Ω500 Display Controller Reference Manual

(Order Number 004-02207-00)

April 1, 1984

Metheus Corporation
5510 N.E. Elam Young Parkway
Hillsboro, OR 97123

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

Revision Number	Revision History	Date
-00	First Issue. Describes the Ω500 Display Controller Instructions for version 2.3 microcode.	4/84

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.

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.

CONTENTS

FIGURES

TABLES

CHAPTER 1

INTRODUCTION TO THE Ω500

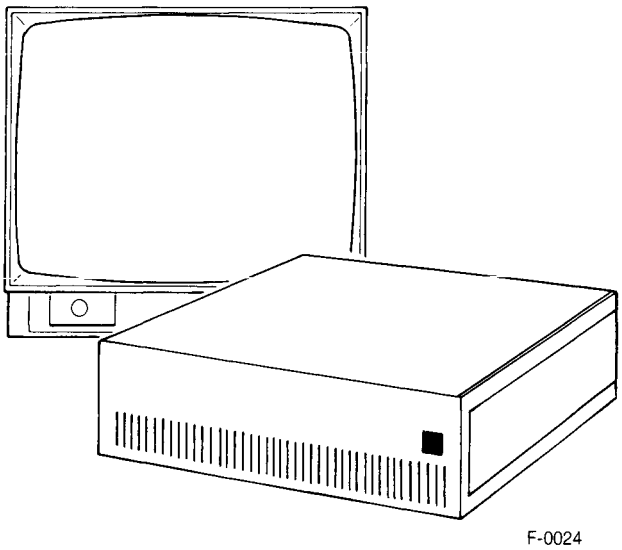


Figure 1-1. The Ω500 Display Controller and Monitor

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-fee display. The difference between the two models is the amount of memory resolution they provide.

The Ω510

The Ω510 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 Ω510 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 Ω530

The Ω530 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 Ω530 can be operates in folded mode. Folded mode provides 32 bit-planes, each with 640 x 512 resolution.

Speed

The Ω500 writes four to ten times faster than other system 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 Ω500 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.

- o Display units
- o Coordinate addresses
- o Lines, rectangles, arcs, and polygons
- o Pointers
- o Pan
- o Raster-ops
- o 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.

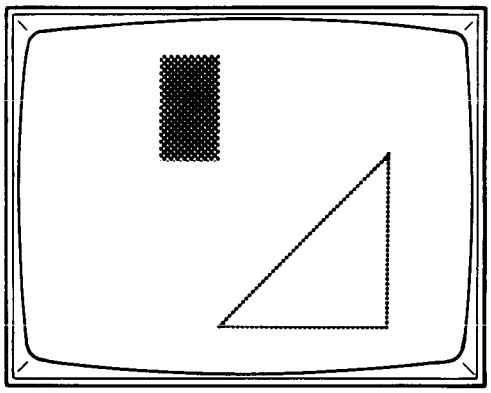


Figure 2-1. Creating Images With Pixel Combinations

Coordinate Addresses

The Ω500 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 consists 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 Ω530 coordinates range from 0 through 639 for x and 0 through 511 for y.

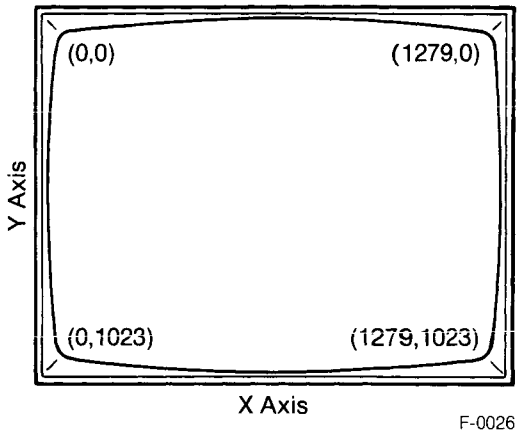


Figure 2-2. Coordinate Addresses

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 POLYV 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 Ω530 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 PIXBLT 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 tranfers for that pixel location. The LDROC instruction, in Chapter 7, loads values into the ROC.

Ω530 Folded Mode

The Ω530 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 2-1. Ω500 Instructions Listed Alphabetically

INSTRUCTION	CHAPTER TITLE	CHAPTER NUMBER
AFILL1	Drawing Instructions	6
AFILL2	Drawing Instructions	6
AFILL3	Drawing Instructions	6
ARC	Drawing Instructions	6
BANK	Folded-Mode Instructions	11
BLANK	Display-Control Instructions	8
BLINK	Display-Control Instructions	8
CHAR	Drawing Instructions	6
CLEAR	Drawing Instructions	6
CMAP	Display-Control Instructions	8
CMPDR	Drawing Instructions	7
CSPACE	Drawing-Control Instructions	7
CURS	Display-Control Instructions	8
DRAW	Drawing Instructions	6
DFAULT	Utility Instructions	10
FFILL	Drawing Instructions	6
FSIZE	Drawing-Control Instructions	7
GRAFIN	Data-Transfer Instruction	9
INIT	Utility Instructions	10
INQ	Utility Instructions	10
LDFONT	Utility Instructions	10
LDPAT	Utility Instructions	10
LDROC	Drawing-Control Instructions	7
MOVP1	Display-Pointer-Move Instructions	5
MOVP2	Display-Pointer-Move Instructions	5
OVMAP	Folded-Mode Instructions	11
PATTERN	Drawing-Control Instructions	7
PIXBLT	Data-Transfer Instruction	9
POLYC	Display-Pointer-Move Instructions	5
POLYM	Display-Pointer-Move Instructions	5
POLYF	Drawing Instructions	6
POLYO	Drawing Instructions	6
POLYS	Display-Pointer-Move Instructions	5
POLYV	Display-Pointer-Move Instructions	5
PPAN	Display-Control Instructions	8

Table 2-1. Ω500 Instructions Listed Alphabetically

INSTRUCTION	CHAPTER TITLE	CHAPTER NUMBER
RASTOP	Drawing-Control Instructions	7
RDMASK	Drawing-Control Instructions	7
READ CONF	Utility Instructions	10
RECT1	Drawing Instructions	6
RECT2	Drawing Instructions	6
RDR	Data-Transfer Instructions	9
RLFILL	Drawing Instructions	6
RMOVP1	Display-Pointer-Move Instructions	5
RMOVP2	Display-Pointer-Move Instructions	5
RPIXEL	Data-Transfer Instruction	9
SELRES	Folded-Mode Instructions	11
SET COLOR	Drawing-Control Instructions	7
SETCORN	Drawing-Control Instructions	7
SETCSZ	Drawing-Control Instructions	7
SIG READ	Utility Instructions	10
SYNCH	Utility Instructions	10
SZCUR	Display-Control Instructions	8
WPIXEL	Data-Transfer Instructions	9
WRMASK	Drawing-Control Instructions	7
WRR	Data-Transfer Instructions	9
XBARLD	Folded-Mode Instructions	11
XDRAW	Drawing Instructions	6

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.

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

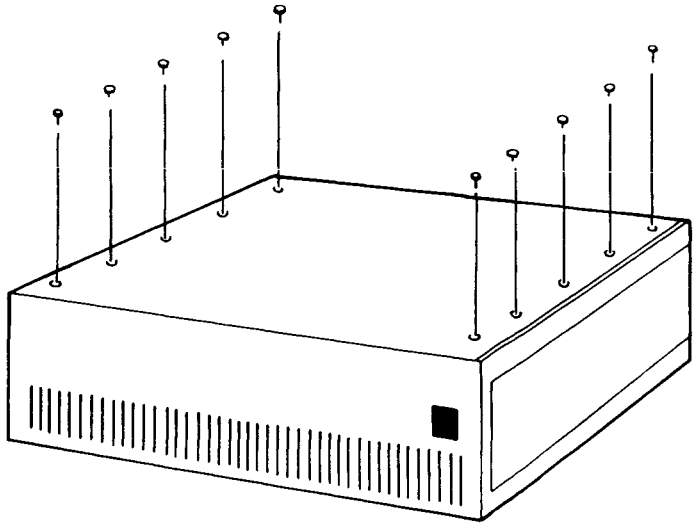


Figure 3-1. Removing the Top Cover

3) Remove the GRAFIN and host interface boards as shown in Figure 3-2.

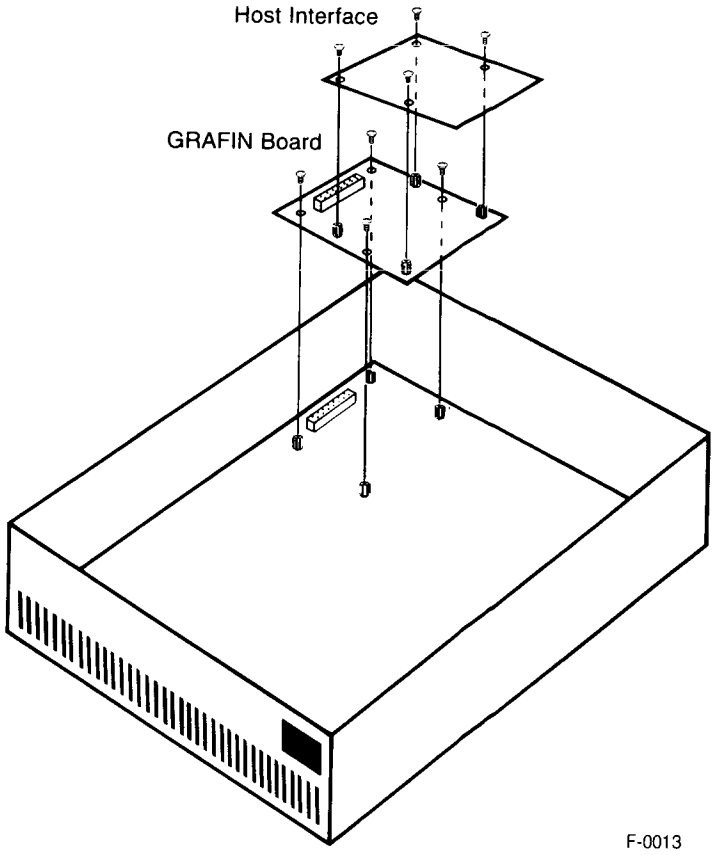


Figure 3-2. Removing the Host and GRAFIN Interface Boards

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.

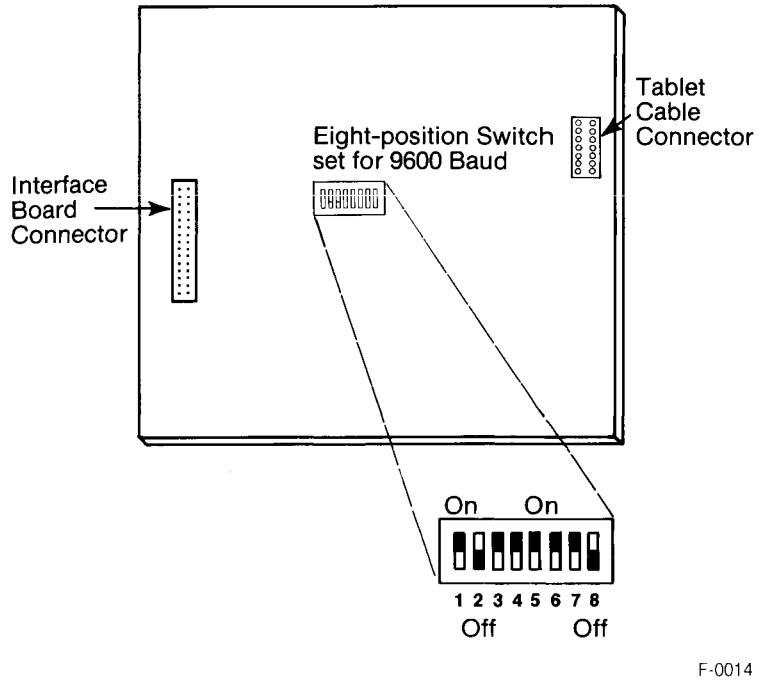


Figure 3-3. Locating the Data Transfer Switch on the GRAFIN Board

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

Table 3-1. Data Rate Selection				
BAUD	SWITCH SETTINGS			
	4	3	2	1
unused	1	1	1	1
19.2K	1	1	1	0
9600	1	1	0	1
7200	1	1	0	0
4800	1	0	1	1
3600	1	0	1	0
2400	1	0	0	1
1800	1	0	0	0
1200	0	1	1	1
600	0	1	1	0
300	0	1	0	1
150	0	1	0	0
134.5	0	0	1	1
110	0	0	1	0
75	0	0	0	1
50	0	0	0	0

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.)

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 Ω500, 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 Ω500. 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 Ω500 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 Ω500 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.

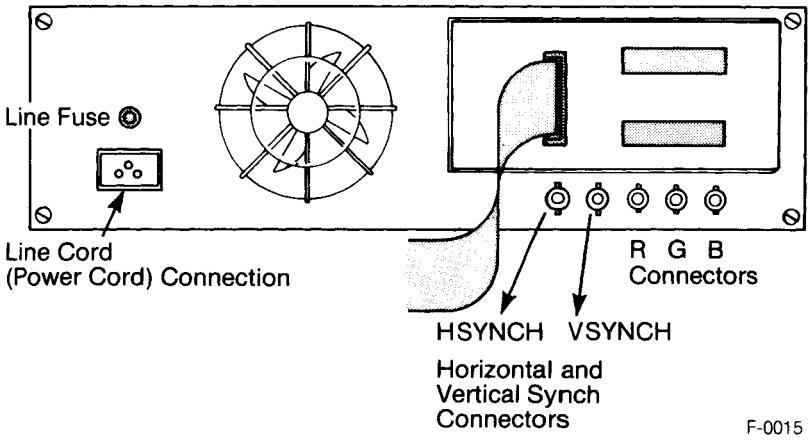


Figure 3-4. Connecting the Monitor to the Back Panel

IMPORTANT

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

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 Ω500 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

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 Ω500 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 Ω530 can operate in two resolution modes: normal or folded. The last chapter describes the instructions the apply only to the Ω530 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**
- **RMOVP1 - Moves pointer P1 to a relative coordinate**
- **RMOVP2 - Moves pointer P2 to a relative coordinate**

MOVP1

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 MOVP1 instruction moves P1 to a point specified in absolute coordinate values. The four bytes immediately following the opcode define the new location for P1.

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

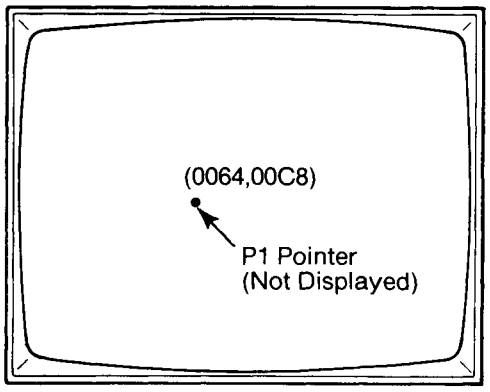


Figure 5-1. Moving the P1 Pointer

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.

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 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

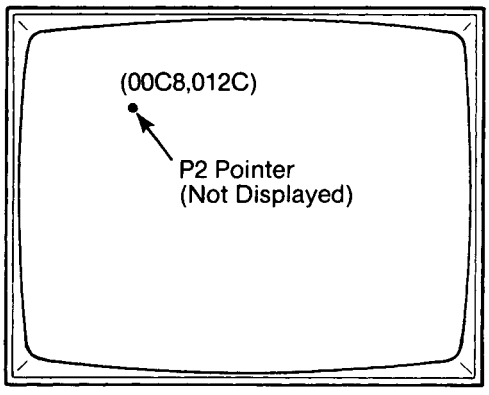


Figure 5-2. Moving the P2 Pointer

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

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

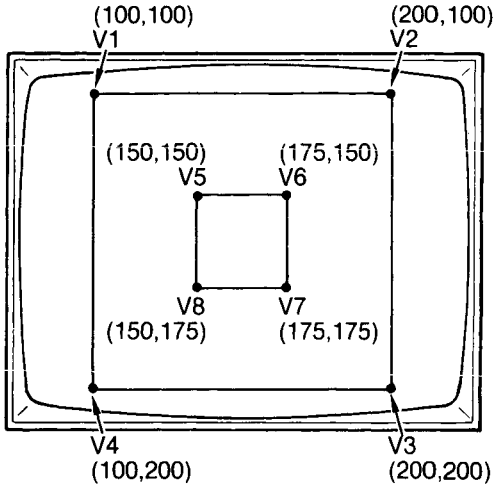


Figure 5-3. Nested Polygons Created with POLYV

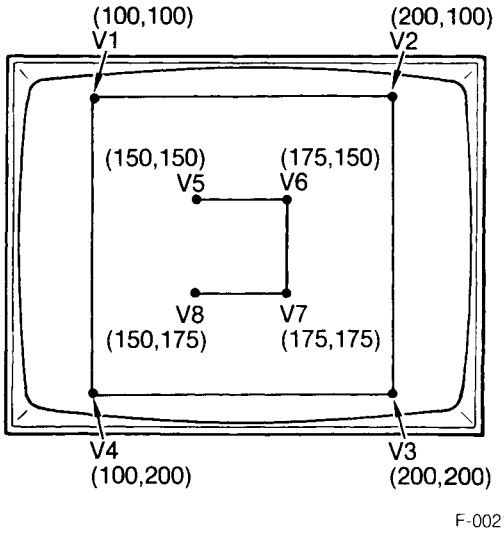


Figure 5-4. Nested Polygons Created with POLYV and POLYM

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 POLYF. This primitive is useful for clipped polygons when clipped regions align with viewport boundaries.

Range

The allowable range for coordinate data is 0 to 1279t for x and 0 to 1023t 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 100,100t (POLYV)
57 C8 00 64 00  Define V2 at 200,100t
45 C8 00 C8 00  Move pointer to V3, 200,200t (POLYM)
45 64 00 C8 00  Move pointer to V4, 100,200t
66              Outline the polygon (POLYO)

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

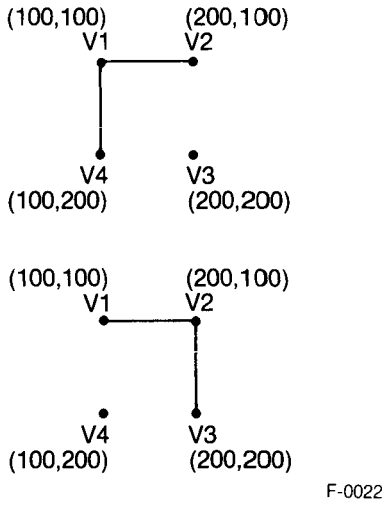


Figure 5-5. Creating Two Open Polygons Using POLYM

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 POLY0 and POLYF instructions in Chapter 6.

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 POLYF instructions display the figure.

POLYV

Adds a vertex to a polygon

Instruction Format

HEX: 57 lox hix loy hiy

DECIMAL: 87 x y

ASCII: W 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

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.

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 C8 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)

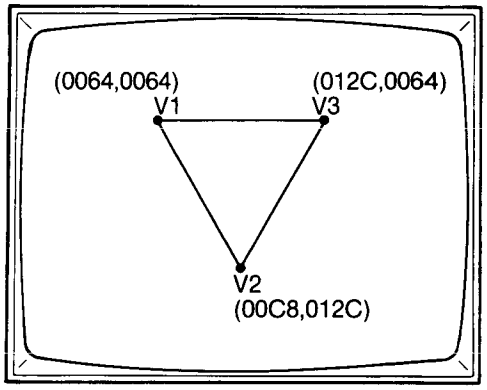


Figure 5-6. Adding Vertices with POLYV

RMOVP1

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

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 -2048t to +2047t. The values are assigned in 12-bit 2's complement. Following a relative move, x must be between 0 and 1279t, and y must be between 0 and 1023t.

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
```

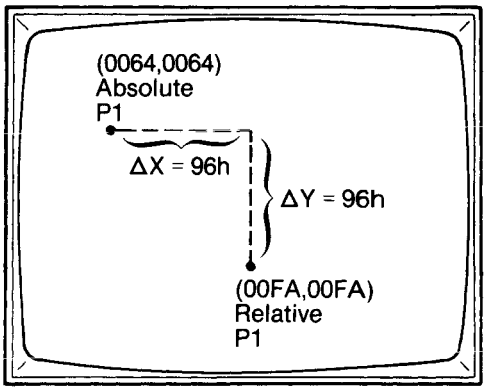


Figure 5-7. Moving P1 to a Relative Location

RMOVP2

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

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 -2048t to +2047t. The values are assigned in 12-bit 2's complement. Following a relative move, x must be between 0 and 1279t, and y must be between 0 and 1023t.

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
```

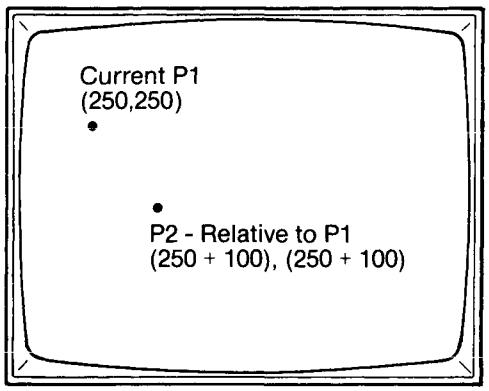


Figure 5-8. Moving P2 to a Relative Location

CHAPTER 6

DRAWING INSTRUCTIONS

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 varicty 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**

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.

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

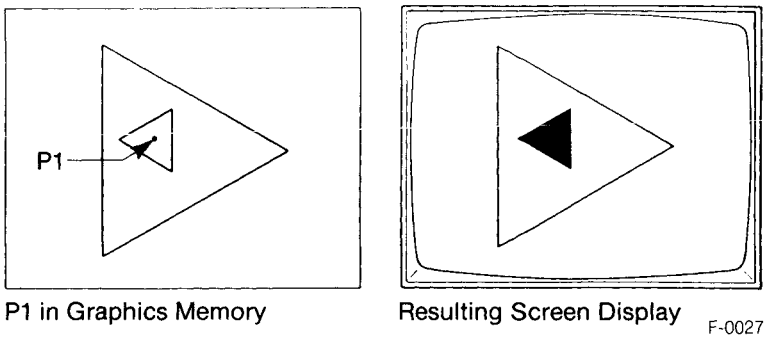


Figure 6-1. P1 Set to a Different Color than the Inner Boundary

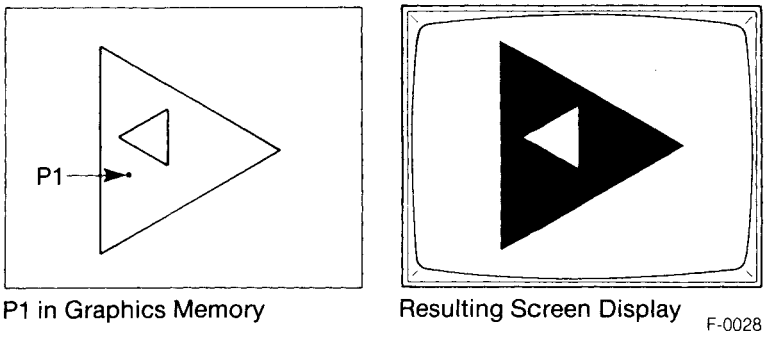


Fig. 6-2. P1 Set Between Boundaries of Two Different Colors

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 255t.

Special Considerations

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

- 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
```

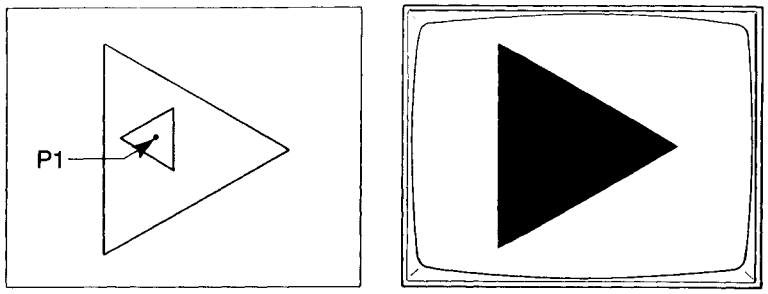


Figure 6-3. Filling to the Outer Boundary with AFILL2

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.

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 2047t

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.

Range

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

Special Considerations

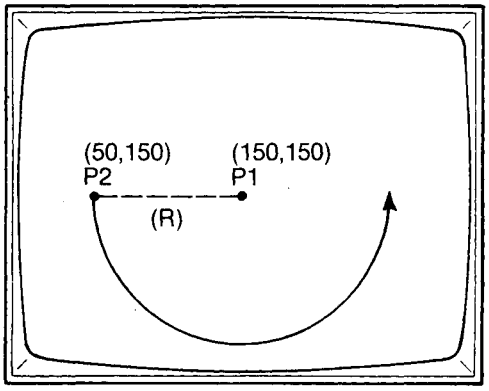
P2 moves to the end point of the arc after it is drawn to easily continue drawing longer arcs or circles.

Example

This example, listed in hexadecimal, sets the center of the arc (P1) at 150, 150t. Then it sets the radius (P2) at 50, 150t 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



F-0030

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

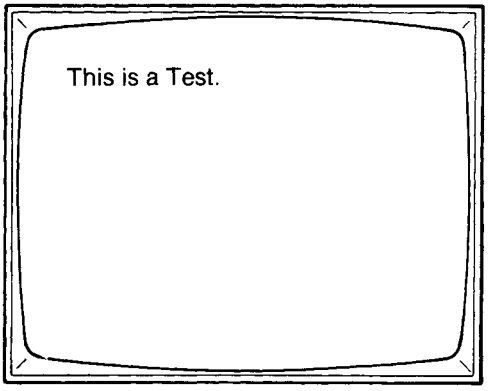
- Characters may be redefined by using the LDFONT instruction.

- 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.)

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



F-0031

Figure 6-5. Typing Characters on the Screen

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.

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.

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.

Example

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

4E 00	Selects black
60	Blank the screen to thc black
52 64 00 64 00	Set P1 to 100, 100t
53 C8 00 2C 01	Set P2 to 200, 300t
72	Draw a complemented line

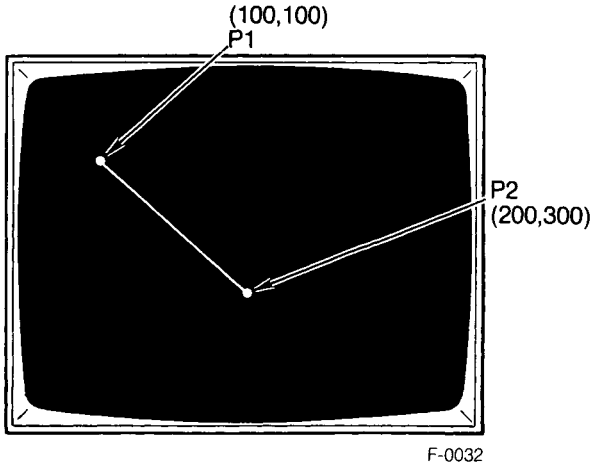


Figure 6-6. Drawing a Line in the Complement Color

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

P1 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 P1 to 100,100t
53 C8 00 2C 01	Sets P2 to 200,300t
61	Draws a vector from 100,100t to 200,300t

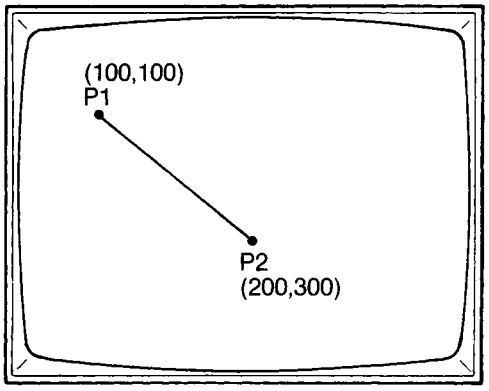


Figure 6-7. Drawing a Line

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

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 100,300t
53 C8 00 64 00	Set P2 to 200,100t
65	FLASH-fill the rectangle

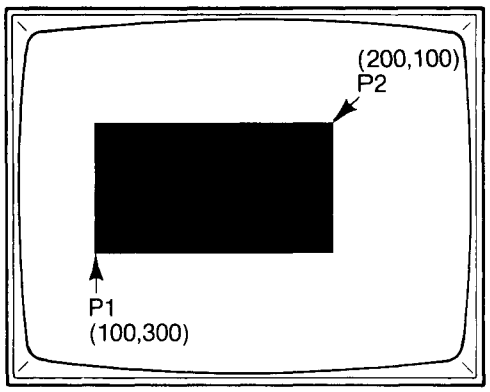


Figure 6-8. FLASH-filling a Rectangle

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:

$$2(V) + 9(AE_{max}) \leq 4000$$

Where:

V	Total number of vertices in the polygon including all sub-polygons
AE_{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

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.

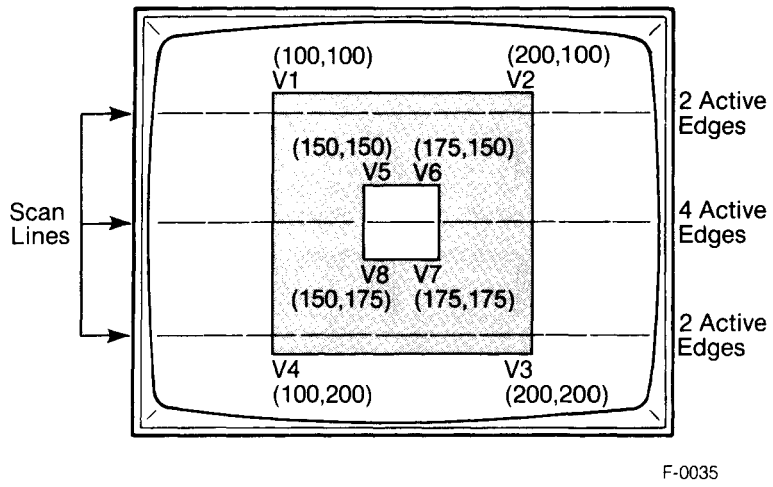


Figure 6-9. Filling a polygon

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

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 ploygon 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)
```

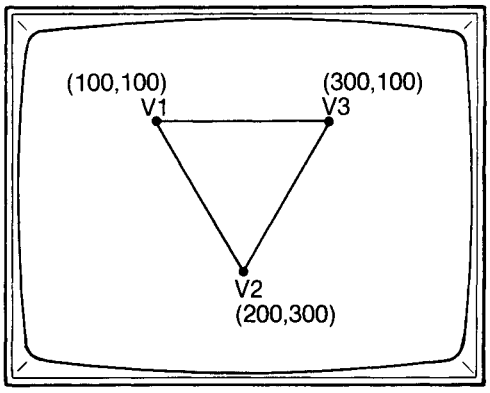


Figure 6-10. Outlining a Polygon

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

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
```

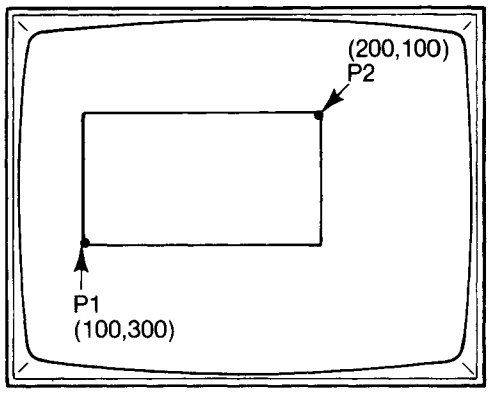


Figure 6-11. Drawing a Rectangle Outline

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

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 C8 00 64 00          Set P2 to 200,100t
64
```

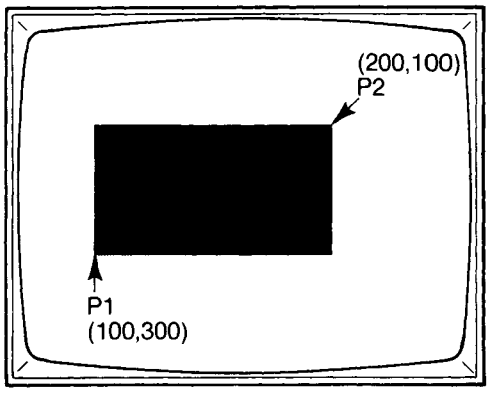


Figure 6-12. Filling a Rectangle

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 Ω500 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 Ω500 System.

Range

The pixel count range is from 1 to 1280t.

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
```

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.

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**
- **SET COLOR - Selects the drawing color**
- **SETCORN - Sets the character orientation**
- **SETCSZ - Sets the character size**
- **WRMASK - Sets the write mask**

CSPACE

Sets the character spacing

Instruction Format

HEX: 48 loΔx hiΔx loΔy hiΔy

DECIMAL: 72 Δx Δy

ASCII: H loΔx hiΔx loΔy hiΔy

Input Arguments

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

hiΔx 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

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

hiΔy 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

$-1279t < \Delta x < 1279t$
$-1023t < \Delta y < 1023t$ (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).

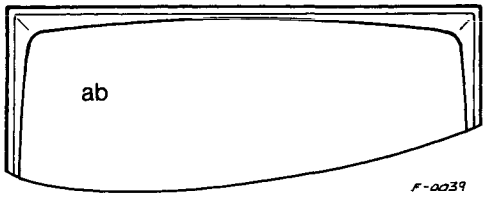
The CSPACE instruction following a FSIZE, SETCORN or SETCSZ instruction overrides the default spacing. Note that the CSPACE must account for changes in SETCSZ. (See example 5.)

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.

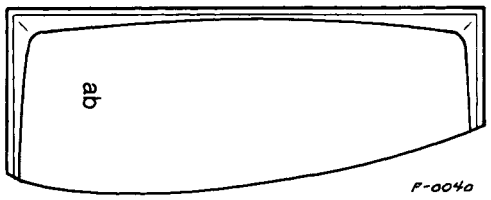
(1)

59 00	Sets the character orientation to 0
6B ab (esc)	(SETCORN)
	Types 'ab' to the screen (CHAR)
	CSPACE automatically defaults to 8x,0y



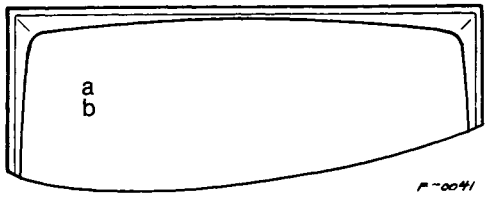
(2)

59 03	Sets the character orientation to 3
6B ab (esc)	(SETCORN)
	Types 'ab' to the screen (CHAR)
	CSPACE automatically defaults to 0x,8y



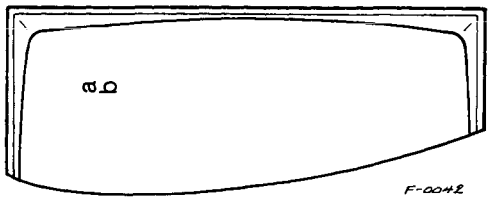
(3)

59 00	Sets the character orientation to 0
48 00 10	(SETCORN)
6B ab (esc)	Sets CSPACE to 0x,16y
	Types 'ab' to the screen



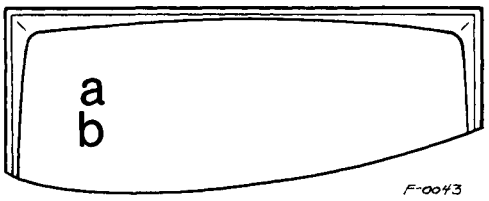
(4)

59 05	Sets the character orientation to 5
48 F0 FF 00	(SETCORN)
6B ab (esc)	Sets CSPACE to -16x,0y
	Types 'ab' to the screen



(5)

59 01	Sets the character orientation to 1
	(SETCORN)
58 00 01	Sets the character size to 1
	(twice as big as normal)
48 00 20 00	Sets CSPACE to 0x,32y
6B ab (esc)	Types 'ab' to the screen



FSIZE

Sets the font size

Instruction Format

HEX: 49 b_1 b_2

DECIMAL: 73 b_1 b_2

ASCII: I b_1 b_2

Input Arguments

b_1 an 8-bit number representing font width

b_2 an 8-bit number representing font height

Outputs

None

Description

This instruction invokes a viewport 8-pixels wide by b_2-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, b_1 and b_2 specify character width and height.

Range

b_1: 1 to 8t
b_2: 1 to 16t

LDR0C

Loads the raster-op comparator array

Instruction Format

HEX: 42 SC_1...SC_n

DECIMAL: 66 SC_1...SC_n

ASCII: B SC_1...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

LDR0C 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 LDR0C 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 LDR0C parameters are defined as follows:

0h	80h
Exits the LDROC instruction.	Exits the LDROC instruction.

1h	81h
Resets ROC array to zeroes (no colors allowed to write).	Sets ROC array to ones (all colors allowed to write).

Use this command to ensure uniform array state before loading.

2h	82h
Resets a single location in the ROC array.	Sets a single location in the ROC array.

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

3h **83h**

Either command block-loads all 256 locations in the ROC array. 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	84h
Clears a range of locations.	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
      Begin
      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
E0          Set location E0 to 0 (disable writes)
00          Exit
```

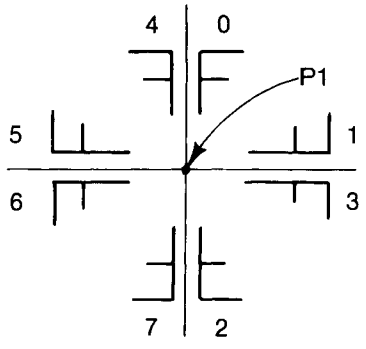


Figure 7-1. Loading the Raster-Op Comparator Array

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.

Lower	Upper	Mask	Conditions Allowing Writes
0	0	1	Bit-plane 0=0 (even-numbered planes)
1	1	1	Bit-plane 0=1 (odd-numbered planes)
0	0	2	Bit-plane 1=0 (even-numbered planes)
2	2	2	Bit-plane 1=1 (odd-numbered planes)
0	0	4	Bit-plane 0=0 (even-numbered planes)
4	4	4	Bit-plane 0=1 (odd-numbered planes)
3	3	3	Bit-plane 0=1 AND bit-plane 1=1
1	1	3	Bit-plane 0=1 AND bit-plane 1=0
2	2	3	Bit-plane 0=0 AND bit-plane 1=1
1	2	3	Bit-plane 0=1 XOR bit-plane 1=0
1	3	3	Bit-plane 0=1 OR bit-plane 1=1

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.

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

Table 7-1
Drawing Modes

Mode	Pattern	Background
0 (000)	data, all planes	zero, all planes
1 (001)	data, all planes	no change
2 (010)	data, selected planes	zero, all planes
3 (011)	data, selected planes	no change
4 (100)	data, all planes	zero, selected planes
5 (101)	same as 4	same as 4
6 (110)	data, selected planes	zero, selected planes
7 (111)	same as 6	same as 6

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.

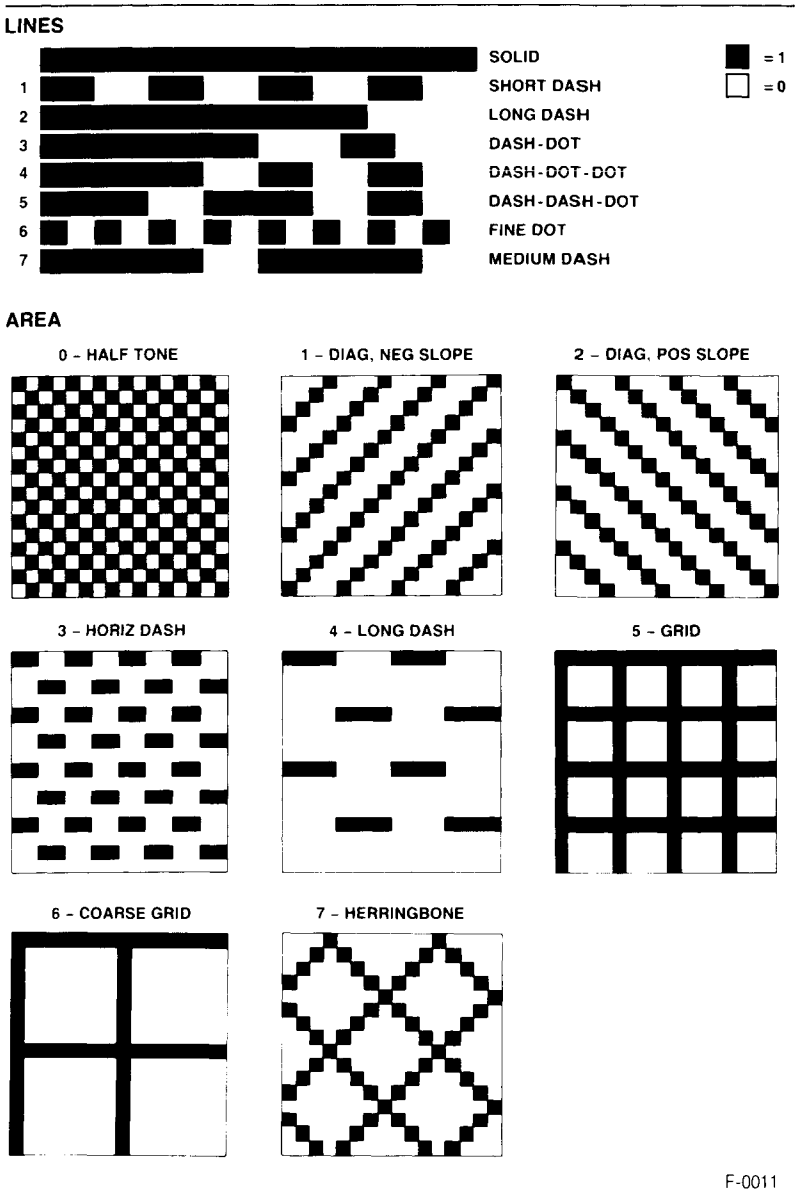


Figure 7-2. Default Drawing and Filling Patterns

RASTOP

Specifies the raster-op

Instruction Format

HEX: 43 b_1 b_2

DECIMAL: 67 b_1 b_2

ASCII: C b_1 b_2

Input Arguments

b_1 specifies data flow

b_2 specifies ALU operation

Outputs

None

Description

RASTOP specifies the raster operation used for subsequent drawing instructions. Two bytes following the opcode specify b_1 and b_2.

Pixel-writing primatives 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 7-2
Data Flow Control

b_1	Description	Function
0	Draw-Normal	D = C
1	Draw-ALU Enabled	D = D op C
2	Draw-Cndtnl	If (D), D = C
3	Draw-ALU Cndtnl	If (D), D = D op C
0	PIXBLT-Normal	D = S
1	PIXBLT-Cndtnl	If (D), D = S
2	PIXBLT-Cndtnl	If (S), D = S
3	PIXBLT-ALU	D = C op S
4	PIXBLT-ALU, Cndtnl	If (D), D = S op C
5	PIXBLT-ALU, Cndtnl	If (S), D = C op S
6	PIXBLT-ALU, Cndtnl	If (S), D = C
7	PIXBLT-ALU	D = S op D
8	PIXBLT-ALU, Cndtnl	If (D), D = S op D
9	PIXBLT-ALU, Cndtnl	If (S), D = S op D

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

S represents the source pixel (used in blt 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 Op1 and Op2. For drawing primitives, the ALU always usesn **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 Op1 and Op2 are as shown below in Table 7-3:

Table 7-3
Op1 and Op2 Assignments

Primitive Type	Op1	Op2
Draw (1,3)	C	D
PIXBLT (3,5)	C	S
PIXBLT (4)	S	C
PIXBLT (7-9)	S	D

The ALU functions are shown in Table 7-4:

Table 7-4
ALU Functions

b_2	ALU Function
0	zeroes
1	Op1 - Op2 - 1
2	Op1 - Op2
3	Op2 - Op1 - 1
4	Op2 - Op1
5	Op1 + Op2
6	Op1 + Op2 + 1
7	Op1 XOR Op2
8	Op1 Or Op2
9	Op1 AND Op2
10	ones

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.

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

b_1	b_2	Function
0	x	Normal drawing mode
1	7	XOR mode for vectors, arcs, polygons, rectangles
1	9	And mode for vectors, arcs, polygons, rectangles
2	x	Conditional write for vectors, arcs, polygons, rectangles

x represents a "don't care"

Table 7-6
PIXBLT Raster-Ops

b_1	b_2	Function
0	x	D = S
3	8	D = C OR S
3	7	D = C EXOR S
3	4	D = S - C
3	2	D = C - S
7	9	D = S AND D
7	7	D = S EXOR D
5	9	If (S), D = C AND S
5	7	If (S), D = C EXOR S
8	7	If (D), D = S EXOR D
9	7	If (S), D = S EXOR D

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 255t, 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.)

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.)

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.

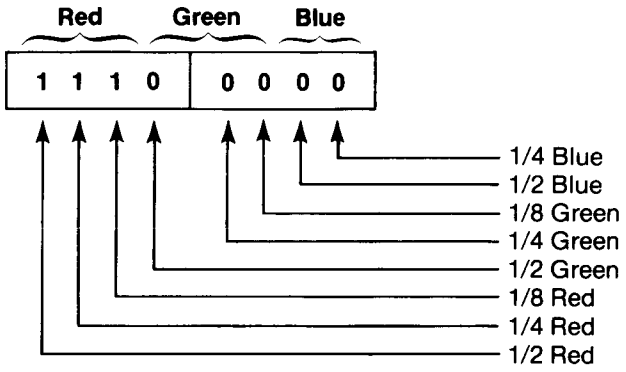


Figure 7-3. The SET COLOR Index to the Default Color Map

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

Orientation	Bits	Function
0	000	Normal
1	001	Mirror
2	010	Rotate y
3	011	Mirror and rotate y
4	100	Rotate x
5	101	Mirror and rotate x
6	110	Rotate x and y
7	111	Rotate and mirror x and y

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.

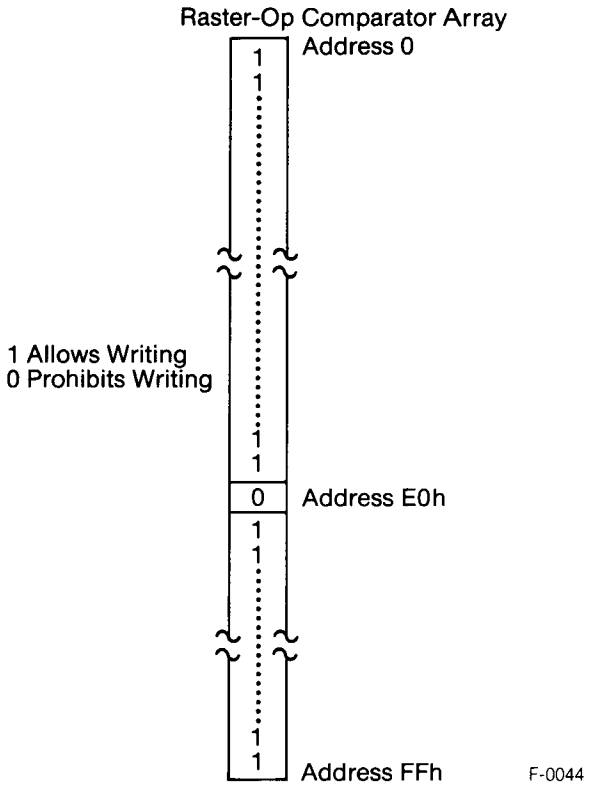


Figure 7-4. Selecting the Orientation of Text

SETCSZ

Sets the character size

Instruction Format

HEX: 58 b_1 b_2

DECIMAL: 88 b_1 b_2

ASCII: X b_1 b_2

Input Arguments

b_1	the width scale-factor (x axis magnification)
b_2	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 255t (256X).

WRMASK

Sets the write mask

Instruction Format

HEX: 4F b

DECIMAL: 79 b

ASCII: O b

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 Ω500 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.

Range

The range is 0 through 255t, 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**

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.

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:

Blank Flag	Effective Read Mask	Effective Write Mask
0	Mask	Mask
1	Mask & F0h	Mask & 0Fh
2	Mask & 0Fh	Mask & F0h
3	0	Mask

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).

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 255t into 0 through 127.

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.

Range

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

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 Ω500 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.

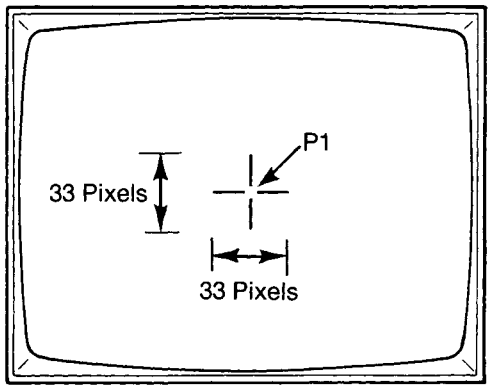


Figure 8-1. Displaying the Cursor

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 120,112t
5B	Pans display memory to P1

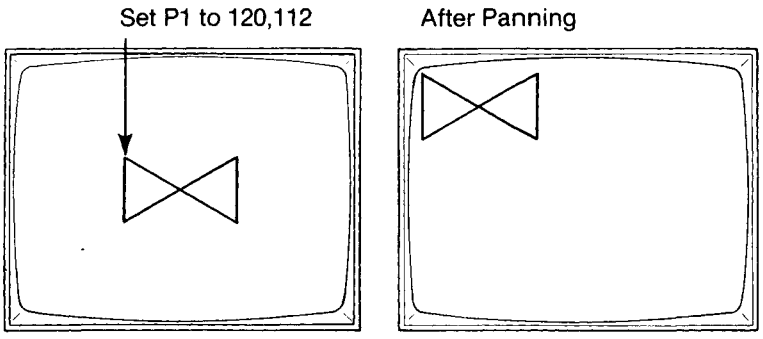


Figure 8-2. Panning to a New Display-Memory Origin

SZCUR

Changes the size of the cursor

Instruction Format

HEX: 47 w_1 w_2

DECIMAL: 71 w_1 w_2

ASCII: G w_1 w_2

Input Arguments

w_1 two bytes that control cursor width; actual cursor width is $2w_1+1$. 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.

w_2 two bytes that control cursor height; actual cursor height is $2w_2+1$. 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.

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 Ω500 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.

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

GRAFIN

Selects the graphic input mode

Instruction Format

HEX: 4A f

DECIMAL: 74 f

ASCII: J f

Input Arguments

f the GRAFIN mode-select byte:

0	Software INIT
1	Local Cursor Control
2	Set Transparent Mode
3	Set Offset and Scale Factors
4	Set Delimiter
5	Sample Current Screen x,y Coordinates
6	Sample Current Tablet x,y Coordinates
7	Set Mode Register

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 Ω500 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 9-1
Ω500 STATUS MESSAGE FORMAT

Byte	Bit							
	7	6	5	4	3	2	1	0
1	0	1	pb8	pb4	pb2	pb1	kp*	0
2	0	0	x5	x4	x3	x2	x1	x0
3	0	0	0*	0*	x9	x8	x7	x6
4	0	0	y5	y4	y3	y2	y1	y0
5	0	0	0*	0*	y9	y8	y7	y6

*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

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 Ω500 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:

Bit	IF = 0(default)	IF = 1
0	Wraparound	Clip to screen boundary
1	Level buttons	Edge buttons
2	Button xmit	Button not xmit
3	Leading edge report only	Both edges report

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 Ω500 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 Ω500. In Transparent Mode, however, the Ω500 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 Scale * (tablet data + 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.

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

Summagraphics: 4A03 0000 D900 DEF9 5AFF

PIXBLT

Transfers blocks of pixels

Instruction Format

HEX: 70 w_1 w_2 b

DECIMAL: 112 w_1 w_2 b

ASCII: p w_1 w_2 b

Input Arguments

w_1 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

w_2 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

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

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 destination.

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 (w_1) 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.

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 Ω500 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.

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)
Pixel count = (5-2+1)*(7-2+1) = 4*6 = 24 pixels.

RPIXEL

Reads a pixel

Instruction Format

HEX: 6C

DECIMAL: 108

ASCII: l

Input Arguments

None

Outputs

One byte, for the pixel color at P1

Description

The RPIXEL instruction causes the Ω500 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 255t.

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.

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

WPIXEL

Writes a pixel

Instruction Format

HEX: 6D

DECIMAL: 109

ASCII: m

Input Arguments

None

Outputs

None

Description

When the Ω500 System receives a WPIXEL 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.

- 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

Δx times Δy 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 Ω500 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 (Δx times Δy). Δx and Δy 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.

Range

Pixel values range from 0 to 255t.

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 (1279,1023t). 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 dianostic information.

- **DFAULT - 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**

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 stipples, raster-op, and current drawing pattern.

Range

0 through 4t

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 defintion
: The stipples are restored to the standard Omega stipple patterns. See the LDPAT instruction decribed 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.

INIT

Initializes the system

Instruction Format

HEX: 5E

DECIMAL: 94

ASCII: ↑

Input Arguments

None

Outputs

None

Description

When the Ω500 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 Ω500 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

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

b_1 — The lower 8 bits of the drawing parameter.

b_2 — 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.

Special Considerations

None

Table 10-1
INQ Parameters

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

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 [(STOP-START+1) > 0], then that number of data bytes follow the CDB length byte.

A negative result ends the LDFONT instruction. For example,

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':

```
LDFONT      COMMAND

3Eh
6Fh         start with 'o'
A5h         define six rows (5,6,7,8,9,and Ah)
3Ch
42h
42h
42h
42h
3Ch
A2h         define nine rows for 'p' (2,3,...,9,Ah)
40h
40h
40h
5Ch
62h
42h
42h
42h
7Ch
1Bh         exit
```

LDPAT

Specifies a new pattern

Instruction Format

HEX: 40 b_1 {[2*b_2] or [32*b_2]}

DECIMAL: 64 b_1 {[2*b_2] or [32*b_2]}

ASCII: @ b_1 {[2*b_2] or [32*b_2]}

Input Arguments

b_1 this byte specifies which pattern to change.

bits 0-2	lower three pattern select lines
bit 3	line (1) or area (0)
bit 4	upper pattern select bit

b_2 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 can not be redefined.) The bytes following the opcode decribe these line or area patterns.

Lines. Three bytes define lines. The first parameter (b_1) specifies the pattern number to alter and selects lines instead of areas. The second parameter (b_2) 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,

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

Areas. Thirty-three bytes define areas. The first parameter (b_1) specifies the pattern number to alter and selects areas instead of lines. The second parameter (b_2) 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

b_1: 0 through 31t

Special Considerations

Line style 0 cannot be redefined.

READ CONF

Reads and returns the system configuration

Instruction Format

HEX: 5D

DECIMAL: 93

ASCII:]

Input Arguments

None

Outputs

b_1	hardware configuration
b_2	microcode version number

Description

When the Ω530 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 Ω530 supports two resolutions: 1280 by 1024t pixels or 640 by 512t 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

SIG READ

Reads the signature-analyzer register

Instruction Format

HEX: 5C

DECIMAL: 92

ASCII: \

Input Arguments

None

Outputs

Two signature bytes

Description

When the Ω500 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.

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 255t.

When the Ω500 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 Ω500 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.

Range

The range for the wait count is 0 through 255t.

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 Ω530 Display Controller.

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

BANK

Selects the memory bank

Instruction Format

HEX: 3D b

DECIMAL: 61 b

ASCII: = b

Input Arguments

b — the bank-select byte. On the Ω530, 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.

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 8t

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 INIT 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).

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.

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.

- 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: 3C b

DECIMAL: 60 b

ASCII: < b

Input Arguments

b — the memory-resolution select byte. The least-significant bit of this byte selects the memory resolution:

0 Folded mode (640 * 512 * 32)
1 High-resolution mode (1280 * 1024 * 8)

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

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.

Range

0 through 255t

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:

00 10 01 00 = 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:

00 00 00 00 = 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:

00 01 10 01 = 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:

10 01 01 01 = 95h

APPENDIX A

Table A-1. Code Conversion Chart

ASCII	DECIMAL	OCTAL	HEX
NUL	0	0	0
SOH	1	1	1
STX	2	2	2
ETX	3	3	3
EOT	4	4	4
ENQ	5	5	5
ACK	6	6	6
BEL	7	7	7
BS	8	10	8
HT	9	11	9
LF	10	12	A
VT	11	13	B
FF	12	14	C
CR	13	15	D
SO	14	16	E
S1	15	17	F
DLE	16	20	10
DC1	17	21	11
DC2	18	22	12
DC3	19	23	13
DC4	20	24	14
NAK	21	25	15
SYN	22	26	16
ETB	23	27	17
CAN	24	30	18
EM	25	31	19
SUB	26	32	1A
ESC	27	33	1B
FS	28	34	1C
GS	29	35	1D
RS	30	36	1E
US	31	37	1F

Table A-1. Code Conversion Chart

ASCII	DECIMAL	OCTAL	HEX
SP	32	40	20
!	33	41	21
"	34	42	22
#	35	43	23
$	36	44	24
%	37	45	25
&	38	46	26
'	39	47	27
(	40	50	28
)	41	51	29
*	42	52	2A
+	43	53	2B
,	44	54	2C
-	45	55	2D
/	47	57	2F
0	48	60	30
1	49	61	31
2	50	62	32
3	51	63	33
4	52	64	34
5	53	65	35
6	54	66	36
7	55	67	37
8	56	70	38
9	57	71	39
:	58	72	3A
;	59	73	3B
<	60	74	3C
=	61	75	3D
>	62	76	3E
?	63	77	3F
@	64	100	40
A	65	101	41
B	66	102	42
C	67	103	43
D	68	104	44

Table A-1. Code Conversion Chart

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

Table A-1. Code Conversion Chart

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