ω500 Instructions Listed Alphabetically

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>CHAPTER TITLE</th>
<th>CHAPTER NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>RASTOP</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>RDMASK</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>READ CONF</td>
<td>Utility Instructions</td>
<td>10</td>
</tr>
<tr>
<td>RECT1</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>RECT2</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>RDR</td>
<td>Data-Transfer Instructions</td>
<td>9</td>
</tr>
<tr>
<td>RLFILL</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
<tr>
<td>RMOV1</td>
<td>Display-Pointer-Move Instructions</td>
<td>5</td>
</tr>
<tr>
<td>RMOV2</td>
<td>Display-Pointer-Move Instructions</td>
<td>5</td>
</tr>
<tr>
<td>RPIXEL</td>
<td>Data-Transfer Instruction</td>
<td>9</td>
</tr>
<tr>
<td>SELRES</td>
<td>Folded-Mode Instructions</td>
<td>11</td>
</tr>
<tr>
<td>SET COLOR</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>SETCORN</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>SETCSZ</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>SIG READ</td>
<td>Utility Instructions</td>
<td>10</td>
</tr>
<tr>
<td>SYNCH</td>
<td>Utility Instructions</td>
<td>10</td>
</tr>
<tr>
<td>SZCUR</td>
<td>Display-Control Instructions</td>
<td>8</td>
</tr>
<tr>
<td>WPIXEL</td>
<td>Data-Transfer Instructions</td>
<td>9</td>
</tr>
<tr>
<td>WRMASK</td>
<td>Drawing-Control Instructions</td>
<td>7</td>
</tr>
<tr>
<td>WRR</td>
<td>Data-Transfer Instructions</td>
<td>9</td>
</tr>
<tr>
<td>XBARLD</td>
<td>Folded-Mode Instructions</td>
<td>11</td>
</tr>
<tr>
<td>XDRAW</td>
<td>Drawing Instructions</td>
<td>6</td>
</tr>
</tbody>
</table>
CHAPTER 3
INSTALLATION INSTRUCTIONS

After unpacking the Ω500, place it in a location that provides at least two inches of air space around the top and sides for cooling. Then, before operating the Ω500, read this chapter. It explains how to connect the Ω500 system to your hardware. Specifically, the instructions in this chapter describe:

- Selecting GRAFIN data rates
- Installing the interface
- Replacing the fuse
- Connecting to the monitor
- Connecting to the computer
- Connecting to power

CAUTION

Only qualified service personnel should attempt any procedure where the covers must be removed. Read and follow the installation instructions carefully. Failure to install the Ω500 properly could result in improper operation or equipment damage.

Selecting GRAFIN Data Rates

GRAFIN interface boards are set at the factory for 9600 baud. If you need to change this setting, follow these instructions.

1) Unplug the Ω500 from the AC power source.
Installation Instructions

2) Remove the top cover by removing the 10 screws that hold it in place. Figure 3-1 shows the location of these screws.

Figure 3-1. Removing the Top Cover
3) Remove the GRAFIN and host interface boards as shown in Figure 3-2.
Installation Instructions

4) Find the 8-position switch near the center of the GRAFIN board. Four positions, numbered 1 through 4 in Figure 3-3, select the transfer rates. This figure shows the default switch positions selecting 9600 baud.

5) Select the new baud rate from Table 3-1 and set the switches accordingly.

Figure 3-3. Locating the Data Transfer Switch on the GRAFIN Board
### Table 3-1. Data Rate Selection

<table>
<thead>
<tr>
<th>BAUD</th>
<th>SWITCH SETTINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>unused</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>19.2K</td>
<td>1 1 1 0</td>
</tr>
<tr>
<td>9600</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td>7200</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>4800</td>
<td>1 0 1 1</td>
</tr>
<tr>
<td>3600</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>2400</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>1800</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>1200</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>600</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>300</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>150</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>134.5</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>110</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>75</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>50</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

**NOTE:** 0=Off, 1=On

6) Switch 5 determines resolution format. When set in the 0 (off) position, the GTCO binary high resolution format is selected. The 1 (on) position selects the Summagraphics Bit-Pad format. (Refer to the tablet manual for compatibility information.)

April 1, 1964
Installation Instructions

7) After changing switch positions, replace boards, the top cover, and the power cord.

Installing the Interface

In most cases, the interface is installed at the factory. If it is necessary to install an interface after receiving the 0500, refer to the Operator's Manual for that interface for installation information.

Replacing the Fuse

The line fuse inserts into the back panel of the 0500. Replace the fuse by first disconnecting power, then removing the fuse cover. Remove the fuse and replace it with one of the same value. Figure 3-4 shows the location of the fuse.

WARNING

Never change fuses while the 0500 is connected to the power source. Always disconnect the power cord first to prevent equipment damage and personal injury.

CAUTION

To avoid possible damage to the equipment, always replace the original fuse with an equivalent type.
Connecting to the Monitor

Three separate cables connect the 0500 to the display monitor. Figure 3-4 shows these connections. One cable connects to each of the three BNC connectors that correspond to the three primary colors: red, green, and blue. These connections are labeled R, G, and B on the back panel.

If your application requires external horizontal and vertical synch, connect two more cables to these BNC connectors provided on the back panel.

---

**Figure 3-4. Connecting the Monitor to the Back Panel**

---

**IMPORTANT**

To comply with FCC Class A operation requirements, fully-shielded cables must be used.

April 1, 1984
Installation Instructions

Connecting to the Computer

Refer to the Operator's Manual for the Ω500 interface you are using for connection information.

Connecting to Power

After all installation procedures have been followed, connect a power cord to the Ω500 back-panel connector, then connect to an appropriate power source.

The Ω500 is ready to operate.
CHAPTER 4
USING THE INSTRUCTION REFERENCE CHAPTERS

Chapters 5 through 11 describe the instructions that control the graphics functions of the Q500 system. These instructions are divided into seven groups, listed below:

- Display-Pointer-Move Instructions
- Drawing Instructions
- Drawing-Control Instructions
- Display-Control Instructions
- Data-Transfer Instructions
- Utility Instructions
- Folded-Mode Instructions

To make instructions easy to find and understand, each one appears in a standard format. Also, within each group, instructions appear in alphabetical order by instruction name. Table 2-1 contains all the instruction names in order with the number and title of the chapter where they appear.

These chapters use examples written in both hexadecimal and decimal. To distinguish between them, hexadecimal numbers are followed with the letter h, and decimal numbers are followed with the letter t.

Group Descriptions

The first group describes the instructions that position pointers. Positioning the display pointers is basic to many graphics actions, such as drawing lines, rectangles, arcs, or polygons. The Display Pointer Move instructions are described in Section 4.

The next group of instructions use the pointers to create images on the monitor screen. Drawing Instructions draw vectors, draw rectangles, and fill objects.

The next two groups of instructions globally control the way
Using the Instruction Reference Chapters

figures are drawn and displayed. Drawing-Control Instructions set the environmental parameters for the Drawing Instructions. For example, they select the drawing colors and patterns, and the size and orientation of characters. Display-Control Instructions define the colors for the color map and select the Pan origin.

Data-Transfer Instructions transfer graphics data between the Omega and the host system, as well as from one display position to another.

The Utility Instructions provide a number of non-graphic functions. They report the status of the system and control special functions like SYNCH (used in animation, for instance), and the on-board signature analyzer used for troubleshooting.

The Omega can operate in two resolution modes: normal or folded. The last chapter describes the instructions the apply only to the Omega operating in folded mode.
CHAPTER 5

DISPLAY-POINTER-MOVE INSTRUCTIONS

The instructions in this chapter control the position of the display pointers. Pointers define the coordinate position of lines, polygons, and text.

- MOVP1 - Moves pointer P1 to an absolute coordinate
- MOVP2 - Moves pointer P2 to an absolute coordinate
- POLYC - Closes a polygon definition
- POLYM - Moves pointer P1 to a polygon vertex
- POLYS - Starts a polygon definition
- POLYV - Adds a vertex to a polygon
- RM0VP1 - Moves pointer P1 to a relative coordinate
- RM0VP2 - Moves pointer P2 to a relative coordinate
Display Pointer-Move Instructions

**MOVPI**

Moves pointer P1 to an absolute coordinate

**Instruction Format**

**HEX:** 52 lox hix loy hiy

**DECIMAL:** 82 x y

**ASCII:** R lox hix loy hiy

**Input Arguments**

- **lox:** the low-order eight bits of the eleven bits required to define the x coordinate value
- **hix:** only the three low-order bits of this byte are used; these three bits set the three high-order bits of the eleven bits required to define the x coordinate value
- **loy:** the low-order eight bits of the ten bits required to define the y coordinate value
- **hiy:** only the two low-order bits of this byte are used; these two bits set the two high-order bits of the ten bits required to define the y coordinate value

**Outputs**

None

**Description**

The MOVPI instruction moves P1 to a point specified in absolute coordinate values. The four bytes immediately following the opcode define the new location for P1.

5-2 April 1, 1984
Display Pointer-Move Instructions

Range

The allowable range for the x coordinate data is 0 through 1279t, and for the y coordinate data is 0 through 1023t.

Special Considerations

None

Example

This example, listed in hexadecimal, moves P1 to decimal coordinates 100, 200. Although you cannot see the position of P1 unless you issue a Drawing Instruction, Figure 5-1 shows where P1 would be.

52 64 00 C8 00

Figure 5-1. Moving the P1 Pointer
Display-Pointer-Move Instructions

MOVP2

Moves pointer P2 to an absolute coordinate

Instruction Format

HEX: 53 lox hix loy hiy

DECIMAL: 83 x y

ASCII: S lox hix loy hiy

Input Arguments

lox   the low-order eight bits of the eleven bits required to define the x coordinate value
hix   only the three low-order bits of this byte are used; these three bits set the three high-order bits of the eleven bits required to define the x coordinate value
loy   the low-order eight bits of the ten bits required to define the y coordinate value
hiy   only the two low-order bits of this byte are used; these two bits set the two high-order bits of the ten bits required to define the y coordinate value

Outputs

None

Description

The MOVP2 instruction moves P2 to a point specified in absolute coordinate values. The four bytes immediately following the opcode define the new location for P2.

5-4        April 1, 1984
Display-Pointer-Move Instructions

Range

The allowable range for the x coordinate data is 0 through 1279, and for the y coordinate data is 0 through 1023.

Special Considerations

None

Example

This example, listed in hexadecimal, moves P2 to absolute decimal coordinates 200, 300. Although you cannot see the position of P2 unless you issue a Drawing Instruction, Figure 5-2 shows where P2 would be.

53 C8 00 2C 01

Figure 5-2. Moving the P2 Pointer
Display-Pointer-Move Instructions

POLYC

Closes a polygon definition

Instruction Format

HEX: 44
DECIMAL: 68
ASCII: D

Input Arguments

None

Outputs

None

Description

This instruction closes, or ends, a polygon definition opened with the POLYS instruction. It also serves as a delimiter used to concatenate polygons. The connecting primitive will be an automatically generated move to the following vertex group. This move will not be overwritten after the polygon fill.

Range

Does not apply

Special Considerations

Following the POLYC instruction with a POLYV instruction begins a new polygon definition without having to issue the POLYS instruction.
Use caution when following a POLYC command with POLYM. Moving the pointer with POLYM before the polygon definition is closed causes undefined results.

Example

This example shows two sets of instructions in hexadecimal. Both draw two rectangles, one inside the other. The first set of instructions opens the polygon definition, defines eight vertices, closes the definition, then outlines them. Figure 5-3 shows the results. The next set of instructions also draws two rectangles using the same coordinates, but moves the pointer to the fifth vertex rather than defines it. Figure 5-4 shows the open inner figure that results.

```
56  Start the polygon definition (POLYS)
57 64 00 64 00  Define V1 at 100,100 (POLYV)
57 C8 00 64 00  Define V2 at 200,100
57 C8 00 C8 00  Define V3 at 200,200
57 64 00 C8 00  Define V4 at 100,200
44  Close the polygon definition (POLYC)
57 96 00 96 00  Define V5 at 150,150 (POLYV)
57 AF 00 96 00  Define V6 at 175,150
57 AF 00 AF 00  Define V7 at 175,175
57 96 00 AF 00  Define V8 at 150,175
66  Outline the polygons (POLYO)

56  Start the polygon definition (POLYS)
57 64 00 64 00  Define V1 at 100,100 (POLYV)
57 C8 00 64 00  Define V2 at 200,100
57 C8 00 C8 00  Define V3 at 200,200
57 64 00 C8 00  Define V4 at 100,200
44  Close the polygon definition (POLYC)
45 96 00 96 00  Move to V5, 150,150 (POLYM)
57 AF 00 96 00  Define V6 at 175,150
57 AF 00 AF 00  Define V7 at 175,175
57 96 00 AF 00  Define V8 at 150,175
66  Outline the polygons (POLYO)
```
Display-Pointer-Move Instructions

Figure 5-3. Nested Polygons Created with POLYV

Figure 5-4. Nested Polygons Created with POLYV and POLYM

5-8
April 1, 1984
Display-Pointer-Move Instructions

**POLYM**

*Moves pointer P1 to a polygon vertex*

**Instruction Format**

**HEX:** 45 lox hix loy hiy

**DECIMAL:** 69 x y

**ASCII:** E lox hix loy hiy

---

**Input Arguments**

- **lox** the low-order eight bits of the eleven bits required to define the x coordinate
- **hix** only the three low-order bits are used; these three bits set the three high-order bits required to define the x coordinate
- **loy** the low-order eight bits of the ten bits required to define the y coordinate
- **hiy** only the two low-order bits are used; these two bits set the two high-order bits required to define the y coordinate

**Outputs**

None

---

**Description**

Adding a vertex to the polygon data structure with POLYM moves the pointer the last x,y point. The resulting vector is not drawn during a POLYO, but is filled during a POLYP. This primitive is useful for clipped polygons when clipped regions align with viewport boundaries.

April 1, 1984
Display-Pointer-Move Instructions

Range

The allowable range for coordinate data is 0 to 1279 for x and 0 to 1023 for y.

Special Considerations

Following a POLYC or POLYS with a POLYM instruction causes the pointer to move after the polygon definition is finished and can cause undefined results.

Example

This example, listed in hexadecimal, draws two polygons using the same coordinates. The first set of instructions, however, defines two vertices then two moves while the second set of instructions defines a move, then two vertices, then a move again. Figure 5-5 shows the differences between the polygons after outlining. In both cases, the rectangles will fill identically.

```
56  Start a polygon definition (POLYS)
57 64 00 64 00    Define V1 at 100,100 (POLYV)
57 C8 00 64 00    Define V2 at 200,100
45 C8 00 C8 00    Move pointer to V3, 200,200 (POLYM)
45 64 00 C8 00    Move pointer to V4, 100,200
66    Outline the polygon (POLYO)

56  Start a polygon definition (POLYS)
45 64 00 64 00    Move pointer to V1, 100,100 (POLYM)
57 C8 00 64 00    Define V2 at 200,100 (POLYV)
57 C8 00 C8 00    Define V3 at 200,200
45 64 00 C8 00    Move pointer to V4, 100,200
66    Outline the polygon (POLYO)
```
Display-Pointer-Move Instructions

Figure 5-5. Creating Two Open Polygons Using POLYM
Display-Pointer-Move Instructions

POLYS

Starts a polygon definition

Instruction Format

HEX:

56

DECIMAL:

86

ASCII:

V

Input Arguments

None

Outputs

None

Description

When it receives the POLYS instruction, the display controller begins constructing a new polygon, using the polygon vertices defined with succeeding POLYV instructions. Each POLYV instruction adds another vertex to the new polygon.

For further information on polygon construction, see the POLYV and POLYM instructions in this chapter. For information on drawing and filling polygons see the POLYO and POLYF instructions in Chapter 6.
Display-Pointer-Move Instructions

Range

Does not apply

Special Considerations

The POLYS instruction directs the display controller to build a new polygon but takes no display action on its own. The POLYV instruction adds vertices to the new polygon while the polygon remains in memory. The POLYO or POLYP instructions display the figure.
Display-Pointer-Move Instructions

POLYV

Adds a vertex to a polygon

Instruction Format

HEX: \( 57 \; \text{lox} \; \text{hix} \; \text{loy} \; \text{hiy} \)

DECIMAL: \( 87 \; x \; y \)

ASCII: \( \text{W} \; \text{lox} \; \text{hix} \; \text{loy} \; \text{hiy} \)

Input Arguments

- \( \text{lox} \): the low-order eight bits of the eleven bits required to define the x coordinate value
- \( \text{hix} \): only the three low-order bits of this byte are used; these three bits set the three high-order bits of the eleven bits required to define the x coordinate value
- \( \text{loy} \): the low-order eight bits of the ten bits required to define the y coordinate value
- \( \text{hiy} \): only the two low-order bits of this byte are used; these two bits set the two high-order bits of the ten bits required to define the y coordinate value

Outputs

- None

Description

This instruction adds each vertex to the polygon started with the POLYS instruction. Four bytes immediately following the opcode define the coordinate for each vertex.

The P1 and P2 pointers remain in their previous position; they are not moved during the POLYV instruction.
Display-Pointer-Move Instructions

Range

The allowable range for x coordinate data is 0 to 1279t, and for y coordinate data is 0 to 1023t. The maximum number of polygon vertices is at least:

- POLYO: 2000t
- POLYF: 360t (See POLYF instruction.)

Special Considerations

No display action takes place on the monitor during polygon construction. Once all vertices are defined, the polygon is filled using the POLYF Drawing Instruction, outlined using the POLYO Drawing Instruction, or filled and outlined in another color using both instructions.

Example

This example, listed in hexadecimal, creates a triangle using the POLYS and POLYV instructions, then outlines it with a POLYO instruction and fills it with a POLYF instruction. Figure 5-6 shows the outlined polygon.

```
56                      Start a polygon definition (POLYS)
57 64 00 64 00          Define V1 at 100,100t (POLYV)
57 CB 00 2C 01          Define V2 at 200,300t
57 2C 01 64 00          Define V3 at 300,100t
66                      Outline the polygon (POLYO)
67                      Fill the polygon (POLYF)
```

April 1, 1984
Display Pointer Move Instructions

Figure 5-6. Adding Vertices with POLYV
DISPLAY-POINTER-MOVE INSTRUCTIONS

RM0VP1

Moves pointer P1 to a relative coordinate

Instruction Format

HEX:
54 lox hix loy hiy

DECIMAL:
84 x y

ASCII:
T lox hix loy hiy

Input Arguments

lox  the low-order eight bits of the 12 bits required to define the 2's complement x coordinate move

hix  only the four low-order bits of this byte are used; these four bits set the four high-order bits of the 12 bits required to define the x coordinate move; note that 12-bit 2's complement arithmetic is used for defining relative moves; positive or negative move directions are allowed, depending on the 2's complement number assigned

loy  the low-order eight bits of the 12 bits required to define the 2's complement y coordinate move

hiy  only the four low-order bits of this byte are used; these four bits set the four high-order bits of the 12 bits required to define the y coordinate move; note that 12-bit 2's complement arithmetic is used for defining relative moves; positive or negative move directions are allowed, depending on the 2's complement number assigned

Outputs

None

April 1, 1984
Display-Pointer-Move Instructions

Description

This instruction moves P1 relative to its present position, by the specified coordinate distances. Four bytes immediately following the opcode define the relative distance. They form a 12-bit 2's complement argument for x and y.

Range

The allowable range of arguments for the x and y coordinate move is -2048 to +2047. The values are assigned in 12-bit 2's complement. Following a relative move, x must be between 0 and 1279, and y must be between 0 and 1023.

Special Considerations

None

Example

This example, listed in hexadecimal, moves the pointer P1 to an absolute decimal location (100,100) then moves P1, relative to that position, to decimal coordinates 250,250. Figure 5-7 shows the positions of P1.

52 64 00 64 00
54 96 00 96 00
Display-Pointer Move Instructions

Figure 5-7. Moving P1 to a Relative Location
Display-Pointer-Move Instructions

RM0VP2

Moves pointer P2 to a relative coordinate

Instruction Format

HEX: 55 lox hix loy hiy

DECIMAL: 85 x y

ASCII: U lox hix loy hiy

Input Arguments

lox  the low-order eight bits of the 12 bits required to define the 2's complement x coordinate move

hix  only the four low-order bits of this byte are used; these four bits set the four high-order bits of the 12 bits required to define the x coordinate move; note that 12-bit 2's complement arithmetic is used for defining relative moves; positive or negative move directions are allowed, depending on the 2's complement number assigned

loy  the low-order eight bits of the 12 bits required to define the 2's complement y coordinate move

hiy  only the four low-order bits of this byte are used; these four bits set the four high-order bits of the 12 bits required to define the y coordinate move; note that 12-bit 2's complement arithmetic is used for defining relative moves; positive or negative move directions are allowed, depending on the 2's complement number assigned

Outputs

None
Display-Pointer Move Instructions

Description

This instruction moves P2 by the specified increment, relative to the present position of P1. Four bytes immediately following the opcode define the relative distance in each direction. They form a 12-bit 2's complement argument for x and y.

Range

The allowable range of arguments for the x and y coordinate move is -2048 to +2047. The values are assigned in 12-bit 2's complement. Following a relative move, x must be between 0 and 1279, and y must be between 0 and 1023.

Special Considerations

None

Example

This example, listed in hexadecimal, moves the pointer P2 to a position relative to P1 which is currently located at 250,250. Therefore, after this example, P2 will be at location 350,350 which is (250 + 100)x, (250 + 100)y. Figure 5-8 shows the locations of P1 and P2.

55 64 00 64 00
Display-Pointer-Move Instructions

Current P1
(250,250)

P2 - Relative to P1
(250 + 100), (250 + 100)

Figure 5-8. Moving P2 to a Relative Location
The Drawing Instructions display graphic images on the monitor. The instructions in this chapter use the display pointers (P1 and P2) to draw lines, characters, and arcs, and to outline and fill polygons in a variety of colors.

- **AFILL1** - Fills a random area with the current color
- **AFILL2** - Fills a random area to a specific color
- **AFILL3** - Fills a random area with a new color
- **ARC** - Draws an arc or circle
- **CHAR** - Draws a character
- **CLEAR** - Clears image memory
- **COMPDR** - Draws a vector in the complement color
- **DRAW** - Draws a vector
- **FFILL** - FLASH-fills a rectangle
- **POLYF** - Fills a polygon
- **POLYO** - Outlines a polygon
- **RECT1** - Outlines a rectangle
- **RECT2** - Fills a rectangle
- **RLFILL** - Horizontally fills using a specific run length
- **XDRAW** - Draws a vector in the XOR of the current color
Drawing Instructions

AFILL 1

Fills a random area with the current color

Instruction Format

HEX: 68
DECIMAL: 104
ASCII: h

Input Arguments

None

Outputs

None

Description

When this instruction is issued, the area starting with and surrounding P1 is overwritten with the current drawing color as long as pixels in the P1 color are encountered. P1 and P2 remain unchanged after an area fill operation.

Special Considerations

- If there is more than one graphic area, and some areas are presently written in the current color, the P1 position determines the display as shown in Figures 6-1 and 6-2.
- The write mask affects the fill color.
Drawing Instructions

- Fill styles and raster-ops cannot be used.
- Filled areas may be limited in complexity due to stack overflow.
- AFILL1 alters the ROC (raster-op comparator).

Figure 6-1. P1 Set to a Different Color than the Inner Boundary

Fig. 6-2. P1 Set Between Boundaries of Two Different Colors
Drawing Instructions

AFILL 2
Fills a random area to a specific color

Instruction Format
HEX: 69 b
DECIMAL: 105 b
ASCII: i b

Input Arguments
b the color-map address of the edge color

Outputs
None

Description
AFILL2 begins filling a random area at P1 with the current drawing color until the specified edge color is reached. A one-byte argument following the opcode selects the color-map address of the edge color.

Range
The color-map address range is 0 through 255.

Special Considerations
- The fill color is affected by the current write mask.
- Fill styles and raster-ops cannot be used.
Drawing Instructions

- Filled areas may be limited in complexity due to stack overflow.
- AFILL2 alters the ROC (raster-op comparator).

Example

This example shows an AFILL2 instruction format that fills a polygon to its outer boundary color which is E0 (pure red in the default color map). Figure 6-3 shows how the position of P1 in graphic memory affects the figure in this example.

69 E0 Fill to the red boundary

Figure 6-3. Filling to the Outer Boundary with AFILL2
Drawing Instructions

AFILL 3

Fills a random area with a new color

Instruction Format

HEX: 74

DECIMAL: 116

ASCII: t

Input Arguments
None

Outputs
None

Description

AFILL3 fills a random area with a target color defined by the ROC (Raster-op Comparator) starting at P1. (See RASTOP and LDROC instructions in Chapter 7 of this manual.)

If the color at P1 is not the target color, no fill occurs. If the color at P1 is the target color, that pixel and all contiguous pixels containing the target color are filled.

AFILL3 is similar to AFILL1, except that a range of colors may be overwritten with the current drawing color.

Locations in the ROC set to a 1 (target colors) correspond to the pixel values to be filled. The ROC location corresponding to the current drawing color should not be set. AFILL3 clears this bit before filling; only in this case is the ROC modified by this command.
Special Considerations

- The current drawing color may not be a target color. If the current drawing color is a target color, the ROC will be changed so it is not a target color.
- If P1 falls on a pixel that does not contain a target color, a no-op will occur.
- P1 is unchanged after execution of an AFILL3.
Drawing Instructions

ARC

Draws an arc or circle

Instruction Format

HEX: 62 w

DECIMAL: 98 w

ASCII: b w

Input Arguments

w two bytes that define the length of the arc in number of pixels, in the range 0 to 2047

The number of pixels required to draw a given circle can be calculated using the following formula, then rounding up to the next higher integer: \(4r\sqrt{2} + 4\) (\(r\) is the circle’s radius.) For an arc of angle “a”, where “a” is given in degrees, use the formula: length = \(4ra\sqrt{2} / 360\)

Outputs

None

Description

The ARC instruction draws an arc or circle of a specific length. Two bytes immediately following the opcode specify the arc length in pixels. When this instruction is executed, an arc is drawn counterclockwise starting from P2, using P1 as the center point. P1 and P2 are defined previously, using the Display-Pointer-Move Instructions.

Arcs are drawn in the current drawing color, line style, and raster-op, and include the defined starting point P2.

6-8 April 1, 1984
Drawing instructions

Range

The arc can be specified from 0 to 2047 pixels long.

Special Considerations

P2 moves to the end point of the arc after it is drawn to easily continue drawing longer arcs or circles.

Example

This example, listed in hexadecimal, sets the center of the arc (P1) at 150, 150. Then it sets the radius (P2) at 50, 150 and draws an arc 283 pixels long. Figure 6-4 shows the resulting arc.

This arc is exactly one-half a circle. The length of the arc can be verified using 180 degrees for variable "a" in the formula listed under Input Arguments in this instruction.

52 96 00 96 00  Set P1 to 150, 150 (the arc's center)
53 32 00 96 00  Set P2 to 50, 150
62 1B 01  Draw an arc 283 pixels long

Figure 6-4. Drawing an Arc
CHAR

Draws a character

Instruction Format

HEX: 6B b...b (escape)

DECIMAL: 107 b...b (escape)

ASCII: k b...b (escape)

Input Arguments

b...b the ASCII text characters to be displayed

Outputs

None

Description

All displayable ASCII characters received after this instruction are displayed as text until an escape character is encountered. Characters are drawn using the character size, spacing, and rotation parameters selected previously. These parameters are described in the Drawing-Control Instructions in Chapter 7. Previously selected colors and drawing patterns also apply.

Range

ASCII codes 0 - 127t are 8 x 16 pixel characters on power-up. Codes 128 - 255t are undefined on power-up.

Special Considerations

- Characters may be redefined by using the LDFONT instruction.
Drawing Instructions

- When operating in text mode, the correct character spacing (between characters and between lines) is inserted as a part of each character block. Character spacing may be modified by the CSPACE and FSIZE commands.
- All normal upper and lower case ASCII characters can be displayed. In addition, Carriage Return (CR), Line Feed (LF), and Backspace (BS) perform their normal display functions. Escape (ESC) returns the Ω500 System to graphics mode.
- Stipple patterns are disabled during character drawing, however, the pattern mode bits remain in effect.
- Microcode versions 2.3 and before may require two CLEAR instructions after a CHAR instruction to clear the screen instead of one CLEAR instruction.

Example

This example, shown in ASCII, enters text mode, prints a string on the display, and returns to graphics mode. (See Figure 6-5.)

```
This is a Test. (escape)
```

Figure 6-5. Typing Characters on the Screen
Clearing Instructions

CLEAR

Clears image memory

Instruction Format

HEX: 60
DECIMAL: 96
ASCII: ' (apostrophe)

Input Arguments

None

Outputs

None

Description

This instruction clears all display memory or just selected memory planes to the current drawing color. Using the write mask instruction (WRMASK) prior to the CLEAR instruction, write-enables certain memory planes, clearing them to the current drawing color when the CLEAR instruction is issued. The SET COLOR instruction selects the current drawing color. Refer to the Drawing-Control Instructions in Chapter 7 for more information on WRMASK and SET COLOR.

Range

Does not apply.

April 1, 1984
Special Considerations

- When clearing individual memory planes, be certain that only those planes are write-enabled (WRMASK). If all planes are enabled (the initial state), then all memory will be cleared to the selected color.
- Both raster-op and pattern are ignored when this instruction is entered.
- Microcode versions 2.3 and before may require two CLEAR instructions to clear the screen instead of one CLEAR instruction.
Drawing Instructions

COMPDR

Draws a vector in the complement color

Instruction Format

HEX: 72

DECIMAL: 114

ASCII: r

Input Arguments

None

Outputs

None

Description

The COMPDR instruction draws a vector from P1 to P2 in the color complement of existing pixels.

Range

Does not apply

Special Considerations

- Vector drawing is noticeably slower using COMPDR.
- The current write mask, stipple patterns, and line patterns remain in effect. The current draw color and raster-op are ignored.
- P1 moves to the location of P2 after completing the COMPDR operation.
Drawing Instructions

Example

This example, listed in hexadecimal, draws a white line on a black screen. Figure 6-6 shows the line.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4E 00</td>
<td>Selects black</td>
</tr>
<tr>
<td>60</td>
<td>Blank the screen to the black</td>
</tr>
<tr>
<td>52 64 00 64 00</td>
<td>Set P1 to 100, 100t</td>
</tr>
<tr>
<td>53 C8 00 2C 01</td>
<td>Set P2 to 200, 300t</td>
</tr>
<tr>
<td>72</td>
<td>Draw a complemented line</td>
</tr>
</tbody>
</table>

Figure 6-6. Drawing a Line in the Complement Color
Drawing Instructions

DRAW

Draw a vector

Instruction Format

HEX: 61
DECIMAL: 97
ASCII: a

Input Arguments

None

Outputs

None

Description

The DRAW instruction draws a vector one pixel wide from P1 to P2, including both points. Define P1 and P2 before using this instruction with the Display-Pointer-Move Instructions, described in Chapter 5.

The vector is drawn in the currently selected drawing color, pattern, and raster-op.

A one-pixel dot is displayed if P1 and P2 are set to the same coordinate position.

Range

Does not apply
Special Considerations

Pi moves to the P2 coordinate after the vector is drawn so the line can be easily extended with relative moves and vector chains.

Example

This example, listed in hexadecimal, draws a line (vector) from 100,100t. Figure 6-7 shows the line.

52 64 00 64 00  Sets Pi to 100,100t
53 C8 00 2C 01  Sets P2 to 200,300t
61  Draws a vector from 100,100t to 200,300t

Figure 6-7. Drawing a Line
Drawing Instructions

FFILL

FLASH-Fills a rectangle

Instruction Format

HEX: 65

DECIMAL: 101

ASCII: e

Input Arguments

None

Outputs

None

Description

This instruction fills a rectangle quicker than other area-fill instructions. Its speed is gained by limiting the filling options. FLASH-fill always uses the current drawing color, only permits solid fills, and only operates on rectangles.

P1 and P2 define diagonally opposed corners of the rectangle and are part of the rectangle. P1 and P2 should be defined before using FLASH-fill. Refer to the Display-Pointer-Move instructions for more information about pointers.

Range

Does not apply

6-18

April 1, 1984
Drawing Instructions

Special Considerations

- The rectangle includes the corner points, P1 and P2, and they remain at the corners after the FLASH-fill operation.
- To fill a rectangle with the currently selected color and outline it in another color, first fill the rectangle with the FLASH-fill instructions then select another color and outline the rectangle with the DRAW instruction.

Example

This example, listed in hexadecimal, FLASH-fills a rectangle. Figure 6-8 shows the filled rectangle.

```
52 64 00 2C 01  Set P1 to 100,300
53 C8 00 64 00  Set P2 to 200,100
65        FLASH-fill the rectangle
```

Figure 6-8. FLASH-filling a Rectangle
POLYF

Fills a polygon

Instruction Format

HEX: 67

DECIMAL: 103

ASCII: g

Input Arguments

None

Outputs

None

Description

POLYF fills the current polygon using the current drawing color, fill pattern, and raster-op. Use the POLYS instruction and some combination of the POLYV, POLYC, and POLYM instructions to define the current polygon before filling. Chapter 5 describes these instructions.

Pointer 1 and Pointer 2 are not affected by the POLYF instruction.

Range

Does not apply.
Special Considerations

To draw a polygon of one color outlined in another color, first fill the polygon with the POLYF instruction. Then select another color and outline with the POLYO instruction. It is not necessary to redefine the vertices of the polygon. Performing this sequence in the opposite order (POLYO then POLYF), overwrites the outline color.

The POLYF command invokes a parity-fill algorithm which fills the interior region of convex, concave, non-planar, and nested polygons. can be reached from the left edge of memory after an odd number of polygon edge crossings and before an even number of edge crossings. The first crossing is number 1, not 0, and considered an odd crossing. (See the Example section.) The algorithm will properly fill arbitrarily complex polygons that do not exceed the formula:

\[ 2(V) + 9(A_{E_{\text{max}}}) \leq 4000 \]

Where:

\( V \) Total number of vertices in the polygon including all sub-polylines

\( A_{E_{\text{max}}} \) Maximum number of active edges the scan line will cross

Example

This example, listed in hexadecimal, draws two rectangles, one inside the other. The structure is filled with the POLYF instruction. Notice that following a POLYC with a PLOYV begins a new polygon definition without having to issue another POLYS instruction, and that issuing a POLYF automatically ends the definition. Only the outer area of the rectangle is filled according to the parity-fill formula. Figure 6-9 shows the result of the fill and marks the active edges.

56 Start a polygon definition (POLYS)
57 64 00 64 00 Define V1 at 100,100 (POLYV)
57 C8 00 64 00 Define V2 at 200,100

April 1, 1984
Drawing Instructions

57 C8 00 C8 00  Define V2 at 200,200
57 64 00 C8 00  Define V4 at 100,200
44  Close the polygon definition (POLYC)
57 96 00 96 00  Define V5 at 150,150
57 AF 00 96 00  Define V6 at 175,150
57 AF 00 AF 00  Define V7 at 175,175
57 96 00 AF 00  Define V8 at 150,175
67  Fill the polygons (POLYF)

NOTE: The outlines in this figure are for clarity. In the given sequence, no outlining would occur.

Figure 6-9. Filling a polygon
POLYO

Outlines a polygon

Instruction Format

HEX: 66

DECIMAL: 102

ASCII: f

Input Arguments

None

Outputs

None

Description

This instruction outlines the current polygon in the currently selected drawing color, line pattern, and raster-op. Use the POLYC, POLYM, and POLYV instructions to construct the polygon before outlining. (Refer to Chapter 5 for information about using these instructions.)

The position of P1 and P2 remain unchanged after executing a POLYO.

Range

Does not apply
Drawing Instructions

Special Considerations

To outline a polygon of one color with another color, first fill the polygon with the POLYF instruction, select another color for the outline (using the SET COLOR instruction in Chapter 7), and outline with the POLYO instruction. To overwrite the outline color, perform this operation in the opposite order (POLYO then POLYF). It is not necessary to redefine the vertices of the polygon.

Example

This example, listed in hexadecimal, draws and outlines a triangle. Figure 6-10 shows this polygon.

56 Start a polygon definition (POLYS)
57 64 00 64 00 Set V1 at 100,100t (POLYV)
57 C8 00 2C 01 Set V2 at 200,300t
57 2C 01 64 00 Set V3 at 300,100t
66 Outline the polygon (POLYO)

Figure 6-10. Outlining a Polygon
RECT1

Outlines a rectangle

Instruction Format

HEX: 63
DECIMAL: 99
ASCII: c

Input Arguments
None

Outputs
None

Description

This instruction outlines a rectangle (not a polygon) in vectors one pixel wide using the current draw color, line style, and raster-op. RECT1 requires fewer instruction to use that drawing an equivalent figure by defining a polygon.

The location and size of the rectangle depends on P1 and P2 which define the diagonally opposed corners of the rectangle. The rectangle includes P1 and P2. Their location remains unchanged after this instruction.

Range

Does not apply
Drawing Instructions

Special Considerations

To outline a rectangle in another color, first fill the figure with the POLYF or FFILL (FLASH-fill) instruction, select another color, then outline using the POLYO instruction.

Example

This example, listed in hexadecimal, draws a rectangle with the lower left corner at 100, 300t and the upper right corner at 200, 100t. Figure 6-11 shows the outlined figure.

52 64 00 2C 01 Set P1 to 100,300t
53 C8 00 64 00 Set P2 to 200,100t
63

Figure 6-11. Drawing a Rectangle Outline

F-0037
RECT2

Fills a rectangle

Instruction Format

HEX: 64

DECIMAL: 100

ASCII: d

Input Arguments
None

Outputs
None

Description

This instruction draws and fills a rectangle (not a polygon) using the current drawing color, fill pattern, and raster-op. Filling a figure with RECT2 requires fewer instructions than filling the same figure constructed as a polygon and filled with the POLYO instruction.

The size and location of the rectangle depends on P1 and P2 which defines the diagonally opposed corners. The rectangle includes P1 and P2. Their location remains unchanged after this instruction.

Range

Does not apply
Drawing Instructions

Special Considerations

To fill a rectangle with the currently selected color and outline it in another color, first fill the figure with the RECT2 or FFILL (FLASH-fill) instructions, select another color, then outline the figure using the RECT2 instruction.

Example

This example, listed in hexadecimal, displays a filled rectangle with the lower left corner at 100,300t and the upper right corner at 200,100t. Figure 6-12 shows the filled rectangle.

52 64 00 2C 01 Set P1 to 100,300t
53 CB 00 64 00 Set P2 to 200,100t
64

Figure 6-12. Filling a Rectangle
Drawing Instructions

RLFILL

Horizontally fills using a specific run length

Instruction Format

HEX: 6A w
DECIMAL: 106 w
ASCII: j w

Input Arguments

w two bytes specifying the number of pixels to fill

Outputs

None

Description

RLFILL draws a line a specific number of pixels long. The Q500 System fills from left to right starting at P1, parallel to the x axis. The pixels are filled in the current drawing color. Two bytes immediately following the opcode describe the length of the line.

One use for the RLFILL instruction is to compress data in a predefined picture sent from the host to the Q500 System.

Range

The pixel count range is from 1 to 1280.
Drawing Instructions

Special Considerations

- After RLFILL, P1 is located to the right of the last pixel filled.
- A fill length of zero results in no fill.

Example

This example, listed in hexadecimal, draws a line from 100,100t (in the current color) to 850x,100y.

52 64 00 64 00  Set P1 to 100,100t
6A EE 02  Fill with a run length=750t
**Drawing Instructions**

**XDRAW**

Draws a vector in the XOR of the current color

**Instruction Format**

**HEX:** 73

**DECIMAL:** 115

**ASCII:** s

**Input Arguments**

None

**Outputs**

None

**Description**

The XDRAW instruction draws a vector, one pixel wide, between P1 and P2, inclusive. The line is written using the Exclusive OR (XOR) of the pixel and its current drawing color. Define P1 and P2 before using this instruction with the Display-Pointer-Move Instructions described in Chapter 5. After the draw is completed, both P1 and P2 are set to the P2 position (vector end).

**Range**

Does not apply

**Special Considerations**

Line and area patterns are in effect, but should be used with caution. The currently selected raster-op is ignored.

April 1, 1984
Drawing Instructions

Example

This example, listed in hexadecimal, draws a line through a filled rectangle in the XOR of the background color. Using the default color map, the background color is green and the line drawing color is yellow but the XDRAW instruction draws a red line (the XOR of 3C and DC) through the rectangle.

4E 00 60  Blank the screen to black
4E 3C  Select green
52 FF 01 FF 01  Set P1
53 FF 02 FF 02  Set P2
65  Fill the rectangle (FFILL)
4E DC  Select yellow
73  XDRAW a line
CHAPTER 7

DRAWING-CONTROL INSTRUCTIONS

The Drawing-Control Instructions, listed below, affect the environment of the Drawing Instructions. They select the current drawing color, enable the read or write mask, select the fill and line patterns, and control the size and orientation of text characters.

- **CSPACE** - Sets the character spacing
- **FSIZE** - Sets the font size
- **LDROC** - Loads the raster-op comparator array
- **PATTERN** - Selects the fill or line pattern
- **RASTOP** - Specifies the raster-op
- **RDMASK** - Sets the read mask
- **SetColor** - Selects the drawing color
- **SETCORN** - Sets the character orientation
- **SETCSZ** - Sets the character size
- **WRMASK** - Sets the write mask
Drawing Control Instructions

CSPACE

Sets the character spacing

Instruction Format

HEX: 48 loAx hiAx loAy hiAy

DECIMAL: 72 Ax Ay

ASCII: H loAx hiAx loAy hiAy

Input Arguments

loAx the low-order eight bits of the 12 bits required to define the 2's complement x character spacing.

hiAx only the four low-order bits of this byte are used; these four bits set the four high-order bits of the 12 bits required to define the x character spacing; note that 12-bit 2's complement arithmetic is used for defining the spacing; positive or negative spacing directions are allowed, depending on the 2's complement number assigned.

loAy the low-order eight bits of the 12 bits required to define the 2's complement y character spacing.

hiAy only the four low-order bits of this byte are used; these four bits set the four high-order bits of the 12 bits 2's complement y character spacing.

Outputs

None
Description

This instruction controls the auto-incrementing of pointer P1 when character strings are used. The increment is applied after the character is drawn.

Range

-1279t < Δx < 1279t
-1023t < Δy < 1023t (positive Δy moves top to bottom)

Following a character draw, x must be between 0 and 1279t, and y must be between 0 and 1023t.

Special Considerations

The Δx and Δy character spacing is automatically adjusted according to the character orientation and the character size (see SETCORN, SETCSZ and FSIZE in this chapter).

The CSPACE instruction following aFSIZE, SETCORN or SETCSZ instruction overrides the default spacing. Note that the CSPACE must account for changes in SETCSZ. (See example 5.)

Example

This example has five parts. The first two show how changing the orientation of the character automatically changes the default setting for CSPACE. The next two examples, 3 and 4, show the same character spacing (CSPACE) but two different orientations. The last example, 5, shows how the CSPACE must account for an enlarged character. In example 5, the SETCZ instruction creates a character twice as large as normal, so the character spacing must change accordingly.
**Drawing-Control Instructions**

(1)  
59 00
6B ab (esc)

Sets the character orientation to 0 (SETCORN)

Types 'ab' to the screen (CHAR)

CSPACE automatically defaults to 8x,0y

(2)  
59 03
6B ab (esc)

Sets the character orientation to 3 (SETCORN)

Types 'ab' to the screen (CHAR)

CSPACE automatically defaults to 0x,8y

(3)  
59 00
48 00 10
6B ab (esc)

Sets the character orientation to 0 (SETCORN)

Sets CSPACE to 0x,16y

Types 'ab' to the screen

April 1, 1984
### Drawing-Control Instructions

#### (4)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>59 05</td>
<td>Sets the character orientation to 5</td>
</tr>
<tr>
<td>48 F0 FF 00</td>
<td>(SETCORN)</td>
</tr>
<tr>
<td>6B ab (esc)</td>
<td>Sets CSPACE to -16x,0y</td>
</tr>
<tr>
<td></td>
<td>Types 'ab' to the screen</td>
</tr>
</tbody>
</table>

#### (5)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>59 01</td>
<td>Sets the character orientation to 1</td>
</tr>
<tr>
<td>58 00 01</td>
<td>(SETCORN)</td>
</tr>
<tr>
<td></td>
<td>Sets the character size to 1</td>
</tr>
<tr>
<td></td>
<td>(twice as big as normal)</td>
</tr>
<tr>
<td>48 00 20 00</td>
<td>Sets CSPACE to 0x,32y</td>
</tr>
<tr>
<td>6B ab (esc)</td>
<td>Types 'ab' to the screen</td>
</tr>
</tbody>
</table>

April 1, 1984
**Drawing-Control Instructions**

**FSIZE**

Sets the font size

**Instruction Format**

**HEX:** 49 b₁ b₂

**DECIMAL:** 73 b₁ b₂

**ASCII:** I b₁ b₂

---

**Input Arguments**

b₁ an 8-bit number representing font width

b₂ an 8-bit number representing font height

**Outputs**

None

---

**Description**

This instruction invokes a viewport 8-pixels wide by b₂-pixels high on the 8 x 16 character cell. This instruction also calculates the default character spacing invoked by the SETCORN and the SETCSZ commands. In this regard, b₁ and b₂ specify character width and height.

**Range**

b₁: 1 to 8t

b₂: 1 to 16t

---

April 1, 1984
LDROC

Loads the raster-op comparator array

Instruction Format

**HEX**: 42 \( SC_1 \ldots SC_n \)

**DECIMAL**: 66 \( SC_1 \ldots SC_n \)

**ASCII**: B \( SC_1 \ldots SC_n \)

---

Input Arguments

- \( SC_1 \): A sub-command byte followed by 0 to 32 data bytes.
- \( SC_n \): An exit command

---

Outputs

None

---

Description

LDROC loads a calculated set of values into the raster-op comparator (ROC) array. The RASTOP instruction, in this chapter, enables and disables the ROC.

The ROC array has 256 locations, one for each color. Writing a one in a Raster-Op Comparator location enables writing that pixel; conversely, writing a zero in the location disables writing that pixel.

The LDROC opcode is followed by a variable number of bytes that specify the data patterns to load into the array. Bytes continue to be processed until an exit instruction is encountered. The LDROC parameters are defined as follows:
Drawing-Control Instructions

**0h**  
Exits the LDROC instruction.

**80h**  
Exits the LDROC instruction.

**1h**  
Resets ROC array to zeroes (no colors allowed to write).

**81h**  
Sets ROC array to ones (all colors allowed to write).

Use this command to ensure uniform array state before loading.

**2h**  
Resets a single location in the ROC array.

**82h**  
Sets a single location in the ROC array.

This command is followed by a byte defining the location to be altered.

**3h**  
Either command block-loads all 256 locations in the ROC array.

**83h**  
Sets a range of locations.

32 data bytes fill the array; the LSB of byte one defines location 0 of the ROC and the MSB of byte one defines location 7 of the ROC; the LSB of byte two defines location 8 of the ROC and the MSB of byte two defines location 15 of the ROC; and so on for all 32 bytes. A one sets the corresponding bit in the array, and a zero clears that bit.

**4h**  
Clears a range of locations.

**84h**  
Sets a range of locations.

Three bytes follow the parameter: the first sets the lower limit, the second sets the upper limit, and the third is a mask value. This is the algorithm for LDROC followed by either the 4h or 84h commands:

```
For address = 0 to 255 Do
    i = Bit_AND(address, mask)
    if (i >= lower) AND (i <= upper) then
        ROC[address] = value;
    End;
```
Special Considerations

- The AFILL1 and AFILL2 instructions alter the ROC.
- LDROC loops upon itself until an exit command is encountered.
- Use LDROC before using any RASTOP instruction.

Examples

1) This simple example, listed in hexadecimal, calculates and loads a zero into location E0h in the raster-op comparator array (ROC) so that no pixel with that value can be written. The value 02 clears a location and the value E0 is the address to clear. When the ROC is enabled, pixels in this color will not be written. Figure 7-1 shows the array after loading.

```
42 Load the raster-op comparator (LDROC)
81 Initialize the array with all 1's
02
EO Set location E0 to 0 (disable writes)
00 Exit
```

![Figure 7-1. Loading the Raster-Op Comparator Array](F-0004)
2) This more complex example is a table showing some selected values for the upper limit, the lower limit, and the mask in the LDROC algorithm. To better understand how the LDROC instruction works, try using working through the formula, shown under the 4h and 84h commands appearing earlier in this entry, using the values shown in the following table. The top group of values in this table show changes to single bit-planes; the bottom group of values show changes to two bit-planes.

<table>
<thead>
<tr>
<th>Lower</th>
<th>Upper</th>
<th>Mask</th>
<th>Conditions Allowing Writes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Bit-plane 0=0 (even-numbered planes)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Bit-plane 0=1 (odd-numbered planes)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>Bit-plane 1=0 (even-numbered planes)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>Bit-plane 1=1 (odd-numbered planes)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>Bit-plane 0=0 (even-numbered planes)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>Bit-plane 0=1 (odd-numbered planes)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>Bit-plane 0=1 AND bit-plane 1=1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>Bit-plane 0=1 AND bit-plane 1=0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>Bit-plane 0=0 AND bit-plane 1=1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Bit-plane 0=1 XOR bit-plane 1=0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>Bit-plane 0=1 OR bit-plane 1=1</td>
</tr>
</tbody>
</table>
Drawing-Control Instructions

PATTERN

Sets the fill or line pattern

Instruction Format

HEX: 50 b
DECIMAL: 80 b
ASCII: P b

Input Arguments

b the pattern-select byte; the eight bits are used as follows:

- **least-significant three bits**: set pattern
- **bit four**: select line (1) or fill (0) pattern
- **bit five**: normal (0) or inverted (1)

**most-significant three bits**: select mode

Outputs

None

Description

This instruction selects the line pattern used for drawing lines, and the stipple pattern for filling polygons. (Note that a solid line pattern will also select a solid fill.)

Eight bits following the opcode select these patterns. The least-significant three bits select one of eight possible line styles or one of seven possible area patterns. In addition, a solid line style is available that is also used for solid fill.

To set an area pattern, set bit 4 to 0. To set a line style or solid fill, set bit 4 to 1. To use data to describe the pattern, set bit 5 to 0. To switch the background and pattern, set bit 5 to 1.

April 1, 1984
Table 7-1 shows the effect of the 8 combinations of the three most-significant bits.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Pattern</th>
<th>Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (000)</td>
<td>data, all planes</td>
<td>zero, all planes</td>
</tr>
<tr>
<td>1 (001)</td>
<td>data, all planes</td>
<td>no change</td>
</tr>
<tr>
<td>2 (010)</td>
<td>data, selected planes</td>
<td>zero, all planes</td>
</tr>
<tr>
<td>3 (011)</td>
<td>data, selected planes</td>
<td>no change</td>
</tr>
<tr>
<td>4 (100)</td>
<td>data, all planes</td>
<td>zero, selected planes</td>
</tr>
<tr>
<td>5 (101)</td>
<td>same as 4</td>
<td>same as 4</td>
</tr>
<tr>
<td>6 (110)</td>
<td>data, selected planes</td>
<td>zero, selected planes</td>
</tr>
<tr>
<td>7 (111)</td>
<td>same as 6</td>
<td>same as 6</td>
</tr>
</tbody>
</table>

**Table Key**

- **Data** means the current drawing color.
- **All** means all memory planes are affected, regardless of the write mask.
- **Selected** means only the write-enabled memory planes are affected.
- **No change** means that memory is not written at all.
- **Background** means the bits other than the pattern.
- **Pattern** means the bits loaded to generate the line or fill style.
Range

Does not apply.

Special Considerations

The fourth bit determines whether the selected pattern is a line pattern or a fill pattern. Note, however, that selecting a solid line pattern simultaneously selects a solid fill pattern. If a solid line is used along with a pattern fill, the fill pattern must be reset after each solid line selection. The "normal" drawing pattern (solid lines, solid fills) is 68h. Note that using any other pattern will reduce the speed of polygon and rectangle fills.

Figure 7-2 shows the patterns available and the number that selects them. The LDPAT instruction in Chapter 10 explains how to change the default patterns.
Drawing-Control Instructions

**LINES**

- SOLID
- SHORT DASH
- LONG DASH
- DASH-DOT
- DASH-DOT-DOT
- DASH-DASH-DOT
- FINE DOT
- MEDIUM DASH

**AREA**

0 - HALF TONE
1 - DIAG, NEG SLOPE
2 - DIAG, POS SLOPE
3 - HORIZ DASH
4 - LONG DASH
5 - GRID
6 - COARSE GRID
7 - HERRINGBONE

Figure 7-2. Default Drawing and Filling Patterns

April 1, 1984
RASTOP

Specifies the raster-op

Instruction Format

HEX: 43 b₁ b₂

DECIMAL: 67 b₁ b₂

ASCII: C b₁ b₂

Input Arguments

b₁ specifies data flow

b₂ specifies ALU operation

Outputs

None

Description

RASTOP specifies the raster operation used for subsequent drawing instructions. Two bytes following the opcode specify b₁ and b₂.

Pixel-writing primitives fall into two classes: draws and transfers (PIXBLTs). The draw primitives include vectors, arcs, polygons, text, and rectangles. They use only the current drawing color and the destination pixels as ALU (Arithmetic Logic Unit) operands.

PIXBLTs are pixel block transfers, hence there is always a source and destination pixel for each transfer. The PIXBLTs use either the source pixel and the current drawing color or the source and destination pixels as inputs to the ALU.
Table 7-2 lists the data flow options set by parameter \( b_1 \). The table is divided into two levels; one for draw operations, the other for PIXBLT operations.

<table>
<thead>
<tr>
<th>( b_1 )</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Draw-Normal</td>
<td>( D = C )</td>
</tr>
<tr>
<td>1</td>
<td>Draw-ALU Enabled</td>
<td>( D = D \ \text{op} \ C )</td>
</tr>
<tr>
<td>2</td>
<td>Draw-Cndtnl</td>
<td>If (( D )), ( D = C )</td>
</tr>
<tr>
<td>3</td>
<td>Draw-ALU Cndtnl</td>
<td>If (( D )), ( D = D \ \text{op} \ C )</td>
</tr>
<tr>
<td>0</td>
<td>PIXBLT-Normal</td>
<td>( D = S )</td>
</tr>
<tr>
<td>1</td>
<td>PIXBLT-Cndtnl</td>
<td>If (( D )), ( D = S )</td>
</tr>
<tr>
<td>2</td>
<td>PIXBLT-Cndtnl</td>
<td>If (( S )), ( D = S )</td>
</tr>
<tr>
<td>3</td>
<td>PIXBLT-ALU</td>
<td>( D = C \ \text{op} \ S )</td>
</tr>
<tr>
<td>4</td>
<td>PIXBLT-ALU, Cndtnl</td>
<td>If (( D )), ( D = S \ \text{op} \ C )</td>
</tr>
<tr>
<td>5</td>
<td>PIXBLT-ALU, Cndtnl</td>
<td>If (( S )), ( D = C \ \text{op} \ S )</td>
</tr>
<tr>
<td>6</td>
<td>PIXBLT-ALU, Cndtnl</td>
<td>If (( S )), ( D = C )</td>
</tr>
<tr>
<td>7</td>
<td>PIXBLT-ALU</td>
<td>( D = S \ \text{op} \ D )</td>
</tr>
<tr>
<td>8</td>
<td>PIXBLT-ALU, Cndtnl</td>
<td>If (( D )), ( D = S \ \text{op} \ D )</td>
</tr>
<tr>
<td>9</td>
<td>PIXBLT-ALU, Cndtnl</td>
<td>If (( S )), ( D = S \ \text{op} \ D )</td>
</tr>
</tbody>
</table>

\( D \) represents the destination pixel (the pixel to be modified).

\( S \) represents the source pixel (used in bit operations only).

\( C \) represents the current drawing color.

ALU means that the color written to the destination pixel is affected by the selected ALU function (see Table 7-4).

Cndtnl means that the destination pixel will not be written if the Raster-Op Comparator disables the write.

If \( (D) \) means that the Raster-Op Comparator uses the destination pixel for the condition test.

If \( (S) \) means that the Raster-Op Comparator uses the source pixel for the condition test.
Parameter \( b_2 \) specifies the Pixel ALU operation. The ALU uses two operands, which are called \( Opl \) and \( Op2 \). For drawing primitives, the ALU always uses \( C \) and \( D \) as the ALU operands. For PIXBLTs, the ALU operands are either \( C \) and \( S \) or \( D \) and \( S \). The assignments of \( S \) (source), \( D \) (destination), and \( C \) (current color) for \( Opl \) and \( Op2 \) are as shown below in Table 7-3:

<table>
<thead>
<tr>
<th>Table 7-3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Op1 and Op2 Assignments</strong></td>
</tr>
<tr>
<td><strong>Primitive Type</strong></td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>Draw (1,3)</td>
</tr>
<tr>
<td>PIXBLT (3,5)</td>
</tr>
<tr>
<td>PIXBLT (4)</td>
</tr>
<tr>
<td>PIXBLT (7-9)</td>
</tr>
</tbody>
</table>

The ALU functions are shown in Table 7-4:

<table>
<thead>
<tr>
<th>Table 7-4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALU Functions</strong></td>
</tr>
<tr>
<td><strong>b_2</strong></td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

When first using the RASTOP instruction, the user is faced with an often confusing array of options. These narrow down to three overall operations:

1. Decide the data flow. This uses the choices listed in Table 7-2. The operation is either a Draw or a PIXBLT, so a choice is made from the upper or the lower level of the table. This determines the value of \( b_1 \).
**Drawing-Control Instructions**

2. Decide the ALU function. This choice is made from the next two tables, and determines the value of $b_2$.

3. Combine $b_1$ and $b_2$. Not all combinations are useful; some of the more useful combinations are listed in Table 7-5, Drawing Raster-Ops, and Table 7-6, PIXBLT Raster-Ops.

**Example**

An XOR Arc operation is drawn.

Table 7-6 is not used, since this is not a PIXBLT Raster-Op. Using Table 7-5, the code for arcs is 1 and 7 for $b_1$ and $b_2$.

To reach this conclusion:

1. Using Table 7-2, the upper level is used, since this is a draw operation. Since the draw is unconditional and must use the ALU function, the choice narrows to $b_1=1$.

2. Using Table 7-4, an XOR operation is 7 for $b_2$.

3. Table 7-3 is unused in this case, since a subtraction operation is not present. (Operand order is not important in this case.)
### Table 7-5
#### Drawing Raster-Ops

<table>
<thead>
<tr>
<th>b₁</th>
<th>b₂</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>Normal drawing mode</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>XOR mode for vectors, arcs, polygons, rectangles</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>And mode for vectors, arcs, polygons, rectangles</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td>Conditional write for vectors, arcs, polygons, rectangles</td>
</tr>
</tbody>
</table>

x represents a "don't care"

### Table 7-6
#### PIXBLT Raster-Ops

<table>
<thead>
<tr>
<th>b₁</th>
<th>b₂</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>D = S</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>D = C OR S</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>D = C EXOR S</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>D = S - C</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>D = C - S</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>D = S AND D</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>D = S EXOR D</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>If (S), D = C AND S</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>If (S), D = C EXOR S</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>If (D), D = S EXOR D</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>If (S), D = S EXOR D</td>
</tr>
</tbody>
</table>
Drawing-Control Instructions

RDMASK

Sets the read mask

Instruction Format

HEX: 4C b
DECIMAL: 76 b
ASCII: L b

Input Arguments

b the mask-select byte. All eight bits are used to select the read-enabled bit planes; the least-significant bit represents the low-order bit plane; the most-significant of the applicable bits selects the high-order bit plane

Outputs

None

Description

The RDMASK instruction, followed by the mask byte, enables or disables reading bit-planes. Each bit of the mask byte sets one bit-plane on (1) or off (0). Disabled bit-planes do not display data.

Range

The range is 0 through 255, corresponding to the 256 possible combinations of eight bit-planes.
Special Considerations

The read mask does not affect data transfers from the Ω500 system memory to a host; it affects only the display of data. At power-up or after an INIT instruction, RDMASK = FFh. (See Chapter 10 for information about INIT.)
Drawing-Control Instructions

SET COLOR

Selects the drawing color

Instruction Format

HEX: 4E b

DECIMAL: 78 b

ASCII: N b

Input Arguments

b a byte that selects one of 256t color-map addresses

Outputs

None

Description

SET COLOR selects one color from the color map for displaying vectors, fills, arcs, rectangles, and text characters until another color is selected with a subsequent SET COLOR instruction.

All eight bits of the color-select byte are used to set one of 256t possible colors at the current drawing color.

Range

The allowable range is 0 through 255t. (This corresponds to the 256 available addresses in the color map.)
Drawing-Control Instructions

Special Considerations

The SET COLOR instruction is the last of three steps that select the color to used to draw on the monitor screen. These are the steps:

(1) Select the colors to load into the color map using the CMAP instruction. (See Chapter 8.)

(2) Write-enable the desired memory planes using the WRMASK instruction described in this chapter.

No drawing occurs in the bit planes that are not enabled through the write mask. (All planes are normally enabled when the system is powered up, or when an INIT instruction occurs.)

At power-up or after an INIT, the color is set to 0. Chapter 10 contains more information about INIT.

(3) Set the color with the SET COLOR instruction.

Example

This example shows the hexadecimal format of the SET COLOR instruction. It selects color E0 from the color map. In the default color map, E0 is red at full intensity. Figure 7-3 shows how this instruction indexes the color map.

4E E0
A "1" bit increases the intensity of that color. For example, E0 is full-intensity red.

Figure 7-3. The SET COLOR Index to the Default Color Map
SETCORN

Sets the character orientation

Instruction Format

HEX: 59 b
DECIMAL: 89 b
ASCII: Y b

Input Arguments

b the orientation byte; the three least-significant bits of this byte specify one of the eight possible orientation combinations

Outputs

None

Description

SETCORN selects the orientation of the text on the screen. Table 7-7 and Figure 7-4 show how the bits in the orientation byte select the rotation of each character.

Table 7-7. Meaning of Bits in the Orientation Byte

<table>
<thead>
<tr>
<th>Orientation</th>
<th>Bits</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>Normal</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>Mirror</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>Rotate y</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>Mirror and rotate y</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>Rotate x</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>Mirror and rotate x</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>Rotate x and y</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>Rotate and mirror x and y</td>
</tr>
</tbody>
</table>

April 1, 1964
Drawing-Control Instructions

Range

The range is 0 through 7t.

Special Considerations

Refer to the Special Consideration section under the CSPACE instruction in this chapter.

Example

Refer to the example in the CSPACE instruction in this chapter.

Figure 7-4. Selecting the Orientation of Text
SETCSZ

Sets the character size

Instruction Format

HEX: 58 b₁ b₂

DECIMAL: 88 b₁ b₂

ASCII: X b₁ b₂

Input Arguments

b₁ the width scale-factor (x axis magnification)
b₂ the height scale-factor (y axis magnification)

Outputs

None

Description

This instruction enlarges or reduces the displayed text characters. Two bytes follow the opcode and select the character scale-factor for each axis.

The default character size (magnification 1X - scale factor 0) displays characters within a block 8 pixels wide by 16 pixels high. A one-pixel intercharacter space and a two-pixel interline gap is included in the block.

Range

The range of valid values in each axis is from 0 (1X) through 255 (256X).
Drawing-Control Instructions

WRMASK

Sets the write mask

Instruction Format

HEX: 4Fb

DECIMAL: 79b

ASCII: 0b

Input Arguments

b the mask-select byte. All eight bits are used to select the write-enabled bit planes; the least-significant bit represents the low-order bit plane; the most-significant of the applicable bits selects the high-order bit plane

Outputs

None

Description

Setting the write mask allows the Q500 System to write-enable specific bit planes. Each of the bits in the byte following the opcode is associated with one of the bit planes. These bits are used to set the write mask in any of the 256 possible combinations of the bit planes. A "one" bit write-enables the corresponding bit plane; a "zero" bit disables the corresponding plane.

The write mask is especially useful for writing some bit planes while leaving others unchanged. The CLEAR instruction, for instance, uses the write mask to select which memory planes to clear.

7-28 April 1, 1984
Range
The range is 0 through 255, corresponding to the possible combinations of eight bit-planes.

Special Considerations

- The SET COLOR instruction, described in this chapter, selects a drawing color from the available colors in the color map. To display the color, the appropriate memory plane must be write-enabled through the write mask.
- No drawing occurs in planes that are not write-enabled.
- All planes are normally enabled at power-up, and after an INIT instruction. (Write mask = FFh.)
CHAPTER 8

DISPLAY-CONTROL INSTRUCTIONS

The Display-Control Instructions, listed below, provide additional control over the graphics display. The six instructions in this group control the color map, the cursor, and the memory map to the monitor screen.

- **BLANK** - Blanks the display
- **BLINK** - Blinks the high-order bit plane
- **CMAP** - Loads the color map
- **CURS** - Displays the cursor
- **PPAN** - Pans to a defined origin
- **SZCUR** - Changes the size of the cursor
Display-Control Instructions

BLANK

Blanks the display

Instruction Format

HEX: 4B b
DECIMAL: 75 b
ASCII: K b

Input Arguments

b an 8-bit blank flag following the opcode
0 normal mode
1 blank lower four bit planes
2 blank upper four bit planes
3 blank all bit planes

Outputs
None

Description

The BLANK instruction, followed immediately by the blank-flag byte, selects which bit-planes are not displayed.

Blanking speeds up writing new display data into the Ω500 display memory by blanking the display while writing into it. Pixel writing is approximately four times faster when the screen is blanked. However, the speed enhancement of instructions that perform computation as well as pixel writing, such as AFILL1, will not be as significant as less complex instructions.

April 1, 1984
Display-Control Instructions

Range

The range for b is 0 through 3t.

Special Considerations

The BLANK instruction affects the read mask and write mask, as outlined in the following table, while blanking is enabled:

<table>
<thead>
<tr>
<th>Blank Flag</th>
<th>Effective Read Mask</th>
<th>Effective Write Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mask</td>
<td>Mask</td>
</tr>
<tr>
<td>1</td>
<td>Mask &amp; F0h</td>
<td>Mask &amp; 0Fh</td>
</tr>
<tr>
<td>2</td>
<td>Mask &amp; 0Fh</td>
<td>Mask &amp; F0h</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>Mask</td>
</tr>
</tbody>
</table>

Mask refers to the currently selected read or write mask.
& means bit-wise AND

Example

This example shows how to use the table to calculate the effective read or write mask.

If the read mask is A5h, the write mask is 52h, and the blank flag is one, the effective read mask is the combination of the A5h read mask and the F0h blanking, or A0h. That is, the lower four planes are disabled and two of the upper planes are still disabled. The effective write mask is 02h, which likewise is the combination of the 52h write mask and the 0Fh blanking.

When blank mode 0 is selected, both masks are restored to their original values (0 at power-up or after INIT).
Display-Control Instructions

BLINK

Blinks the high-order bit plane

Instruction Format

HEX: 4D b
DECIMAL: 77 b
ASCII: M b

Input Arguments

b  the 8-bit blink flag following the opcode
    0  blink off
    1  blink on

Outputs

None

Description

The BLINK instruction, followed immediately by a flag byte, blinks the eighth (highest-order) bit plane for emphasis. If the flag is a 0, BLINK is off. If it is a 1, the high-order bit plane is switched in and out at 2 Hz.

Range

The range for the blink flag is 0 or 1.
Special Considerations

To emphasize a particular portion of the display, that portion must be written into the high-order bit plane. This effectively maps colors 128 through 255 into 0 through 127.
Display-Control Instructions

CMAP

Loads the color map

Instruction Format

**HEX:** \(51 \ b_1 \ b_2 \ b_3 \ b_4\)

**DECIMAL:** \(81 \ b_1 \ b_2 \ b_3 \ b_4\)

**ASCII:** \(Q \ b_1 \ b_2 \ b_3 \ b_4\)

Input Arguments

- \(b_1\): the color-map address; the binary sum of these bits select one color-map address to be loaded; all eight bits of this byte are used.
- \(b_2\): Red intensity byte: a value of zero (0) turns the red off; a value of 255 sets full intensity.
- \(b_3\): Green intensity byte
- \(b_4\): Blue intensity byte

Outputs

None

Description

This instruction loads the colors into the color map. Once loaded, these colors can be selected as the current drawing color using the SET COLOR instruction described in Chapter 7.

Four parameters follow the CMAP opcode. The first selects which color map entry to change, and the last three specify new red, green, and blue intensities for the entry. Three bytes of color specification allow color selection from a palette of 16.7 million colors.
Display-Control Instructions

Range

Color-map addresses and intensities range from 0 through 255t.
Display-Control Instructions

CURS
Displays the cursor

Instruction Format
HEX: 71
DECIMAL: 113
ASCII: q

Input Arguments
None

Outputs
None

Description
This instruction places a crosshair cursor at P1 on the display. It remains on the display until another instruction is received by the Q500 System.

The cursor is 33 pixels high and 33 pixels wide and is displayed in the complement color of the pixels it writes. The SZCUR instruction, described in this chapter, changes the size of the cursor.

Range
Does not apply
Special Considerations

- The cursor display is immediately disabled when any other instruction is received.
- The cursor is not affected by the write mask or the line patterns.

Example

Figure 8-1 shows the cursor displayed on the screen.

Figure 8-1. Displaying the Cursor
Display-Control Instructions

PPAN

Pans to a defined origin

Instruction Format

HEX: 5B
DECIMAL: 91
ASCII: [

Input Arguments
None

Outputs
None

Description

This instruction moves the origin of the displayed memory area (the upper left corner) to the P1 position. P1 is positioned prior to the PPAN instruction using a Display-Pointer-Move Instruction. PPAN is most often used to move to a particular area of display memory.

Range

Does not apply
Special Considerations

The PPAN instruction pans by moving P1. While P1 can usually move one pixel at a time, when PPAN is invoked P1 must move at least 40 pixels at a time in the x-axis and at least 16 pixels at a time in the y-axis. If a smaller increment is selected, the instruction defaults to the next highest multiple. For example, if a y-axis movement of 39 pixels is selected, the default is 3 x 16, or 48 pixels.

If P1 is moved to the right, the display wraps around. That is, the display memory area to the right of the viewable area appears on the left of the monitor screen. Likewise, vertical movement of P1 causes vertical wraparound to occur.

Example

This example, listed in hexadecimal, establishes 120x, 112y as the origin point in display memory. Figure 8-2 shows this origin point and how it maps to the screen.

```
52 78 00 70 00  Sets P1 to 120,112
5B                   After Panning
      Sets P1 to 120,112
Pans display memory to P1
```

![Figure 8-2. Panning to a New Display-Memory Origin](F-0047)
Display Control Instructions

SZCUR

Changes the size of the cursor

Instruction Format

HEX: 47 w₁ w₂

DECIMAL: 71 w₁ w₂

ASCII: G w₁ w₂

Input Arguments

w₁ : two bytes that control cursor width; actual cursor width is 2w₁ + 1. The first byte consists of the lower eight of the eleven x-axis bits used to express cursor width; the second (high-order) byte uses only three of the eight bits.

w₂ : two bytes that control cursor height; actual cursor height is 2w₂ + 1. The first byte consists of the lower eight of the ten y-axis bits used to express cursor height; the second (high-order) byte uses only two of the eight bits.

Outputs

None

Description

This instruction programs the size of the cursor. The default cursor size is 33t pixels wide and 33t pixels high. The largest possible cursor measures 1279t pixels wide by 1023t pixels high and fills the screen.

8-12
April 1, 1984
Display-Control Instructions

Range

Width ($w_1$) can vary from 1 to 1279t. Height ($w_2$) can vary from 1 to 1024t.

Example

This decimal example shows the format for selecting a cursor 60 pixels wide by 60 pixels high.

71 30 30 Specify one half the total cursor size
CHAPTER 9

DATA-TRANSFER INSTRUCTIONS

The Data-Transfer Instructions move blocks of graphics data either between the Q500 System and the host computer or from one location to another in memory. When using a graphics tablet, another transfer instruction selects the mode for graphic input and local cursor control.

- **GRAFIN** - Selects the graphic input mode
- **PIXBLT** - Transfers blocks of pixels
- **RDR** - Reads a rectangle
- **RPIXEL** - Reads a pixel
- **WPIXEL** - Writes a pixel
- **WRR** - Writes a rectangle
Data-Transfer Instructions

GRAFIN

Selects the graphic input mode

Instruction Format

HEX: 4Af

DECIMAL: 74 f

ASCII: J f

Input Arguments

<table>
<thead>
<tr>
<th>f</th>
<th>the GRAFIN mode-select byte:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Software INIT</td>
</tr>
<tr>
<td>1</td>
<td>Local Cursor Control</td>
</tr>
<tr>
<td>2</td>
<td>Set Transparent Mode</td>
</tr>
<tr>
<td>3</td>
<td>Set Offset and Scale Factors</td>
</tr>
<tr>
<td>4</td>
<td>Set Delimiter</td>
</tr>
<tr>
<td>5</td>
<td>Sample Current Screen x,y Coordinates</td>
</tr>
<tr>
<td>6</td>
<td>Sample Current Tablet x,y Coordinates</td>
</tr>
<tr>
<td>7</td>
<td>Set Mode Register</td>
</tr>
</tbody>
</table>

Outputs

None

Description

The GRAFIN instruction, followed by a mode-select byte, controls the input mode for data received by a graphics tablet. The next several sections describe the mode selected by each mode-select byte.

0  Software INIT. Resets all GRAFIN attributes to the default values and clears the coordinate queue. The attributes include the offset and scale factors, plus
the mode register. Delimiter is set to 80 (hex) as the default. Scale and offset are set for the Summagraphics or GTCO tablet, depending on internal switch positions.

1 **Local Cursor Control.** A non-zero device status message (a switch closure, for instance) causes the full device message to be sent to the host. (See Table 9-1 for the full device status message format.) The cursor is displayed in the color complement of existing pixels. The x,y values are returned in Q500 coordinates. This mode is immediately disabled when any other instruction is received.

The cursor is continually repositioned at a rate limited by the device stream rate or the video refresh rate, whichever is slower. No drawing operations are allowed during GRAFIN.

<table>
<thead>
<tr>
<th>Table 9-1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Q500 STATUS MESSAGE FORMAT</strong></td>
</tr>
<tr>
<td>Byte</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

*coordinates are those of the screen cursor, after applying offset and scaling; transparent mode should be used if full tablet precision is required. kp = key pressed. This detects the "0" key on the GTCO tablet.

2 **Set Transparent Mode.** The Ω500 is effectively removed from the data link between the Ω500 and the host, and full-duplex communication continues until the delimiter is received to terminate Transparent Mode. The default delimiter is 80 (hex). Binary communications are supported with the exclusion of the
Data-Transfer Instructions

delimiter and hex value FF. The x,y values are returned in the raw tablet coordinates.

3 Set Offset and Scale Factors. The device coordinates are subjected to an offset and scale operation in the Q500 for cursor position control. The next eight bytes specify the following, in two's complement format:

   1 - x offset low byte
   2 - x offset high byte
   3 - x multiplier integer
   4 - x multiplier fraction
   5 - y offset low byte
   6 - y offset high byte
   7 - y multiplier integer
   8 - y multiplier fraction

4 Set Delimiter. The next byte specifies the delimiter, replacing the default delimiter. Transmitting the delimiter during Transparent Mode terminates the mode. The delimiter can range from 0 to FEh; the default delimiter is 80h.

5 Sample Current Screen X,Y Coordinates. The screen cursor position is relayed in the same format that is sent by GRAFIN 1.

6 Sample Current Tablet X,Y Coordinates. The full tablet coordinate message is relayed upon receipt of this command.

7 Set Mode Register. The Mode Register has the following bit definitions:
### Data-Transfer Instructions

<table>
<thead>
<tr>
<th>Bit</th>
<th>IF = 0 (default)</th>
<th>IF = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Wraparound</td>
<td>Clip to screen boundary</td>
</tr>
<tr>
<td>1</td>
<td>Level buttons</td>
<td>Edge buttons</td>
</tr>
<tr>
<td>2</td>
<td>Button xmit</td>
<td>Button not xmit</td>
</tr>
<tr>
<td>3</td>
<td>Leading edge report only</td>
<td>Both edges report</td>
</tr>
</tbody>
</table>

**NOTE:** These bits may be written as a group, by sending GRAFIN mode 7, followed by a byte with bits 0, 1, and 2 appropriately set and bit 7 = 1.

Alternately, set or clear an individual bit without modifying the others by setting bit 7 to zero, bit 3 to zero or one (for clear or set, respectively), and bits 0, 1, and 2 to a pointer value (i.e., 000 for bit 0, 001 for bit 1, and 010 for bit 2).

### Special Considerations

The GRAFIN function allows the Q500 Display Controller to interface to a Summagraphics Bit Pad One or a GTCO Graphic Tablet.

Note that the coordinate values returned are mode-dependent. In Local-Cursor Mode, the values are those of the Q500. In Transparent Mode, however, the Q500 does not interpret tablet data, but simply passes raw tablet coordinates to the host.

Scale (multiplier) and offset factors are applied to tablet data as follows:

\[(1 \text{ Scale} \times (\text{tablet data} + \text{offset})\]

Since the tablet origin is located in the upper left corner, negative values are required in the y direction.

The use of edge mode is recommended to ensure that only one coordinate message is sent for each button depression.

*April 1, 1984*
Data-Transfer Instructions

Note: The following values are a good starting point for offset and scale factors (hexadecimal values are shown):

Summagraphics: 4A03 0000 D800 DEF9 5AFF
Data Transfer Instructions

PIXBLT

Transfers blocks of pixels

Instruction Format

HEX: 70 w₁ w₂ b

DECIMAL: 112 w₁ w₂ b

ASCII: p w₁ w₂ b

Input Arguments

w₁ these two bytes follow the opcode to determine the width of the block to be moved; the first byte is the lox byte; it determines the low-order 8 bits of the 12 bits required to define the width; the least-significant 4 bits of the next byte (hix) determine the 4 high-order width bits

w₂ these two bytes determine the height of the block to be moved; the first byte is the loy byte; it determines the low-order 8 bits of the 12 bits required to define the height; the least-significant 4 bits of the next byte (hiy) determine the four high-order bits of the height.

b this byte determines the direction of the block from the pointers, in the source and destination

Outputs

None

Description

This instruction moves a block of pixels of a specified width, height, and direction, from one area in display memory to

April 1, 1964
Data-Transfer Instructions

another area in display memory. P1 defines the initial point of
the source block and P2 defines the initial point of the destina-
tion.

The direction of the block from the initial point is defined by
the last byte of the input argument. Five bits (0 through 4) of
this byte provide 32 possible direction combinations; direction
is set for both the source block and the destination.

Bit 0 of the direction byte, when asserted, swaps the x and y
axes in the destination. This rotates the block at the destina-
tion.

Bit 1 of the direction byte is the y destination direction. When
not asserted, the block is incremented in the y direction (from
the initial point) by the count specified in the height (w2)
bytes. When bit 1 is asserted, the y count is decremented by the
specified count, from the initial point.

Bit 2 works the same way as bit 1 for the x axis. It determines
the direction from the initial point of the previously specified
width-count.

The last two bits, 3 and 4, determine the direction from P1 of
the width and height counts in the source. The source block-
count is incremented in the y direction when bit 3 is not
asserted; the y count is decremented when bit 3 is asserted.
Bit 4 works the same way to set the width count direction.

Range

The height range is 0 through 1023t, and the width range is 0
through 1279t.

Special Considerations

The currently selected raster-op, write mask, and pattern
remain in effect. For certain raster-ops, the drawing color
affects the data transfers.
Data-Transfer Instructions

RDR

Reads a rectangle

Instruction Format

HEX: 6E

DECIMAL: 110

ASCII: n

Input Arguments
None

Outputs

A rectangle of pixels defined by P1 and P2

Description

This instruction reads the pixels in a rectangle from the Q500 System's display memory and transfers the data to the host computer. The corners of the rectangle are defined by P1 and P2. P1 and P2 are set prior to issuing the RDR instruction.

One byte is transferred to the host for each pixel. Total memory space and transfer requirements can be determined by calculating the total number of pixels in the rectangle (Δx times Δy). Δx and Δy can be calculated by subtracting the P1 and P2 minimum coordinates from the maximum coordinates + 1. The pixels are read from left to right, and from top to bottom.
Data-Transfer Instructions

Range

Does not apply

Special Considerations

- When reading a rectangle to the host, the read mask, the current drawing color, and the pattern are ignored.
- Existing data in all planes is transferred to the host, in the defined rectangle area.
- RDR does not change P1 or P2, the current drawing color, drawing pattern, read mask, or write mask.
- If P1 and P2 are set to the same point, one byte will be returned.
- To read the entire display memory, set P1 to (0,0) and set P2 to (1279,1023t). These values make the Δx and Δy 1280t and 1024t respectively.

Example

\[
P1 = (2,2) \\
P2 = (5,7) \\
\text{Pixel count} = (5-2+1)*(7-2+1) = 4*6 = 24 \text{ pixels.}
\]
Data-Transfer Instructions

RPIXEL
Reads a pixel

Instruction Format
HEX: 6C
DECIMAL: 108
ASCII: 1

Input Arguments
None

Outputs
One byte, for the pixel color at P1

Description
The RPIXEL instruction causes the Q500 System to return the pixel data at P1. All eight bits of the byte returned represent the color-map address.

Range
The color-map address range is 0 through 255.

Special Considerations
- This instruction does not change P1 or P2, the current drawing color, the drawing pattern, or the write mask.
- When reading a pixel to the host, the read mask, current drawing color, and pattern are ignored.
Data-Transfer Instructions

- Existing data in all planes is transferred to the host for the defined pixel.
Data-Transfer Instructions

WPXIEL

Writes a pixel

Instruction Format

HEX: 6D

DECIMAL: 109

ASCII: m

Input Arguments

None

Outputs

None

Description

When the Ω500 System receives a WPXIEL instruction, it writes a single pixel at P1 using the currently selected drawing color. P1 is defined using a Display-Pointer-Move Instruction, described in Chapter 5.

Range

Does not apply

Special Considerations

- When writing a pixel from the host, the write mask remains in effect so that no writing occurs in disabled memory planes.
Data-Transfer Instructions

- This instruction does not change P1 or P2, the current drawing color, the drawing pattern, or the write mask.
WRR

Writes a rectangle

Instruction Format

HEX: 6F

DECIMAL: 111

ASCII: o

Input Arguments

\( \Delta x \) times \( \Delta y \) bytes (corresponding to the rectangle of pixels)

Outputs

None

Description

This instruction writes a rectangle (defined by P1 and P2) from the host computer into the Q500 display memory. P1 and P2 point to diagonally opposed corners of the rectangle. (P1 may be the lower left corner and P2 may be the upper right corner, for instance.) Define pointers prior to issuing the WRR instruction.

One byte is transferred from the host for each pixel in the rectangle. Total memory space and transfer requirements can be determined by calculating the total number of pixels in the rectangle (\( \Delta x \) times \( \Delta y \)). \( \Delta x \) and \( \Delta y \) may be calculated by subtracting the P1 and P2 minimum coordinates from the maximum coordinates + 1. The pixels are loaded from left to right and from top to bottom.
Data-Transfer Instructions

Range

Pixel values range from 0 to 255.

Special Considerations

- When writing a rectangle from the host, the write mask remains in effect so that no writing can occur in disabled memory planes. To write into all planes, the write mask must be so enabled prior to this instruction.
- The pattern register (line and fill patterns) and raster-op also affect the WRR operation.
- This instruction does not change P1 or P2, the current drawing color, the drawing pattern, or the write mask.
- If P1 and P2 are the same point, only 1 pixel is written.
- To fill the entire display memory, set P1 to (0,0), and set P2 to (1279,1023). Δx and Δy will then be 1280t and 1024t respectively.
CHAPTER 10

UTILITY INSTRUCTIONS

The Utility Instructions control system functions. These are essentially non-graphic or special functions that report the status of the system, restore defaults, load new patterns or fonts into memory, or read diagnostic information.

- DEFAULT - Restores the default (power-on) values
- INIT - Initializes the system
- INQ - Reports the status of the system
- LDFONT - Loads a new font into memory
- LDPAT - Specifies a new pattern
- READ CONF - Reads and returns the system configuration
- SIG READ - Reads the signature-analyzer register
- SYNCH - Waits for vertical retrace
Utility Instructions

DFAULT

Restores the default (power-on) values

Instruction Format

HEX: 3F b

DECIMAL: 63 b

ASCII: ? b

Input Arguments

b a byte that selects the item to reset.

0 reset color map
1 reset character font
2 reset line and area stipple definition
3 reset raster-op
4 reset current drawing pattern

Outputs

None

Description

DFAULT selects which parameter to initialize. These parameters include the color map, character font, line and area stipple, raster-op, and current drawing pattern.

Range

0 through 4
Utility instructions

Special Considerations

The parameter named in the byte following the opcode is restored to these power-up defaults:

Color map
The color map is initialized so that bits 0 and 1 of the color address select blue intensity, bits 2, 3, and 4 select green intensity, and bits 5, 6, and 7 select red intensity.

Character font
The characters 0 through 127 of the font are reset to the power-up definition (8 by 16 character cell). See the LDFONT instruction described in this chapter.

Stipple definition
The stipbles are restored to the standard Omega stipple patterns. See the LDPAT instruction described in this chapter.

Raster-op
The raster-op is reset to normal drawing mode. See the RASTOP instruction described in Chapter 7.

Drawing pattern
The drawing pattern is reset to solid lines and fills. See the PATTERN instruction described in Chapter 7.
Utility Instructions

INIT

Initializes the system

Instruction Format

HEX: 5E
DECIMAL: 94
ASCII: ↑

Input Arguments

None

Outputs

None

Description

When the Q500 receives this instruction, it resets all attributes to their default values. The INIT instruction is identical to the power-up reset, except that no self-test is performed during execution of an INIT. The Q500 default values are:

- Pan = 0,0
- Pattern Register (Solid lines, solid fills) = 68h
- Color Register = 0
- Write Mask (All planes enabled) = FFh
- Read Mask (All planes enabled) = FFh
- Color Map = Default color selection
- GRAFIN: Scale, Offset, and Delimiter = defaults
- RASTOP (normal drawing mode) = 0
- Cursor Size = 33 pixels high and 33 pixels wide
- Character Size = 0
- Character Orientation = 0
Utility Instructions

INQ

Reports the status of the system

Instruction Format

HEX: 41 w

DECIMAL: 65 w

ASCII: A w

Input Arguments

w This word specifies which drawing parameter to return. The first byte contains the lower 8 bits of the 12-bit number, the lower four bits of the second byte contain the remaining bits. See Table 10-1.

Outputs

b₁ The lower 8 bits of the drawing parameter.
b₂ The upper 8 bits of the drawing parameter.

Description

The INQ instruction interrogates the Ω500 and reports the status of any one of 19 drawing parameters listed in Table 10-1. The Ω500 reports the status with a low byte followed by a high byte.

Range

The range for drawing-parameter arguments is 1 through 19t.

April 1, 1984
Utility Instructions

Special Considerations

None

Table 10-1
INQ Parameters

<table>
<thead>
<tr>
<th>Parameter #</th>
<th>Drawing Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P1X</td>
</tr>
<tr>
<td>2</td>
<td>P1Y</td>
</tr>
<tr>
<td>3</td>
<td>P2X</td>
</tr>
<tr>
<td>4</td>
<td>P2Y</td>
</tr>
<tr>
<td>5</td>
<td>horizontal char size</td>
</tr>
<tr>
<td>6</td>
<td>vertical char size</td>
</tr>
<tr>
<td>7</td>
<td>char orientation</td>
</tr>
<tr>
<td>8</td>
<td>font width</td>
</tr>
<tr>
<td>9</td>
<td>font height</td>
</tr>
<tr>
<td>10</td>
<td>dx after each character</td>
</tr>
<tr>
<td>11</td>
<td>dy after each character</td>
</tr>
<tr>
<td>12</td>
<td>current board number</td>
</tr>
<tr>
<td>13</td>
<td>memory mode</td>
</tr>
<tr>
<td>14</td>
<td>drawing pattern</td>
</tr>
<tr>
<td>15</td>
<td>raster-op</td>
</tr>
<tr>
<td>16</td>
<td>read mask</td>
</tr>
<tr>
<td>17</td>
<td>write mask</td>
</tr>
<tr>
<td>18</td>
<td>drawing color</td>
</tr>
<tr>
<td>19</td>
<td>grafin id</td>
</tr>
</tbody>
</table>
Utility instructions

LDFONT

Loads a new font into memory

Instruction Format

**HEX:** 3E b CDB

**DECIMAL:** 62 b CDB

**ASCII:** > b CDB

---

**Input Arguments**

b defines the first character to be changed

CDB Character Definition Block

**Outputs**

None

---

**Description**

LDFONT loads a custom font into font RAM. The information following the opcode redefines a contiguous group of characters. The first byte following the opcode describes the first character to change. Next, Character Definition Blocks (CDB), up to 17 bytes long, specify the new bit patterns.

The CDB begins with a length byte, which encodes the row numbers of the character cell to be changed. The length byte states the start and stop rows of the character being redefined. (The row start-address is in the lower nibble of byte one, and the row stop-address is in the upper nibble.) If 

\[(\text{STOP-START}+1) > 0\]

then that number of data bytes follow the CDB length byte.

A negative result ends the LDFONT instruction. For example,
Utility Instructions

An escape code (1Bh) results in a negative number and ends the instruction. The CDBs repeat, defining subsequent characters, until a terminating length byte occurs.

Special Considerations

- A 1Bh (escape character) terminates the LDFONT instruction.
- All rows in the character cell not set by the CDB are cleared to 0.
- Some characters are non-printing (e.g., BS, LF, CR) and cannot be refined.

Example

The following hexadecimal example redefines the characters 'o' and 'p':

<table>
<thead>
<tr>
<th>LDFONT COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Eh</td>
</tr>
<tr>
<td>6Fh</td>
</tr>
<tr>
<td>A5h</td>
</tr>
<tr>
<td>3Ch</td>
</tr>
<tr>
<td>42h</td>
</tr>
<tr>
<td>42h</td>
</tr>
<tr>
<td>42h</td>
</tr>
<tr>
<td>42h</td>
</tr>
<tr>
<td>42h</td>
</tr>
<tr>
<td>3Ch</td>
</tr>
<tr>
<td>A2h</td>
</tr>
<tr>
<td>40h</td>
</tr>
<tr>
<td>40h</td>
</tr>
<tr>
<td>40h</td>
</tr>
<tr>
<td>5Ch</td>
</tr>
<tr>
<td>62h</td>
</tr>
<tr>
<td>42h</td>
</tr>
<tr>
<td>42h</td>
</tr>
<tr>
<td>42h</td>
</tr>
<tr>
<td>42h</td>
</tr>
<tr>
<td>7Ch</td>
</tr>
<tr>
<td>1Bh</td>
</tr>
</tbody>
</table>

April 1, 1984
LDPAT

Specifies a new pattern

Instruction Format

HEX: 40 b₁ \([2^*b₂] \text{ or } [32^*b₂]\)

DECIMAL: 64 b₁ \([2^*b₂] \text{ or } [32^*b₂]\)

ASCII: @ b₁ \([2^*b₂] \text{ or } [32^*b₂]\)

Input Arguments

\(b₁\) this byte specifies which pattern to change.

<table>
<thead>
<tr>
<th>Bits 0-2</th>
<th>lower three pattern select lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 3</td>
<td>line (1) or area (0)</td>
</tr>
<tr>
<td>Bit 4</td>
<td>upper pattern select bit</td>
</tr>
</tbody>
</table>

\(b₂\) 2 or 32 data bytes

Outputs

None

Description

LDPAT changes the stipple pattern in memory. Thirty-one patterns, divided into 15 line styles and 16 area patterns, can be chosen. (Line style 0 is reserved for solid lines and solid area fills and cannot be redefined.) The bytes following the opcode describe these line or area patterns.

Lines. Three bytes define lines. The first parameter (\(b₁\)) specifies the pattern number to alter and selects lines instead of areas. The second parameter (\(b₂\)) defines the pattern, using the second and third bytes to do so. The MSB of the third byte is the first pixel drawn, and the LSB of the second byte is the last, as shown at the end of this section. When drawing objects,
Utility Instructions

patterns repeat once each 16 pixels until the image is completed.

Areas. Thirty-three bytes define areas. The first parameter (b₁) specifies the pattern number to alter and selects areas instead of lines. The second parameter (b₂) defines the pattern, using the remaining 32 bytes to do so. Of the 32 data bytes, the MSB of the second data byte corresponds to the top left-most bit of the pattern. See example below. The LSB of the 31st data byte is at the bottom right corner of the display. When drawing objects, patterns repeat once every 16 pixels until the image is completed.

Line

```
7 0 7 0
data2 data1
```

Area

```
7 0 7 0
data2 data1
data4 data3
*
*
data32 data31
```

(0 = LSB; 7 = MSB)

Range

b₁: 0 through 31t

Special Considerations

Line style 0 cannot be redefined.
Utility Instructions

READ CONF

Reads and returns the system configuration

Instruction Format

HEX: 5D

DECIMAL: 93

ASCII: ]

Input Arguments

None

Outputs

\[ \begin{align*}
  b_1 & \quad \text{hardware configuration} \\
  b_2 & \quad \text{microcode version number}
\end{align*} \]

Description

When the Q530 receives this instruction, it returns two bytes to the host system indicating the resolution of the system and the version number of the microcode.

The Q530 supports two resolutions: 1280 by 1024 pixels or 640 by 512 pixels. The system reports an eight in the lower four bits of the first byte when using 1280 by 1024 resolution. It reports a nine when using 640 by 512.

Range

Does Not Apply

April 1, 1984
Utility Instructions

SIG READ

Reads the signature-analyzer register

Instruction Format

HEX: 5C
DECIMAL: 92
ASCII: \n
Input Arguments
None

Outputs
Two signature bytes

Description

When the Q500 receives this instruction, it returns two bytes that represent data from an internal signature analyzer, and represent data transferring from the color-map register to the video digital-to-analog converter. Signatures aid troubleshooting and system testing and are used internally during power-up self-test.

Each display generates an individual signature. Signatures compared to those received during previous displays of the same image should be identical.
Utility Instructions

SYNCH

Waits for vertical retrace

Instruction Format

HEX: 5F n
DECIMAL: 95 n
ASCII: _ n

Input Arguments
n the wait-count byte

Outputs
None

Description

This instruction, followed by a single byte, synchronizes the display with the 60-cycle refresh rate. The eight bits of the wait byte specify a delay count from 0 to 255.

When the 0500 receives the SYNCH instruction, it waits for n+1 vertical-synch pulses before executing the next instruction. If the count is set to 0, the next instruction executes after the next vertical-synch pulse (0+1). If the count is set to 255, the 0500 waits 255 vertical-synch pulses, then executes another instruction after the next vertical-synch pulse occurs (255+1).

The SYNCH instruction is useful in animation applications, and for other applications where time delays and specific scanning start points are required.

April 1, 1984
Utility Instructions

Range

The range for the wait count is 0 through 255.
CHAPTER 11
FOLDED-MODE INSTRUCTIONS

The Folded-Mode Instructions, listed below, set the various drawing and display parameters that are relevant only to folded mode. Folded-mode operation is available only on the Q530 Display Controller.

- **BANK** - Selects the memory bank
- **OVMAP** - Loads the overlay color map
- **SELRES** - Selects the memory resolution
- **XBARLD** - Loads the cross-bar switch
Folded-Mode Instructions

BANK
Selects the memory bank

Instruction Format
HEX: 3D b
DECIMAL: 61 b
ASCII: = b

Input Arguments
b the bank-select byte. On the Q530, the least-significant four bits select the memory bank for subsequent pixel operations.

Outputs
None

Description
In folded mode, the pixel memory is configured as 32 bit-planes at 640 horizontal by 512 vertical resolution. The 32 bit-planes are divided into four banks of memory, named bank 0 through bank 3. Each bank contains 8 planes of pixel memory.

The BANK instruction selects one of the banks for subsequent pixel reads or writes. Each bank has its own drawing color and write mask. When a new bank is selected, the previous color and write mask for that bank is restored.
Folded-Mode Instructions

The opcode is followed by a byte that selects the memory bank. Each of the least-significant four bits of the byte selects one of the four memory banks. Thus a 1 selects bank 0, a 2 selects bank 1, a 4 selects bank 2, and an 8 selects bank 3. Only one bank may be selected at a time.

Range

1, 2, 4, or 8

Special Considerations

- When a bank is selected, its previous drawing color and write mask are restored.
- Plane 0 of bank 2 supplies the data for one of the two overlay planes, plane 0 of bank 3 supplies the other overlay plane. See the OVMAP and XBARLD instructions in this chapter for enabling and selecting colors for overlay planes.
- After power-up or after an INT instruction, bank 0 is selected, the drawing color for all banks is set to 0, and the write mask for all banks is set to FFh (all planes in the bank are enabled).
Folded-Mode Instructions

OVMAP

Loads the overlay color map

Instruction Format

HEX: 3A a r g b
58 a r g b

ASCII: : a r g b

Input Arguments

a the overlay color map address
r the red intensity
g the green intensity
b the blue intensity

Outputs
None

Description

In folded mode, the OVMAP instruction, followed by four bytes, selects which color-map entry to change, and selects the corresponding new red, green, and blue intensities.

Substituting these values for the 'a' argument selects the address of one of three overlay planes:

- A 1 modifies the entry that corresponds to the bank 2 overlay plane.
- A 2 modifies the entry that corresponds to the bank 3 overlay plane.
- A 3 modifies the entry that corresponds to both bank 2 and bank 3 overlay planes.
Folded-Mode Instructions

Substituting these values for the 'r' argument specifies the intensity value for red:

- 0 (no red)
- 80h (75% red intensity)

Specifying these values for the 'g' argument specifies the intensity values for green:

- 0 (no green),
- 40h (25% green intensity)
- 80h (50% green intensity)
- C0h (75% green intensity)

Specifying these values for the 'b' argument specifies the intensity value for blue:

- 0 (no blue)
- 80h (75% blue intensity)

Range

- a: 1 through 3t
- r: 0 or 80h
- g: 0, 40h, 80h, or C0h
- b: 0 or 80h

Special Considerations

- An overlay plane is 'on' wherever it is non-zero. This displays the overlay color for that plane if the overlay plane is enabled (see XBARLD in this chapter).
- Enable the overlay plane before loading the overlay color map.
- Overlay color-map address 3 is used when both overlay planes are on at the same time.

April 1, 1984
Folded-Mode Instructions

- Note that plane 0 of bank 2 supplies the overlay plane corresponding to color map address 1, and that plane 0 of bank 3 supplies the overlay plane for address 2.

Example

To set bank 3's overlay plane to maximum green intensity (75%):

   3A 02 00 C0 00

To select cyan (blue + green) when both overlay planes are enabled:

   3A 03 00 C0 80
SELRES

Selects the memory resolution

Instruction Format

HEX: \text{3C b}

DECIMAL: 60 b

ASCII: < b

Input Arguments

\text{b} \quad the \text{ memory-resolution select byte.} \quad \text{The least-significant bit of this byte selects the memory resolution:}

\begin{align*}
0 & \quad \text{Folded mode (640 } \times \text{ 512 } \times \text{ 32)} \\
1 & \quad \text{High-resolution mode (1280 } \times \text{ 1024 } \times \text{ 8)}
\end{align*}

Outputs

None

Description

This instruction changes the memory resolution between folded mode and high-resolution mode. In high-resolution mode, the pixel memory is configured as 8 bit-planes at 1280 horizontal by 1024 vertical resolution. In folded mode, the pixel memory is configured as 32 bit-planes at 640 horizontal by 512 vertical resolution.

Range

0 or 1
**Folded-Mode Instructions**

**Special Considerations**

Although the memory configuration may be changed by the SELRES command, many monitors will not be able to run at both resolutions. In high-resolution, the monitor must be capable of a 65.7 KHz horizontal scan rate and 120 MHz pixel rate. In folded mode, the monitor must be capable of a 33 KHz horizontal scan rate and 30 MHz pixel rate.
XBARLD

Loads the cross-bar switch

Instruction Format
HEX: 3B b
DECIMAL: 59 b
ASCII: ; b

Input Arguments
b the cross-bar data byte. This byte selects which memory banks feed the color-lookup tables in folded mode and enables the two overlay planes.

Outputs
None

Description
The XBARLD instruction, followed by a single byte, selects how the four memory banks drive the three color-lookup tables. The cross-bar data byte is split into four fields, each two bits wide.

Bits 0 and 1 of the data byte specify which memory bank supplies data to the blue lookup table. Likewise, bits 2 and 3 specify the green lookup table, and 4 and 5 specify the red lookup table. Bit 6 enables the bank 2 overlay plane, and bit 7 enables the bank 3 overlay plane.
Folded-Mode Instructions

Range

0 through 255

Special Considerations

Enable the overlay planes before loading the overlay color-map.

Example

To display a 24-bit-deep image whose blue component is in bank 0, whose green component is in bank 1, and whose red component is in bank 2, set bank 0 as whose source for the blue lookup table, bank 1 for the green, and bank 2 for the red. The parameter b would be:

0 0 0 1 0 1 0 0 = 24h

Setting bank 0 as the source for the blue lookup table, bank 0 for the green, and bank 0 for the red may be useful to display an eight-bit-deep image stored in bank 0. The parameter b would be:

0 0 0 0 0 0 0 0 = 00h

Setting bank 1 as the source for the blue lookup table, bank 1 for the red, and bank 2 for the red may be useful to display a 16-bit image stored in bank 1 (red and blue data) and bank 2 (green data). The parameter b would be:

0 0 0 1 0 1 0 1 = 19h

To set bank 1 as the source for all three lookup tables with the bank 3 overlay plane enabled, the parameter b would be:

1 0 0 1 0 1 0 1 = 95h

11-10

April 1, 1984
**APPENDIX A**

Table A-1. Code Conversion Chart

<table>
<thead>
<tr>
<th>ASCII</th>
<th>DECIMAL</th>
<th>OCTAL</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SOH</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>STX</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ETX</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>EOT</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>ENQ</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>ACK</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>BEL</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>BS</td>
<td>8</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>HT</td>
<td>9</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>LF</td>
<td>10</td>
<td>12</td>
<td>A</td>
</tr>
<tr>
<td>VT</td>
<td>11</td>
<td>13</td>
<td>B</td>
</tr>
<tr>
<td>FF</td>
<td>12</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>CR</td>
<td>13</td>
<td>15</td>
<td>D</td>
</tr>
<tr>
<td>SO</td>
<td>14</td>
<td>16</td>
<td>E</td>
</tr>
<tr>
<td>SI</td>
<td>15</td>
<td>17</td>
<td>F</td>
</tr>
<tr>
<td>DLE</td>
<td>16</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>DC1</td>
<td>17</td>
<td>21</td>
<td>11</td>
</tr>
<tr>
<td>DC2</td>
<td>18</td>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>DC3</td>
<td>19</td>
<td>23</td>
<td>13</td>
</tr>
<tr>
<td>DC4</td>
<td>20</td>
<td>24</td>
<td>14</td>
</tr>
<tr>
<td>NAK</td>
<td>21</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>SYN</td>
<td>22</td>
<td>26</td>
<td>16</td>
</tr>
<tr>
<td>ETB</td>
<td>23</td>
<td>27</td>
<td>17</td>
</tr>
<tr>
<td>CAN</td>
<td>24</td>
<td>30</td>
<td>18</td>
</tr>
<tr>
<td>EM</td>
<td>25</td>
<td>31</td>
<td>19</td>
</tr>
<tr>
<td>SUB</td>
<td>26</td>
<td>32</td>
<td>1A</td>
</tr>
<tr>
<td>ESC</td>
<td>27</td>
<td>33</td>
<td>1B</td>
</tr>
<tr>
<td>FS</td>
<td>28</td>
<td>34</td>
<td>1C</td>
</tr>
<tr>
<td>GS</td>
<td>29</td>
<td>35</td>
<td>1D</td>
</tr>
<tr>
<td>RS</td>
<td>30</td>
<td>36</td>
<td>1E</td>
</tr>
<tr>
<td>US</td>
<td>31</td>
<td>37</td>
<td>1F</td>
</tr>
</tbody>
</table>
Appendix A

Table A-1. Code Conversion Chart

<table>
<thead>
<tr>
<th>ASCII</th>
<th>DECIMAL</th>
<th>OCTAL</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>32</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>!</td>
<td>33</td>
<td>41</td>
<td>21</td>
</tr>
<tr>
<td>&quot;</td>
<td>34</td>
<td>42</td>
<td>22</td>
</tr>
<tr>
<td>#</td>
<td>35</td>
<td>43</td>
<td>23</td>
</tr>
<tr>
<td>$</td>
<td>36</td>
<td>44</td>
<td>24</td>
</tr>
<tr>
<td>%</td>
<td>37</td>
<td>45</td>
<td>25</td>
</tr>
<tr>
<td>&amp;</td>
<td>38</td>
<td>46</td>
<td>26</td>
</tr>
<tr>
<td>'</td>
<td>39</td>
<td>47</td>
<td>27</td>
</tr>
<tr>
<td>(</td>
<td>40</td>
<td>50</td>
<td>28</td>
</tr>
<tr>
<td>)</td>
<td>41</td>
<td>51</td>
<td>29</td>
</tr>
<tr>
<td>*</td>
<td>42</td>
<td>52</td>
<td>2A</td>
</tr>
<tr>
<td>+</td>
<td>43</td>
<td>53</td>
<td>2B</td>
</tr>
<tr>
<td>,</td>
<td>44</td>
<td>54</td>
<td>2C</td>
</tr>
<tr>
<td>-</td>
<td>45</td>
<td>55</td>
<td>2D</td>
</tr>
<tr>
<td>/</td>
<td>47</td>
<td>57</td>
<td>2F</td>
</tr>
<tr>
<td>0</td>
<td>48</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>49</td>
<td>61</td>
<td>31</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>62</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>51</td>
<td>63</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>52</td>
<td>64</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>53</td>
<td>65</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>54</td>
<td>66</td>
<td>36</td>
</tr>
<tr>
<td>?</td>
<td>55</td>
<td>67</td>
<td>37</td>
</tr>
<tr>
<td>8</td>
<td>56</td>
<td>70</td>
<td>38</td>
</tr>
<tr>
<td>9</td>
<td>57</td>
<td>71</td>
<td>39</td>
</tr>
<tr>
<td>:</td>
<td>58</td>
<td>72</td>
<td>3A</td>
</tr>
<tr>
<td>;</td>
<td>59</td>
<td>73</td>
<td>3B</td>
</tr>
<tr>
<td>&lt;</td>
<td>60</td>
<td>74</td>
<td>3C</td>
</tr>
<tr>
<td>=</td>
<td>61</td>
<td>75</td>
<td>3D</td>
</tr>
<tr>
<td>&gt;</td>
<td>62</td>
<td>76</td>
<td>3E</td>
</tr>
<tr>
<td>?</td>
<td>63</td>
<td>77</td>
<td>3F</td>
</tr>
<tr>
<td>@</td>
<td>64</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>A</td>
<td>65</td>
<td>101</td>
<td>41</td>
</tr>
<tr>
<td>B</td>
<td>66</td>
<td>102</td>
<td>42</td>
</tr>
<tr>
<td>C</td>
<td>67</td>
<td>103</td>
<td>43</td>
</tr>
<tr>
<td>D</td>
<td>68</td>
<td>104</td>
<td>44</td>
</tr>
</tbody>
</table>

A-2

April 1, 1984
# Appendix A

## Table A-1. Code Conversion Chart

<table>
<thead>
<tr>
<th>ASCII</th>
<th>DECIMAL</th>
<th>OCTAL</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>69</td>
<td>105</td>
<td>45</td>
</tr>
<tr>
<td>F</td>
<td>70</td>
<td>106</td>
<td>46</td>
</tr>
<tr>
<td>G</td>
<td>71</td>
<td>107</td>
<td>47</td>
</tr>
<tr>
<td>H</td>
<td>72</td>
<td>110</td>
<td>48</td>
</tr>
<tr>
<td>I</td>
<td>73</td>
<td>111</td>
<td>49</td>
</tr>
<tr>
<td>J</td>
<td>74</td>
<td>112</td>
<td>4A</td>
</tr>
<tr>
<td>K</td>
<td>75</td>
<td>113</td>
<td>4B</td>
</tr>
<tr>
<td>L</td>
<td>76</td>
<td>114</td>
<td>4C</td>
</tr>
<tr>
<td>M</td>
<td>77</td>
<td>115</td>
<td>4D</td>
</tr>
<tr>
<td>N</td>
<td>78</td>
<td>116</td>
<td>4E</td>
</tr>
<tr>
<td>O</td>
<td>79</td>
<td>117</td>
<td>4F</td>
</tr>
<tr>
<td>P</td>
<td>80</td>
<td>120</td>
<td>50</td>
</tr>
<tr>
<td>Q</td>
<td>81</td>
<td>121</td>
<td>51</td>
</tr>
<tr>
<td>R</td>
<td>82</td>
<td>122</td>
<td>52</td>
</tr>
<tr>
<td>S</td>
<td>83</td>
<td>123</td>
<td>53</td>
</tr>
<tr>
<td>T</td>
<td>84</td>
<td>124</td>
<td>54</td>
</tr>
<tr>
<td>U</td>
<td>85</td>
<td>125</td>
<td>55</td>
</tr>
<tr>
<td>V</td>
<td>86</td>
<td>126</td>
<td>56</td>
</tr>
<tr>
<td>W</td>
<td>87</td>
<td>127</td>
<td>57</td>
</tr>
<tr>
<td>X</td>
<td>88</td>
<td>130</td>
<td>58</td>
</tr>
<tr>
<td>Y</td>
<td>89</td>
<td>131</td>
<td>59</td>
</tr>
<tr>
<td>Z</td>
<td>90</td>
<td>132</td>
<td>5A</td>
</tr>
<tr>
<td>\</td>
<td>91</td>
<td>133</td>
<td>5B</td>
</tr>
<tr>
<td>]</td>
<td>92</td>
<td>134</td>
<td>5C</td>
</tr>
<tr>
<td>^</td>
<td>93</td>
<td>135</td>
<td>5D</td>
</tr>
<tr>
<td>_</td>
<td>94</td>
<td>136</td>
<td>5E</td>
</tr>
<tr>
<td>a</td>
<td>95</td>
<td>137</td>
<td>5F</td>
</tr>
<tr>
<td>b</td>
<td>96</td>
<td>140</td>
<td>60</td>
</tr>
<tr>
<td>c</td>
<td>97</td>
<td>141</td>
<td>61</td>
</tr>
<tr>
<td>d</td>
<td>98</td>
<td>142</td>
<td>62</td>
</tr>
<tr>
<td>e</td>
<td>99</td>
<td>143</td>
<td>63</td>
</tr>
<tr>
<td>f</td>
<td>100</td>
<td>144</td>
<td>64</td>
</tr>
<tr>
<td>g</td>
<td>101</td>
<td>145</td>
<td>65</td>
</tr>
<tr>
<td>h</td>
<td>102</td>
<td>146</td>
<td>66</td>
</tr>
<tr>
<td>i</td>
<td>103</td>
<td>147</td>
<td>67</td>
</tr>
<tr>
<td>j</td>
<td>104</td>
<td>150</td>
<td>68</td>
</tr>
</tbody>
</table>

April 1, 1984
## Appendix A

Table A-1. Code Conversion Chart

<table>
<thead>
<tr>
<th>ASCII</th>
<th>DECIMAL</th>
<th>OCTAL</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>105</td>
<td>151</td>
<td>69</td>
</tr>
<tr>
<td>j</td>
<td>106</td>
<td>152</td>
<td>6A</td>
</tr>
<tr>
<td>k</td>
<td>107</td>
<td>153</td>
<td>6B</td>
</tr>
<tr>
<td>l</td>
<td>108</td>
<td>154</td>
<td>6C</td>
</tr>
<tr>
<td>m</td>
<td>109</td>
<td>155</td>
<td>6D</td>
</tr>
<tr>
<td>n</td>
<td>110</td>
<td>156</td>
<td>6E</td>
</tr>
<tr>
<td>o</td>
<td>111</td>
<td>157</td>
<td>6F</td>
</tr>
<tr>
<td>p</td>
<td>112</td>
<td>160</td>
<td>70</td>
</tr>
<tr>
<td>q</td>
<td>113</td>
<td>161</td>
<td>71</td>
</tr>
<tr>
<td>r</td>
<td>114</td>
<td>162</td>
<td>72</td>
</tr>
<tr>
<td>s</td>
<td>115</td>
<td>163</td>
<td>73</td>
</tr>
<tr>
<td>t</td>
<td>116</td>
<td>164</td>
<td>74</td>
</tr>
<tr>
<td>u</td>
<td>117</td>
<td>165</td>
<td>75</td>
</tr>
<tr>
<td>v</td>
<td>118</td>
<td>166</td>
<td>76</td>
</tr>
<tr>
<td>w</td>
<td>119</td>
<td>167</td>
<td>77</td>
</tr>
<tr>
<td>x</td>
<td>120</td>
<td>170</td>
<td>78</td>
</tr>
<tr>
<td>y</td>
<td>121</td>
<td>171</td>
<td>79</td>
</tr>
<tr>
<td>z</td>
<td>122</td>
<td>172</td>
<td>7A</td>
</tr>
<tr>
<td>{</td>
<td>123</td>
<td>173</td>
<td>7B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>124</td>
<td>174</td>
</tr>
<tr>
<td>}</td>
<td>125</td>
<td>175</td>
<td>7D</td>
</tr>
<tr>
<td>~</td>
<td>126</td>
<td>176</td>
<td>7E</td>
</tr>
<tr>
<td>RUBOUT</td>
<td>127</td>
<td>177</td>
<td>7F</td>
</tr>
<tr>
<td>(DEL)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>