Ω400/DC Display Controller

and

Ω400/GS Graphics Subsystem

Operator's Manual

(004-01085-01)

Table of Contents

Section 5 — Drawing Instructions (cont.)

Section 6 — Drawing Control Instructions

Section 7 — Display Control Instructions

Section 8 — Data Transfer Instructions

Section 9 — Utility Instructions

Section 10 — Installation Instructions

Appendix A

Section 1

INTRODUCTION TO THE Ω400

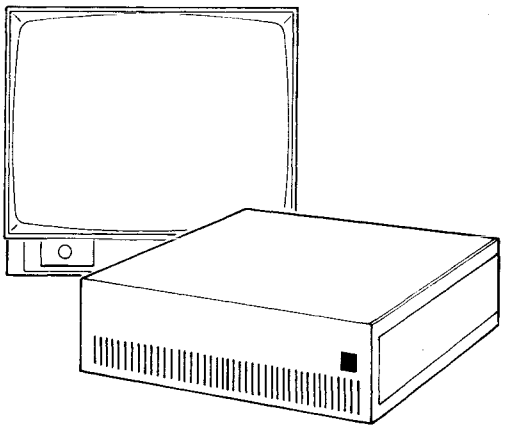


Fig. 1-1. The Ω400 Display Controller and Monitor

Overview

The Ω400/DC Display Controller and the Ω400/GS Graphics Subsystem are high resolution, high performance color graphics devices. Both are illustrated in Fig. 1-1. The Display Controller acts as an interface between a host computer system and a high resolution color display monitor, translating instructions into the timing and color signals that the display needs. The Graphics Subsystem includes the Display Controller and a high resolution color monitor.

The Display Controller's graphics memory provides a resolution of 1024 X 1024 points which, coupled with high display resolution, provides a highly-detailed display. There are two models available. The Ω420 uses four bit planes to yield 16 displayable colors from the color palette. The Ω440 is an eight bit plane version, allowing 256 colors to be placed in the display color map at one time. In both models, colors are dynamically selectable from a palette of 16.7 million color choices.

The display memory can be mapped onto the display in two possible ways, depending on the resolution of the monitor. The monitor can operate in 60 Hz mode or 33 Hz mode; the Display Controller must be set to match the monitor's mode, as described in the Installation Instructions in this manual. In 60 Hz mode, the display is non-interlaced; resolution is 736 pixels in the x axis by 552 pixels in the y axis. In 33 Hz mode, the display is interlaced; display resolution is 1024 pixels (x) by 768 pixels (y).

The Ω400 features a high-speed bipolar bit-slice processor that delivers display writing speed four to ten times faster than available with other systems. This speed also enables special functions such as FLASH-fill and CLEAR which, coupled with other special hardware functions such as Pan and Zoom, provide significant performance enhancements for the user. The faster display speed means that the system is easier and friendlier for the user to work with. There is a minimal wait for graphics changes to appear on the display.

All display electronics in the Ω400 are located on a single printed circuit board, increasing system reliability by eliminating multiple edge connectors. Reliability and system serviceability are also greatly improved by built-in self-test features and a unique on-board signature analyzer. Graphics Tablet input is supported by the GRAFIN functions; a serial input connector for the tablet is located on the Ω400 back panel.

The Ω400 Systems are supported by the optional AXIA Graphics Package, available to run on the host computer system. The AXIA Graphics Package is described in a separate manual.

Installing the Ω400 Systems

Installation of the Ω400 Systems, either the Display Controller or the Graphics Subsystem, is not complicated. All of the necessary interconnects and option selections are described in the Installation Instructions at the back of this manual. Be sure to read and follow the instructions before operating the equipment.

Section 2

INTRODUCTION TO Ω400 GRAPHICS

This section of the manual describes elements of computer graphics as they are applied to the Ω400 Display Controllers and Graphics Subsystems, and as the terms are used in this manual. For those new to computer graphics, this section will provide a brief introduction to the operations and instructions described later in this manual. For those with previous computer graphics experience, this discussion will clarify points of application specific to the Ω400 units.

The rest of this section will provide an overview of the following elements of computer graphics:

- **Display Units**
- **Coordinates and Display Addresses**
- **The Relationship Between Graphic Memory and the Screen**
- **Lines, Rectangles, Arcs, and Polygons**
- **The Pointers**
- **Introduction to Zoom and Pan**

Display Units

In the Ω400 Systems, the basic element of the graphic display is the pixel, or picture element, which represents 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 1024 (x) by 1024 (y), or over one million addressable points. All display images are created by writing combinations of adjoining pixels (see Fig. 2-1).

Display resolution (the total number of displayable points in one display) is strap-selectable within the Ω400 Systems. The strap is factory set to the requested resolution, matching the monitor selection. Strap selections and monitor adjustments are described in the Installation Instructions of this manual. The associated display resolutions are as follows:

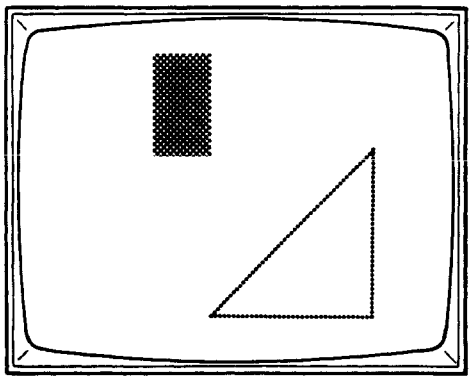


Fig. 2-1. Pixel Combinations Create Images

In the 33 Hz position, display resolution is 1024 points in the x axis (horizontal) by 768 points in the y axis (vertical), corresponding to the 4:3 physical aspect ratio of the display (see Fig. 2-2).

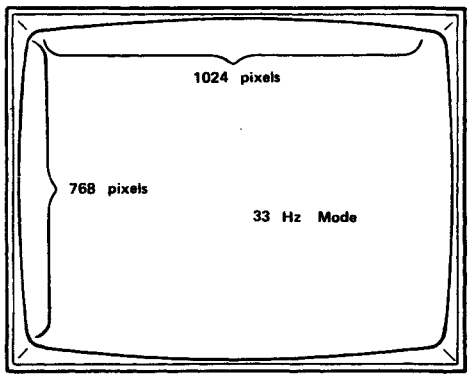


Fig. 2-2. 33 Hz Display Mode Resolution

In the 60 Hz position, display resolution is 736 pixels in the x axis by 552 pixels in the y axis, still maintaining the 4:3 aspect ratio (see Fig. 2-3).

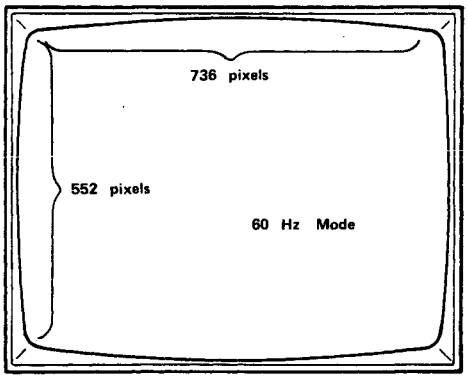


Fig. 2-3. 60 Hz Display Mode Resolution

Any pixel can be displayed in any color available. The Ω420 is a four bit plane Display Controller, with 16.7 million color choices that can be loaded into any of the 16 color map addresses. Any one of the 16 colors in the color map can be selected as the current drawing color, using the SET COLOR instruction.

The Ω440 uses eight bit planes, providing 256 color map addresses that can be loaded from the 16.7 million colors in the palette. Any one of those 256 colors can then be used as the current drawing color.

Coordinate Addresses

The Ω400 devices use coordinate addresses in the range 0 through 1023 in the x axis and 0 through 1023 in the y axis (vertical). Each point in graphic memory is represented by a unique coordinate address, or intersection of each axis; each intersection is one pixel. Each pixel address consists of an x value and a y value representing the point where the x and y axes meet (Fig. 2-4).

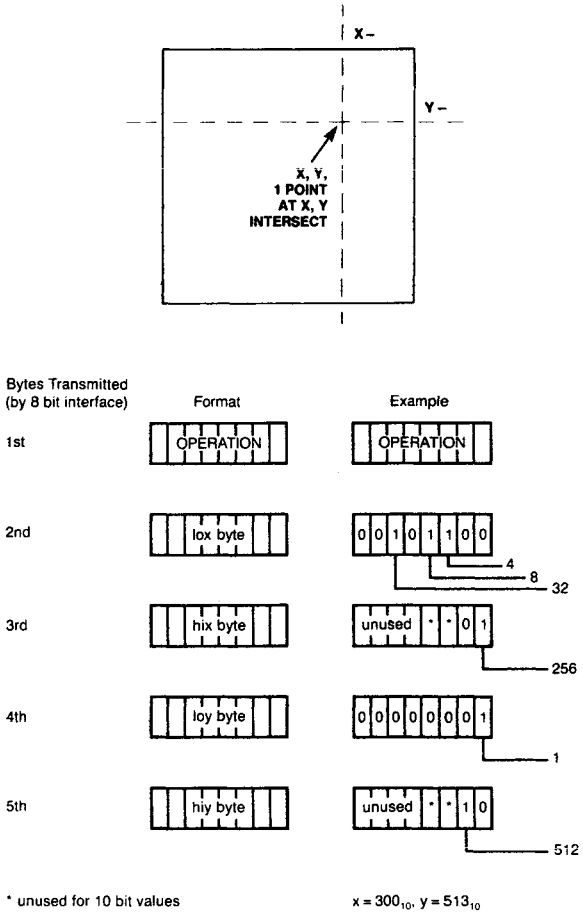


Fig. 2-4. Coordinate Addresses

Ten bits are necessary to represent each x and y coordinate value. A complete address specification consists of four eight-bit bytes: lox, hix, loy, hiy. Eight bits of the x coordinate value are contained in the lox byte; the remaining two bits are in hix byte. The y coordinate value is divided in the same manner between the loy and hiy bytes.

In the coordinate system, the upper left corner of the graphic memory represents (0x,0y). The lower right corner of memory represents (1023x,1023y). Therefore, x address values increase from left to right; y address values increase from the top down. Refer to Fig. 2-5.

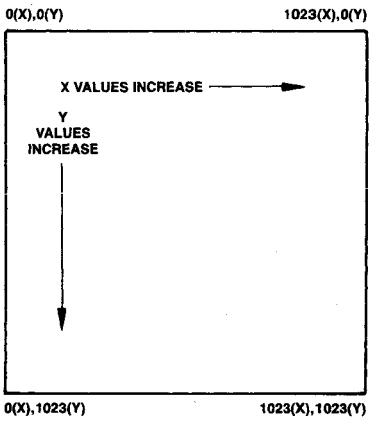


Fig. 2-5. Display Memory Coordinates

In the default state, one pixel in memory space is equivalent to one pixel on the display. Memory and the display use the same addressing scheme. However, the memory space is larger than the display space, in all configurations. This relationship is detailed in the following paragraphs.

Graphic Memory and the Display

In all Ω400 Systems, the display memory resolution is 1024 pixels in each axis. The portion of graphic memory that is mapped to the display depends on the Display Mode strap selection and monitor setting, as well as the Zoom and Pan factors. (Zoom and Pan will be discussed later. For the purposes of this discussion, both functions will be assumed to be in their default states - 1X magnification of the default memory space.)

If the system is operating in 33 Hz mode, the display resolution is 1024 points in the x axis by 768 points in the y axis. This means that the entire x axis of the display memory (0 through 1023) is viewable; the upper 75% of display memory (0 through 767) is mapped to the monitor screen. Refer to Fig. 2-2.

The system can also operate in 60 Hz mode, where display resolution is 736 pixels in the x axis by 552 pixels in the y axis. Memory resolution is unchanged. In default conditions, x values from 0 through 735 are viewable, while y values from 0 to 551 are mapped to the screen. See Fig. 2-3.

Lines, Rectangles, Arcs, and Polygons

Pixels can be combined in many ways to create graphic images. The basic display elements created from the joining of pixels are the line (or vector), the rectangle, the polygon, and the arc. These elements can be described as follows, as they apply to the Ω400:

A line, or vector, is just that: a straight line between two points on the display. A line can be solid. That is, all of the adjacent pixels in the line can be written in the same color. Or a line pattern can be selected, in which some pixels are written in the current color interspersed with spaces, creating various dashed-line patterns.

A rectangle is any four-sided space with ninety-degree corners, including a square, in which the sides are parallel to the x and y axes. Only two corner points are required to define a rectangle. Rectangles can be outlines only, solid fill areas, pattern-filled areas, or filled areas outlined in another color. A number of draw commands are available to create the various rectangle types.

An arc is a curved line. Like the line, it can be a solid line or a dashed line. The arcs created by the Ω400 are circle segments. Circles can be displayed by continuing an arc until the end meets the start point.

The polygon is any multi-sided figure. The polygon is created by defining its vertex points. A polygon can be filled by using the Fill a Polygon instruction, unless it contains concave vertices. These polygons can be filled more slowly with the AFILL1 or AFILL2 instructions, or with host software using the RLFILL instruction.

The Pointers

The pointers are not displayed. They keep track of positions in graphic memory, and are used for positioning display graphics. P1 is the start point for many graphics functions, for instance, as well as the arc center. P1 and P2 are also used for defining rectangles. Pointers are positioned using the Display Pointer Move instructions; these moves can be to absolute positions or to positions that are relative to the present position of a pointer.

Introduction to Zoom and Pan

Zoom and Pan are used to focus on specific portions of graphic memory for a magnified display. Zoom is used to magnify the memory by a specified value for display. In cases other than default, the display memory no longer maintains a 1:1 pixel ratio with the display. Integer Zoom values of 0 - 15 can be specified, corresponding to magnification values of 1X - 16X. Everything in display memory, including text characters, is affected by the Zoom operation.

Pan selects a memory area to be focused upon by the Zoom function, by changing the origin point in memory that is mapped to the upper left corner of the monitor screen.

Section 3

THE INSTRUCTION SET

Overview

The Instruction Set for the Ω400 is divided into six instruction groups, according to the function of the instructions. Each group of instructions is described in one of the following six sections of this manual.

The first group describes the instructions that are used to move the display pointers. The positioning of 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 are those used for drawing actions, creating images on the monitor screen. The Drawing instructions are those used to draw vectors and rectangles, as well as those used to fill objects. These instructions are described in Section 5.

Section 6 describes the Drawing Control instructions. These instructions set the environmental parameters for the Drawing instruction actions. The Display Control instructions include selecting draw color and patterns, as well as setting the character size and orientation.

Section 7 describes special Display Control instructions. These instructions include defining the colors for the color map, as well as selecting the Zoom factor and the Pan origin.

Data Transfer instructions are used to transfer graphics data between the Ω400 and the host system, as well as from one display position to another. These instructions are described in Section 8.

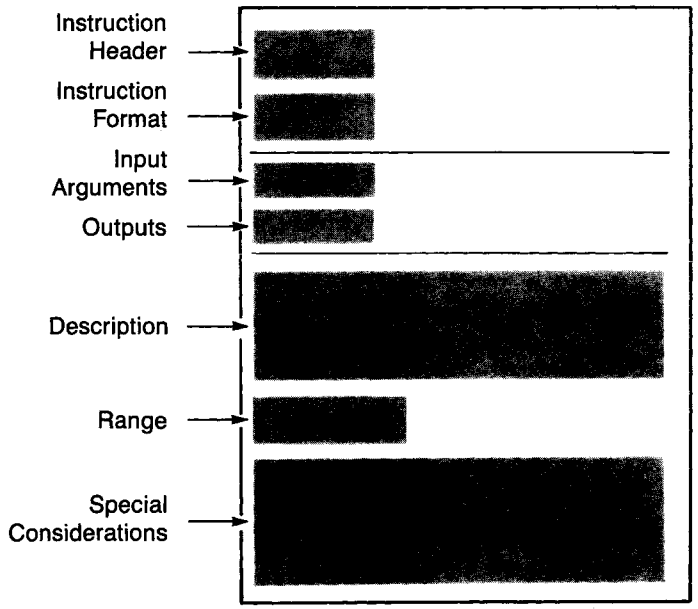
Section 9 contains descriptions of the Utility instructions. The Utilities control such functions as SYNCH (used in animation, for instance), and the on-board signature analyzer for troubleshooting.

Using the Instruction Set Documentation

Each instruction within each group is described in a standard format, designed to make information easily found and accessible. There are some differences in operation between the Ω420 and the Ω440; these are detailed in the instruction descriptions.

Instruction documentation within the next six sections of the manual uses the following format:

Instruction Header	Identifies and describes the instruction
Instruction Format	Identifies the op code and arguments
Input Arguments	Details the instruction arguments
Outputs	Details any outputs from the Ω400
Description	A detailed description of the instruction
Range	The allowable range for numerical arguments
Special Considerations	Additional important information/details



Section 4

DISPLAY POINTER MOVE INSTRUCTIONS

The display pointers are important basic elements in the graphic display. They are used to position the origin of many of the graphic display units, and are positioned with the Display Pointer Move instructions. The display pointers are used to define vector positioning, rectangles, arcs, polygons, and even the text characters on the display.

The Display Pointer Move instructions include the following, described on the following pages:

- **MOVP1 - Pointer 1 Absolute Move**
- **MOVP2 - Pointer 2 Absolute Move**
- **RMOVP1 - Pointer 1 Relative Move**
- **RMOVP2 - Pointer 2 Relative Move**
- **POLYS - Start a Polygon Definition**
- **POLYV - Add a Polygon Vertex**
- **POLYM - Polygon Move**
- **POLYC - Close the Current Polygon**

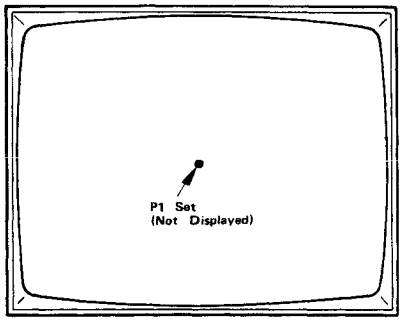
MOVP1

Pointer 1 Absolute Move

Instruction Format

HEX: 52 lox hix loy hiy

ASCII: R lox hix loy hiy



Input Arguments

lox	the low-order eight bits of the ten bits required to define the x coordinate value
hix	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 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 Pointer 1 to a point specified in absolute coordinate values. The **R** character (hexadecimal 52) is the operational code for the Pointer 1 Absolute Move instruction. It is followed immediately by four bytes (lox, hix, loy, hiy) that define the new location for Pointer 1.

Range

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

If the pointer is placed out of range, undesirable "wrap around" results may occur.

Special Considerations

A cross-reference chart for ASCII, decimal, octal, and hexadecimal values is located in Appendix A of this manual.

Example

R 01100100 00000000 01100100 00000000

This moves P1 to 100x,100y.

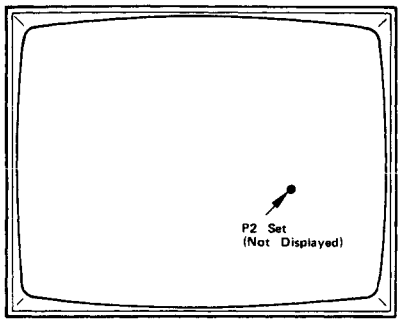
MOVP2

Pointer 2 Absolute Move

Instruction Format

HEX: 53 lox hix loy hiy

ASCII: S lox hix loy hiy



Input Arguments

lox	the low-order eight bits of the ten bits required to define the x coordinate value
hix	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 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 Pointer 2 to a position specified in absolute coordinate values. The S character (hexadecimal 53) is the operational code for the Pointer 2 Absolute Move instruction. It is followed immediately by four bytes (lox, hix, loy, hiy) that define the new location for Pointer 2.

Range

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

If the pointer is placed out of range, undesirable "wrap around" results may occur.

Special Considerations

A cross-reference chart for ASCII, decimal, octal, and hexadecimal values is located in Appendix A of this manual.

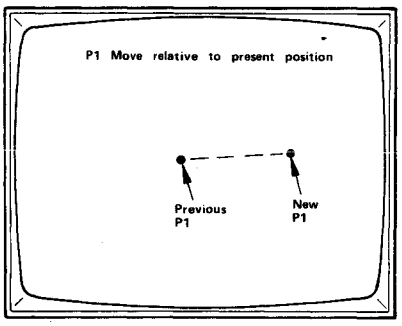
Example

S 11001000 00000000 00101100 00000001

This moves P2 to 200x,300y.

RMOVP1

Pointer 1 Relative Move



Instruction Format

HEX: 54 lox hix loy hiy

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

The T character (hexadecimal 54) is the operational code for the Pointer 1 Relative Move instruction. This instruction moves Pointer 1 relative to its present position, by the specified coordinate distances. The relative distance is specified by the four bytes (lox,hix,loy,hiy) that follow the instruction code. This forms a 12-bit 2's complement argument for x and y. The four bytes are defined above.

Range

The allowable range of arguments for the x and y coordinate move is -2048 to +2047. The values are assigned in 12-bit 2's complement.

The range is illustrated below:

XXXX	0111	1111	1111	+2047
		:		
		:		
		:		
XXXX	0000	0000	0001	+1
XXXX	0000	0000	0000	0
XXXX	1111	1111	1111	-1
		:		
		:		
		:		
XXXX	1000	0000	0000	-2048

A relative move past the 1024 x 1024 graphic memory boundary will "wrap around" to the other side of the display.

Special Considerations

A cross-reference chart for ASCII, decimal, octal, and hexadecimal values is located in Appendix A of this manual.

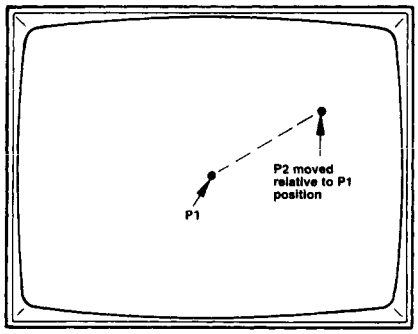
Example

(P1 currently at 100x,100y)
T 10010110 00000000 10010110 00000000 (Binary 150,150)

This example will move P1 to 250,250.

RMOVP2

Pointer 2 Relative Move



Instruction Format

HEX: 55 lox hix loy hiy

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

The U character (hexadecimal 55) is the operational code for the Pointer 2 Relative Move instruction. This instruction moves Pointer 2 by the specified increment, relative to the present position of Pointer 1. The relative distance in each axis is defined by the four bytes (lox,hix,loy,hiy) that follow the instruction code. This forms a 12-bit 2's complement argument for x and y. The four bytes are defined above.

Range

The allowable range of arguments for the x and y coordinate move is -2048 to +2047. The values are assigned in 12-bit 2's complement.

The range is illustrated below:

XXXX	0111	1111	1111	+2047
		:		
		:		
		:		
XXXX	0000	0000	0001	+1
XXXX	0000	0000	0000	0
XXXX	1111	1111	1111	-1
		:		
		:		
		:		
XXXX	1000	0000	0000	-2048

A relative move past the 1024 x 1024 graphic memory boundary will "wrap around" to the other side of the display.

Special Considerations

A cross-reference chart for ASCII, decimal, octal, and hexadecimal values is located in Appendix A of this manual.

Example

(P1 currently located at 250x,250y)
U 01100100 00000000 01100100 00000000 (Binary 100,100)

This example will move P2 to 350,350.

POLYS

Start a Polygon Definition

Instruction Format

HEX: 56

ASCII: V

Input Arguments

None

Outputs

None

Description

The **V** character (hexadecimal 56) is the operational code for the Start a Polygon Definition instruction. Once this instruction is entered, the Display Controller begins constructing a polygon, using the polygon vertices defined with succeeding Add a Polygon Vertex (POLYV) instructions. (For each of these succeeding instructions, another vertex is added to the present polygon.)

For further information on polygon construction, see the Add a Polygon Vertex instruction (POLYV), and the Drawing instructions in Section 5.

Range

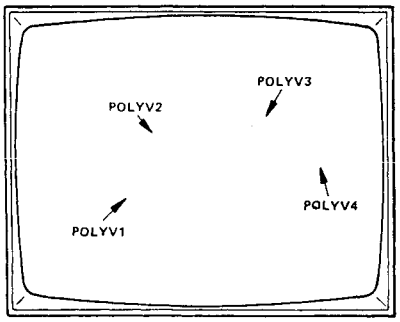
Does not apply

Special Considerations

The Start a Polygon Definition instruction sets up the Display Controller to build a polygon, but takes no display action on its own. Polygon vertices are added with the POLYV instruction, but the polygon remains in memory. The polygon is displayed using the POLYF (Fill a Polygon) Drawing instruction to display a filled polygon, the POLYO (Outline a Polygon) Drawing instruction to display an outlined polygon, or both (with separate colors) to display a filled polygon outlined in another color.

POLYV

Add a Polygon Vertex



Instruction Format

HEX: 57 lox hix loy hiy

ASCII: W lox hix loy hiy

Input Arguments

lox	the low-order eight bits of the ten bits required to define the x coordinate value
hix	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 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

The **W** character (hexadecimal 57) is the operational code for the Add a Polygon Vertex instruction. This instruction is used to add each vertex to the polygon started with the Start a Polygon Definition (POLYS) instruction. The coordinate for each vertex is defined by the four bytes that follow the POLYV instruction code. These bytes are defined above.

The pointers remain in their previous position; they are not moved during the POLYV instruction.

Range

The allowable range for x,y coordinate data is 0 to 1023. The maximum number of polygon vertices is 440.

Special Considerations

A cross-reference chart for ASCII, decimal, octal, and hexadecimal values is located in Appendix A of this manual.

No display action takes place on the monitor during polygon construction (Start a Polygon Definition or Add a Polygon Vertex). Once all vertices are defined, the polygon is displayed using the POLYF (Fill a Polygon) Drawing instruction to display a filled polygon, the POLYO (Outline a Polygon) Drawing instruction to display an outlined polygon, or both (with separate colors) to display a filled polygon outlined in another color.

Note that all convex polygons can be filled with a POLYF instruction. Some others can be filled, if no horizontal line through the polygon crosses more than two polygon edges.

Example

V (Polygon Start)
W 01100100 00000000 01100100 00000000 = First vertex (100x,100y)
W 11001000 00000000 00101100 00000001 = Second vertex (200x,300y)
W 00101100 00000001 01100100 00000000 = Third vertex (300x,100y)
f (Executes an outlined triangular polygon with the above vertices)

or

g (Executes a filled triangular polygon with the above vertices)

POLYM

Polygon Move

Instruction Format

HEX: 45 lox hix loy hiy

ASCII: E lox hix loy hiy

Input Arguments

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

The E character (hexadecimal 45) is the Polygon Move instruction. When a POLYM adds a vertex to the polygon data structure, a move occurs from the last polygon vertex to the POLYM x,y point. The resultant 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.

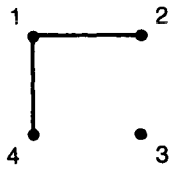
Range

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

Special Considerations

See POLYV in this section for polygon construction. Be aware of the case when POLYM follows a POLYC or POLYS command. This causes a move to occur when the polygon or substructure is closed. (See Fig. 4-1.) Note that both cases shown will fill identically, as rectangles.

```
POLYV X1,Y1
POLYV X2,Y2
POLYM X3,Y3
POLYM X4,Y4
POLYO
```




```
POLYM X1,Y1
POLYV X2,Y2
POLYV X3,Y3
POLYM X4,Y4
POLYO
```

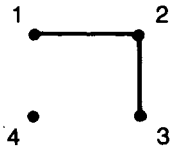


Fig. 4-1. Two Examples of the POLYM Command

POLYC

Close the Current Polygon

Instruction Format

HEX: 44

ASCII: D

Input Arguments

None

Outputs

None

Description

The **D** character (hexadecimal 44) is the operational code for the Close the Current Polygon instruction. This instruction will close the present polygon substructure, and functions 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

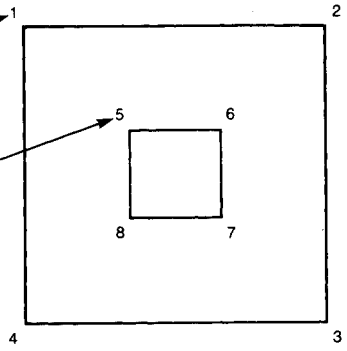
Does not apply

Special Considerations

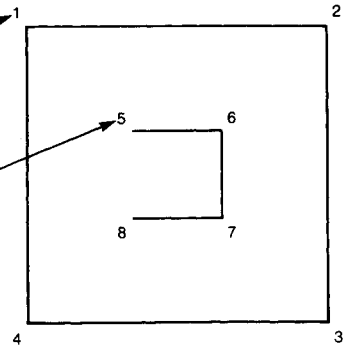
The POLYV command following a POLYC instruction will begin a new polygon substructure. Use caution when following a POLYC command with POLYM.

Examples

```
POLYS
POLYV (100,100)
POLYV (200,100)
POLYV (200,200)
POLYV (100,200)
POLYC
POLYV (150,150)
POLYV (175,150)
POLYV (175,175)
POLYV (150,175)
POLYO
```




```
POLYS
POLYV (100,100)
POLYV (200,100)
POLYV (200,200)
POLYV (100,200)
POLYC
POLYM (150,150)
POLYV (175,150)
POLYV (175,175)
POLYV (150,175)
```



NOTE: POLYM causes no outline to vertex (150,150).

Section 5

DRAWING INSTRUCTIONS

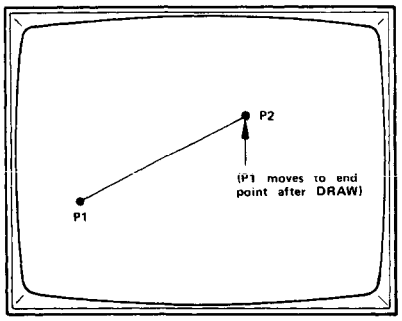
The Drawing instructions are the basic action instructions that are used to display graphic images on the monitor. Many of the instructions use the display pointers (Pointer 1 and Pointer 2) for graphics positioning. Section 4 defines the Display Pointer Move instructions.

The Drawing instructions include the following, described in this section:

- **DRAW - Draw a Vector**
- **COMPDR - Draw a Vector -- Complement Pixels**
- **CHAR - Draw a Character**
- **ARC - Draw an Arc**
- **RECT 1 - Draw Rectangle Outline**
- **RECT 2 - Fill a Rectangle**
- **FFILL - FLASH-fill a Rectangle**
- **CLEAR - Clear Image Memory**
- **POLYF - Fill a Polygon**
- **POLYO - Outline a Polygon**
- **AFILL 1 - Random Area "Seed" Fill**
- **AFILL 2 - Area Fill -- Boundary Color Specified**
- **RLFILL - Runlength Fill**
- **XDRAW - Exclusive OR Draw**

DRAW

Draw a Vector



Instruction Format

HEX: 61

ASCII: a

Input Arguments

None

Outputs

None

Description

The **a** character (hexadecimal 61) is the operational code for the Draw a Vector instruction. Once executed, a vector one pixel wide is drawn from Pointer 1 (P1) to Pointer 2 (P2), inclusive of both points. P1 and P2 are previously defined using the Display Pointer Move instructions, described in Section 4.

The vector is drawn in the currently-selected draw color and pattern. After the draw is completed, both P1 and P2 are set to the P2 coordinate position (the vector end).

Note that if P1 and P2 are set to the same coordinate position, the resulting display is a one-pixel dot.

Range

Does not apply

Special Considerations

P1 moves to the P2 coordinates after the vector is drawn, to aid in relative moves and in chaining of vectors.

Note that the drawn vector includes both the P1 start point and the P2 end point.

Example

R 01100100 00000000 01100100 00000000 (Sets P1 to 100x,100y)
S 11001000 00000000 00101100 00000000 (Sets P2 to 200x,300y)
a (Draws a vector from 100x,100y to 200x,300y)
U 10010110 00000000 10010110 00000000 (RMOVP2 to 350x,450y)
a (Draws a vector from 200x,300y to 350x,450y)

COMPDR

Draw a Vector - Complement Pixels

Instruction Format

HEX: 72

ASCII: r

Input Arguments

None

Outputs

None

Description

The **r** character (hexadecimal 72) is the operational code for the COMPDR instruction, which draws a vector from P1 to P2. When using the COMPDR, each pixel write in the vector is preceeded by a pixel read and pixel complement, thus drawing the vector in the color complement of existing pixels. The current draw color is not used. Vector drawing is noticeably slower using COMPDR.

Range

Does not apply

Special Considerations

Conditions affecting a normal vector draw are disabled during the COMPDR operation. The vector is written in the complement of existing pixel colors. The current write mask and current stipple and line patterns remain in effect. The current draw color is ignored.

P1 moves to the location of P2 after completing the COMPDR operation.

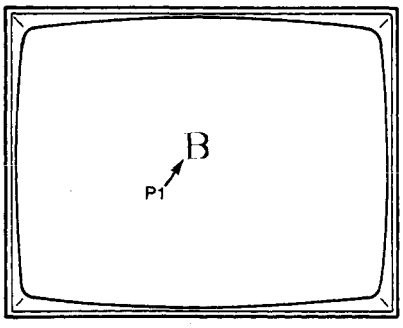
CHAR

Draw a Character

Instruction Format

HEX: 6B text (escape)

ASCII: k text (escape)



Input Arguments

text	the ASCII text characters to be displayed

Outputs

None

Description

The **k** character (hexadecimal 6B) is the operational code for the Draw a Character instruction. This instruction sets the Ω400 System to Text Mode. All displayable ASCII characters received after this instruction will be displayed as text until an escape character is received; the escape character sets the Ω400 System back to Graphics Mode.

Characters are drawn using the character size and rotation parameters selected previously. Previously-selected colors and drawing patterns also apply. These parameters are described in the Drawing Control instructions.

Range

Does not apply

Special Considerations

When operating in normal Text Mode, the correct character spacing (between characters and between lines) is inserted as a part of each character block. When drawing rotated or mirrored characters sequentially, it's best to insert an escape character after each text character, redefine P1 with a Display Pointer Move instruction, then issue another Draw a Character instruction for the next character. This will ensure the desired text placement.

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 Ω400 System to Graphics Mode.

Previously-defined line patterns will be applied to the text characters, and may produce undesirable results on the display. Fill patterns may be used as character backgrounds with positive results, especially with larger character sizes.

Example

k This is a Test. (escape)

"This is a Test." will be displayed, and the Ω400 System will return to Graphics Mode.

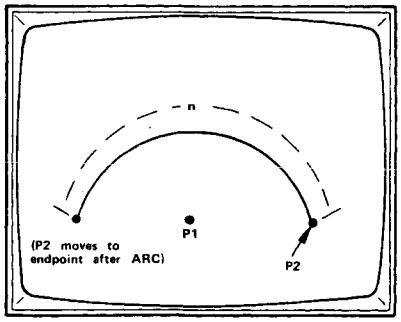
ARC

Draw an Arc

Instruction Format

HEX: 62 nn

ASCII: b nn



Input Arguments

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

The number of pixels required to draw a given circle can be calculated using the following formula, then rounding up to the next higher integer: $4r\sqrt{2}$ ("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 **b** character (hexadecimal 62) is the operational code for the Draw an Arc instruction. It is followed immediately by two bytes (nn) that specify the arc length in pixels, in the range 0 to 2047. When this instruction is executed, an arc one pixel wide is drawn (counter-clockwise) the specified length from P2, using P1 as the center point. P1 and P2 are defined previously, using the Display Pointer Move instructions.

The arc is drawn in the current draw color and line style, and includes the defined start point P2.

Range

The arc can be specified from 0 to 2047 pixels in length.

Special Considerations

P2 moves to the endpoint of the arc after it is drawn, to allow the same arc to be easily continued. This can be used to continue longer arcs, or to use the ARC instruction to draw a circle.

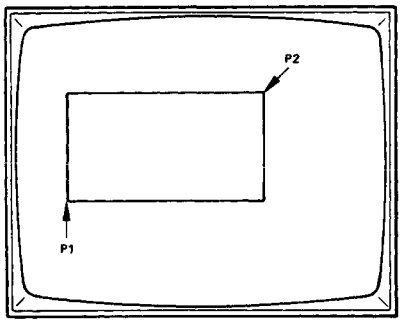
Example

```
R 10010110 00000000 10010110 00000000 (Set P1 to 150x,150y
      - the arc center)
S 00110100 00000000 10010110 00000000 (Set P2 to 50x,150y)
b 00011011 00000001 (Calculated arc length of 283 pixels)
```

This draws an arc 283 pixels long, starting at 50x,150y and ending at 250x, 150y. The arc center is 150x,150y.

RECT 1

Draw Rectangle Outline



Instruction Format

HEX: 63

ASCII: c

Input Arguments

None

Outputs

None

Description

The **c** character (hexadecimal 63) is the operational code for the Draw Rectangle Outline instruction. When this instruction is executed, a rectangle of vectors one pixel wide is drawn on the monitor, using the current draw color and line style. The location and size of the rectangle is defined by P1 and P2; these set two diagonally-opposed corners of the rectangle.

The displayed rectangle includes the defined corner points, P1 and P2. P1 and P2 remain set at these corners after the rectangle is drawn.

Range

Does not apply

Special Considerations

To draw a rectangle of the currently-selected color, outlined in another color, the filled rectangle must be created first using the Fill a Rectangle or FLASH-fill instruction. Then reestablish P1 and P2 at the same coordinates, select another color, and use the Draw Rectangle Outline instruction to outline the rectangle in the new color.

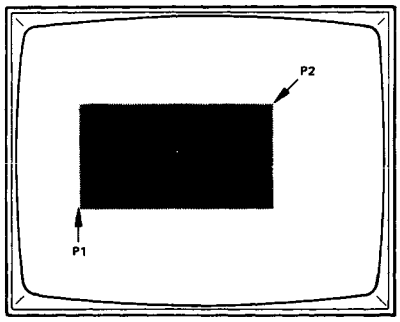
Example

```
R 01100100 00000000 00101100 00000001 (Set P1 to 100x,300y)
S 11001000 00000000 01100100 00000000 (Set P2 to 200x,100y)
c
```

This draws a rectangle with the lower left corner at 100x,300y and the upper right corner at 200x,100y.

RECT 2

Fill a Rectangle



Instruction Format

HEX: 64

ASCII: d

Input Arguments

None

Outputs

None

Description

The **d** character (hexadecimal 64) is the operational code for the Fill a Rectangle instruction. Once executed, a filled rectangle is drawn on the display in the current draw color and fill style (stipple pattern). The size and location of the rectangle is defined by the two pointers (P1 and P2). P1 and P2 are defined previously, using the Display Pointer Move instructions.

The displayed rectangle includes the two diagonal corner pointers, P1 and P2. P1 and P2 remain set at these corners after the rectangle is filled.

Range

The range of the Display Pointer Move instructions applies.

Special Considerations

To draw a rectangle of the currently-selected color, outlined in another color, the filled rectangle must be created first using the Fill a Rectangle or FLASH-fill instruction. Then select another color, and use the Draw Rectangle Outline instruction to outline the rectangle in the new color.

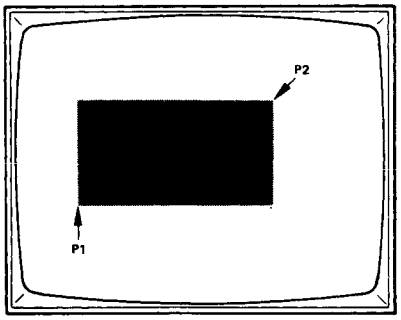
Example

```
R 01100100 00000000 00101100 00000001 (Set P1 to 100x,300y)
S 11001000 00000000 01100100 00000000 (Set P2 to 200x,100y)
d
```

This displays a filled rectangle with the lower left corner at 100x,300y and the upper right corner at 200x,100y.

FFILL

FLASH-fill a Rectangle



Instruction Format

HEX: 65

ASCII: e

Input Arguments

None

Outputs

None

Description

The e character (hexadecimal 65) is the operational code for the FLASH-fill a Rectangle instruction. This instruction performs the FLASH-fill of a rectangle defined by P1 and P2, in the current draw color. P1 and P2 define two diagonally opposed corners of the rectangle. These points are included in the rectangle. P1 and P2 are defined previously using the Display Pointer Move instructions.

FLASH-fill operates significantly faster than Fill a Rectangle, but does not use the current fill pattern. FLASH-fill is always a solid fill.

Range

Does not apply

Special Considerations

Note that the rectangle includes the defined corner points, P1 and P2. P1 and P2 remain at the rectangle corners after the FLASH-fill operation.

To draw a rectangle of the currently-selected color, outlined in another color, the filled rectangle must be created first using the Fill a Rectangle or FLASH-fill a Rectangle instruction. Then select another color, and use the Draw Rectangle Outline instruction to outline the rectangle in the new color.

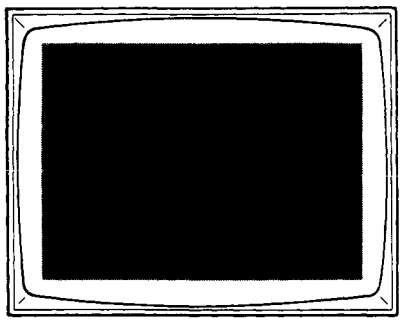
Example

R 01100100 00000000 00101100 00000001 (Set P1 to 100x,300y)
S 11001000 00000000 01100100 00000000 (Set P2 to 200x,100y)
e

This FLASH-fills a rectangle with the lower left corner at 100x,300y and the upper right corner at 200x,100y.

CLEAR

Clear Image Memory



Instruction Format

HEX: 60

ASCII: ' (apostrophe)

Input Arguments

None

Outputs

None

Description

The ' character (apostrophe - hexadecimal 60) is the operational code for the Clear Image Memory instruction. This instruction can be used to clear the entire memory to the current draw color, or to clear selected memory planes to the current color.

The Set the Write Mask instruction is used prior to the CLEAR instruction, to select the memory planes to be write-enabled. These are the planes that will be cleared to the current draw color when the CLEAR instruction is issued. The current color is selected with the Set the Drawing Color instruction.

Range

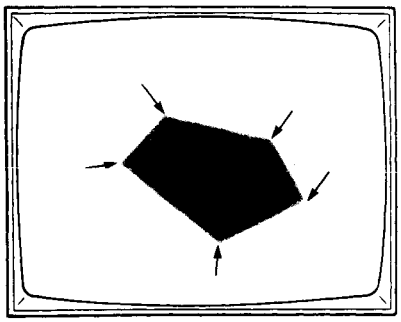
See the Set the Write Mask and Set the Drawing Color instructions.

Special Considerations

If individual memory planes are to be cleared, make certain that only those planes are write-enabled (with the Set the Write Mask instruction). If all planes are enabled (the state at initialization), then the entire memory will be cleared to the selected color.

POLYF

Fill a Polygon



Instruction Format

HEX: 67

ASCII: g

Input Arguments

None

Outputs

None

Description

The **g** character (hexadecimal 67) is the operational code for the Fill a Polygon instruction. When this instruction is executed, the current polygon is filled with the current draw color and fill pattern. The polygon has been constructed using the Start a Polygon Definition (POLYS) and some combination of the POLYV, POLYC, and POLYM instructions.

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

Range

See the description for POLYV; also see **Special Considerations.**

Special Considerations

If the desired result is a polygon of one color, outlined in another color, first fill the polygon with the POLYF instruction. Then select another color for the outline (using Set the Drawing Color), and outline with the POLYO instruction. In this case it is not necessary to redefine the vertices of the polygon.

If the above operation is performed in the opposite order (POLYO then POLYF), the outline color will be overwritten.

The POLYF command invokes a parity fill algorithm which will fill convex, concave, non-planar, and nested polygons. This parity fill will fill the interior regions of a polygon, where the interior region is defined as that part of the polygon which 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 examples below.) The algorithm will properly fill arbitrarily complex polygons which do not exceed the formula:

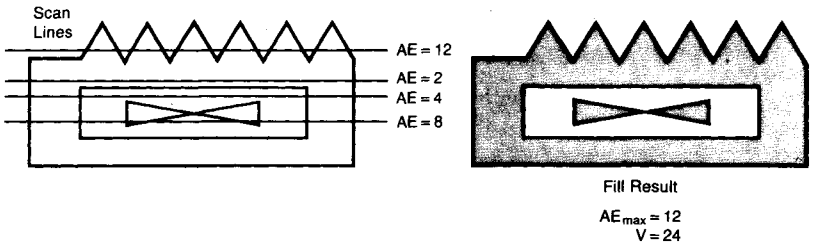
$$2(V) + 6(AE_{max}) \leqslant 990$$

where

V is the total number of vertices in the polygon structure, including all sub-polygons

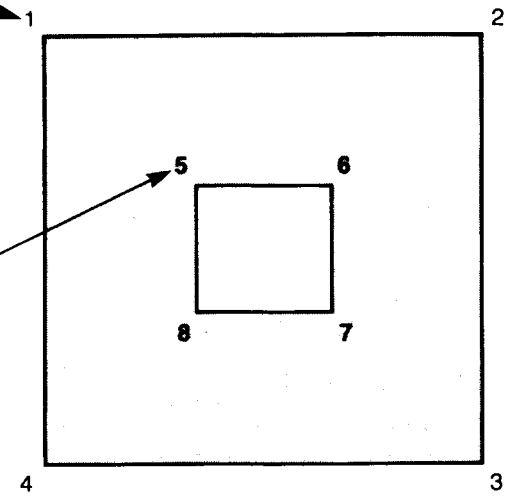
AE_{max} is the maximum number of active edges through which a scan line will cross

Examples



Drawing Instructions
POLYF - Fill a Polygon

```
POLYS
POLYV (100,100)
POLYV (200,100)
POLYV (200,200)
POLYV (100,200)
POLYC
POLYV (150,150)
POLYV (175,150)
POLYV (175,175)
POLYV (150,175)
POLYF
```



NOTE: The outline of the boxes in this figure is for clarity of illustration. In the given sequence, no outlining would occur.

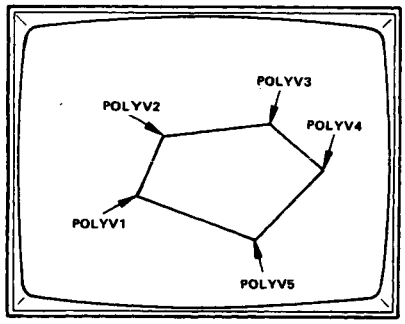
POLYO

Outline a Polygon

Instruction Format

HEX: 66

ASCII: f



Input Arguments

None

Outputs

None

Description

The **f** character (hexadecimal 66) is the operational code for the Outline a Polygon instruction. This instruction outlines the current polygon in the currently-selected draw color and line pattern. The current polygon has been constructed prior to the POLYO instruction, using the Start a Polygon Definition(POLYS) and Add a Polygon Vertex (POLYV) instructions.

P1 and P2 remain unchanged from their previously-set positions.

Range

See the POLYV instruction

Special Considerations

If the desired result is a polygon of one color, outlined in another color, first fill the polygon with the POLYF instruction. Then select the another color for the outline (using Set the Drawing Color), and outline with the POLYO instruction. In this case it is not necessary to redefine the vertices of the polygon.

If the above operation is performed in the opposite order (POLYO then POLYF), the outline color will be overwritten.

Example

```
V (Polygon Start)
W 01100100 00000000 01100100 00000000 = First vertex (100x,100y)
W 11001000 00000000 00101100 00000001 = Second vertex (200x,300y)
W 00101100 00000001 01100100 00000000 = Third vertex (300x,100y)
f
```

This example executes an outlined triangular polygon with the specified vertices.

AFILL 1

Random Area "Seed" Fill

Instruction Format

HEX: 68

ASCII: h

Input Arguments

None

Outputs

None

Description

The **h** character (hexadecimal 68) is the operational code for the Random Area "Seed" Fill instruction. The "seed" point is P1, which is the start point for the area fill. When the instruction is issued, the area starting with and surrounding P1 will be overwritten with the current color as long as pixels in the P1 color are encountered.

The fill stops at any boundary pixels, defined as those pixels not written in the initial color of the seed pixel that P1 points to. 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 "seed" position will determine the resulting display. See the illustrations in Figs. 5-1 and 5-2.

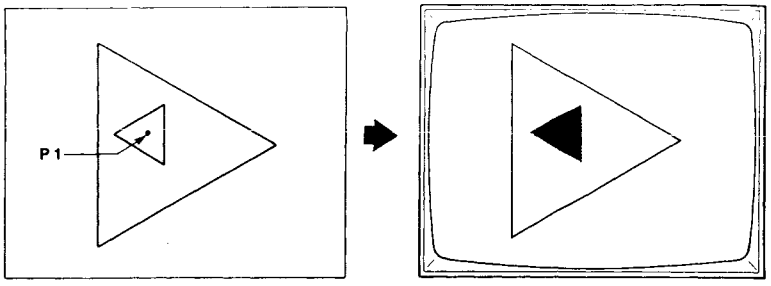


Fig. 5-1. P1 set to different color than the inner boundary

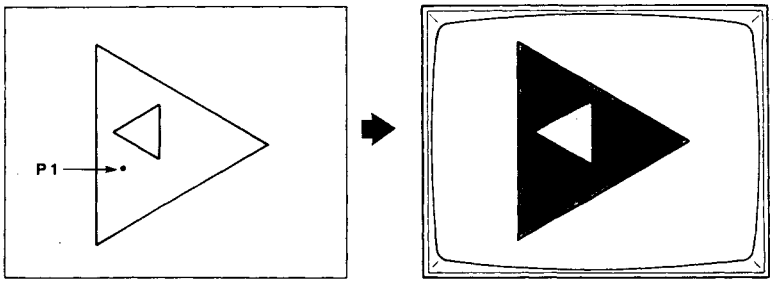


Fig. 5-2. P1 set between boundaries of two different colors

AFILL 2

Fill Area - Boundary Color Specified

Instruction Format

HEX: 69 a

ASCII: i a

Input Arguments

a — the color map address of the edge color

Outputs

None

Description

The **i** character (hexadecimal 69) is the operational code for the AFILL2 instruction. It is followed immediately by a one-byte argument, which selects the color map address of the desired edge color. AFILL2 begins at P1, and continues flooding with the present draw color until the specified edge color is reached.

Range

Ω420: The color map address range is 0 through 15.

Ω440: The color map address range is 0 through 255.

Special Considerations

The fill color is affected by the current write mask, but the edge color is not. Fill styles can be used, but filled areas may be limited in size due to stack overflow.

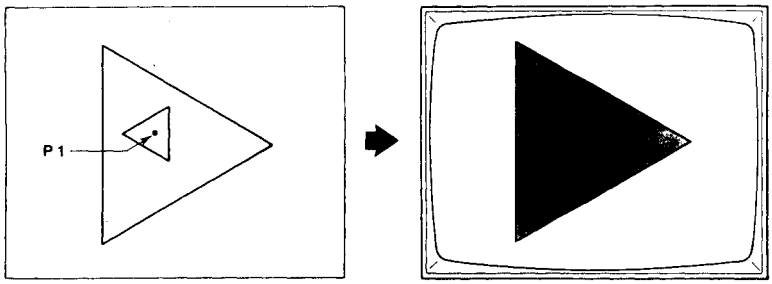


Fig. 5-3. Outer Boundary Color Specified

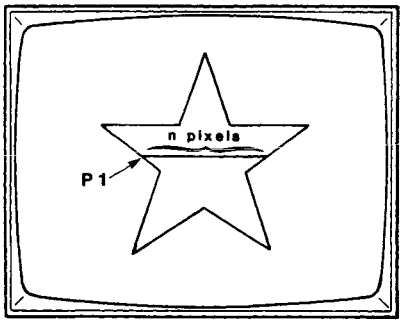
RLFILL

Runlength Fill

Instruction Format

HEX: 6A nn

ASCII: j nn



Input Arguments

nn	the number of pixels to fill (a binary sum in the range 0 - 1023)

Outputs

None

Description

The **j** character (hexadecimal 6A) is the operational code for the Run-length Fill instruction, used to fill complex polygon shapes that cannot be handled with the Fill a Polygon instruction. RLFILL is particularly useful for polygons that have concave vertices.

When RLFILL is issued, the Ω400 System fills from left to right parallel to the x axis, starting at P1. The length of the fill is determined by the binary count of the two bytes that immediately follow the instruction code. P1 is set prior to each RLFILL instruction, using a Display Pointer Move instruction.

Range

The pixel count range is from 0 to 1023.

Special Considerations

After RLFILL, P1 is left set to the next unfilled pixel to the right of the last pixel filled.

An additional use for the RLFILL instruction is to compress data in a predefined picture sent from the host to the Ω400 System.

Example

R 01100100 00000000 01100100 00000000 (Set P1 to 100x,100y)
j 11101110 00000010 (Binary count of 750)

This example fills from 100x,100y (in the current color) to 850x,100y.

XDRAW

Exclusive OR Draw

Instruction Format

HEX: 73

ASCII: s

Input Arguments

None

Outputs

None

Description

The **s** character (hexadecimal 73) is the operational code for the Exclusive OR Draw instruction. This instruction will draw a vector one pixel in width between P1 and P2 (inclusive of both points) using an exclusive OR of the previously existing pixel and the current color. P1 and P2 are previously defined using the move pointer instructions described in Section 4. 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 affect, but should be used with caution.

Section 6

DRAWING CONTROL INSTRUCTIONS

The Drawing Control instructions are environmental controls that affect the Drawing instruction actions. Drawing Controls set the drawing mode environments that are used during the execution of graphics. These include selecting the active color from the color map, enabling the various write mask levels (and therefore write color levels), selecting the fill patterns (stipple patterns) and line patterns, and controlling the size and orientation of displayed text characters.

The Drawing Controls include these instructions, described on the following pages:

- **SET COLOR - Set the Drawing Color**
- **WRMASK - Set the Write Mask**
- **RDMASK - Set the Read Mask**
- **PATTERN - Set the Fill or Line Pattern**
- **SETC SZ - Set the Character Size**
- **SETC ORN - Set the Character Orientation**
- **FSIZE - Set Font Size**
- **CSPACE - Set Character Spacing**

SET COLOR

Set the Drawing Color

Instruction Format

HEX: 4E c

ASCII: N c

Input Arguments

c	the color select byte, which operates differently for the Ω420 and the Ω440; in the Ω420, the binary sum of the least-significant four bits of this byte selects a color from the 16 available colors in the Ω420 color map; in the Ω440, the binary sum of all eight bits is used to select one of 256 color map addresses

Outputs

None

Description

The **N** character (hexadecimal 4E) is the operational code for the Set the Drawing Color instruction. The next byte is used to select one color from the color map, to be used for subsequent display operations (vectors, fills, arcs, rectangles, and text characters) until another color is selected with a subsequent Set the Drawing Color instruction.

In the Ω420, the four least-significant bits of the color select byte pick one color from the 16 in the Ω420's color map.

The Ω440 uses all eight bytes of the color select byte to set one of 256 possible colors at the current drawing color.

Range

Ω420: The allowable range is 0 through 15. (This corresponds to the size of the Ω420's color map.)

Ω440: The allowable range is 0 through 255. (This corresponds to the 256 available addresses in the Ω440's color map.)

Special Considerations

The Set the Drawing Color instruction is the final stage in selecting the color to be used to draw on the monitor screen. The first stage is selecting the colors to be loaded into the color map, from the device's available color palette, using the CMAP (Load Color Map Address) instruction. (The range of the color map and the palette depends on the Ω400 model. See **Range** , above.)

The second stage is write-enabling the desired memory planes through the Set the Write Mask instruction. 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.)

WRMASK

Set the Write Mask

Instruction Format

HEX: 4F d

ASCII: O d

Input Arguments

c	the mask select byte, which operates differently for the Ω420 and the Ω440; in the Ω420, the least-significant four bits of this byte select which bit planes are write-enabled; in the Ω440, all eight bits are used to select the write-enabled bit planes

Outputs

None

Description

The **O** character (hexadecimal 4F) is the operational code for the Set the Write Mask instruction. Setting the write mask allows the Ω400 System to write-enable specific bit planes. Each of the bits in the next byte is associated with one of the bit planes (the four least-significant bits in the Ω420.) These bits are used to set the write mask in any of the 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 if some bit planes are to be written while leaving others unchanged. The CLEAR instruction, for instance, uses the write mask to select which memory planes will be cleared.

Range

Ω420: The range is 0 through 15, corresponding to all possible combinations of four bit planes.

Ω440: The range is 0 through 255, corresponding to the possible combinations of eight bit planes.

Special Considerations

The Set the Drawing Color instruction selects a draw color to be used, from the available colors in the color map. In order for the color to be displayed, the appropriate memory plane must be write-enabled through the write mask. No drawing will occur in planes that are not write-enabled through the write mask. (All planes are normally enabled at power-up, and after an INIT instruction.)

RDMASK

Set the Read Mask

Instruction Format

HEX: 4C m

ASCII: L m

Input Arguments

m	the mask select byte, which operates differently for the Ω420 and the Ω440; in the Ω420, the least- significant four bits of this byte select which bit planes are read-enabled; in the Ω440, 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 L character (hexadecimal 4C) is the operational code for the Set the Read Mask instruction. It is followed by the mask select byte, each bit of which sets one bit plane to be on or off. A "one" bit enables the corresponding bit plane; a "zero" bit disables the corresponding bit plane. Data in disabled bit planes is not displayed.

Range

Ω420: The range is 0 through 15, corresponding to the 16 possible combinations of four bit planes.

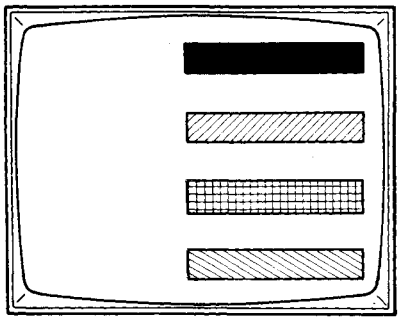
Ω440: The range is 0 through 255, corresponding to the 256 possible combinations of eight bit planes.

Special Considerations

The read mask does not affect data transfers from the Ω400 System memory to a host; it affects only the display of data.

PATTERN

Set the Fill or Line Pattern



Instruction Format

HEX: 50 n

ASCII: P n

Input Arguments

n — the pattern select byte; the eight bits are used as follows:
- least-significant three bits: set pattern
- bit four: select line pattern (1) or stipple pattern (0)
- bit five: normal (1) or inverted (0)
- most-significant three bits: select mode

Outputs

None

Description

The **P** character (hexadecimal 50) is the operational code for the Set the Fill or Line Pattern instruction. This instruction is used to select the line pattern to be used for line drawing operations, and the stipple pattern to be used for fill operations when other than solid fill. (Note that a solid line pattern will also select a solid fill.) The patterns are selected with the 8 bits of the byte following the instruction byte, as described above (n).

Table 6-1 shows the effect of the 8 combinations of the three most significant bits.

Range

Does not apply

Table 6-1
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 bits loaded into the color register.

All means all memory planes are affected, regardless of the write mask.

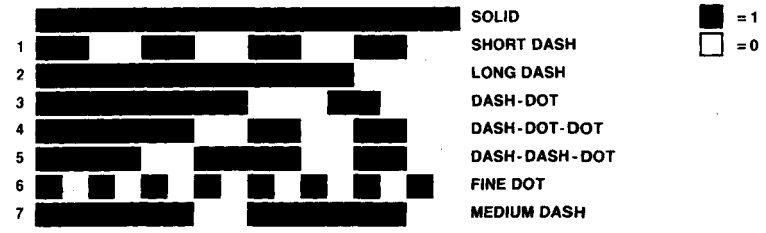
Selected means only the write-enabled memory planes are affected.

No change means that memory is not written at all.

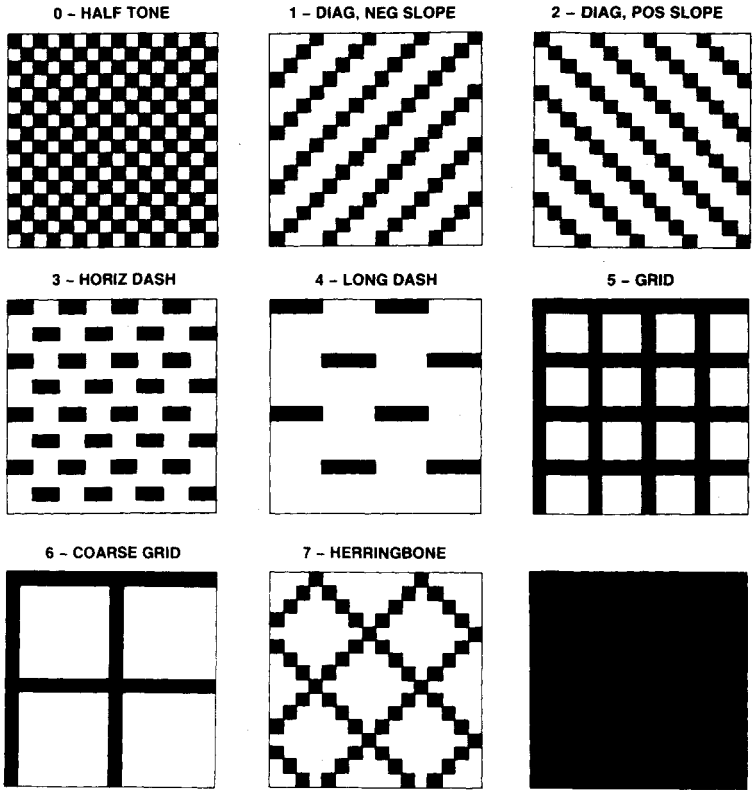
Special Considerations

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 to be used along with a pattern fill, the fill pattern must be reset after each solid line selection.

LINES

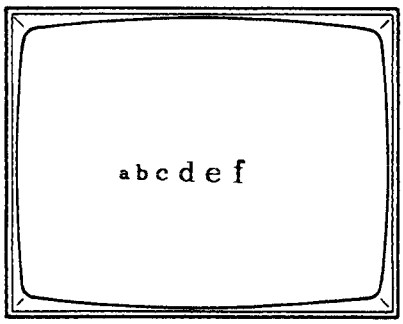

SOLID
SHORT DASH
LONG DASH
DASH-DOT
DASH-DOT-DOT
DASH-DASH-DOT
FINE DOT
MEDIUM DASH
1
2
3
4
5
6
7
= 1
= 0

AREA


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

SETC SZ

Set the Character Size



Instruction Format

HEX: 58 w h

ASCII: X w h

Input Arguments

w	the width scale factor (x axis magnification)
h	the height scale factor (y axis magnification)

Outputs

None

Description

The **X** character (hexadecimal 58) is the operational code for the Set the Character Size instruction. This instruction is used to enlarge the displayed text characters from their default size. The instruction code is followed by two bytes; the binary combination of each byte selects the character scale factor for one axis.

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

Range

The range of valid values in each axis is from 0 (1X) through 255 (256X). The usable screen sizes are noted below.

Special Considerations

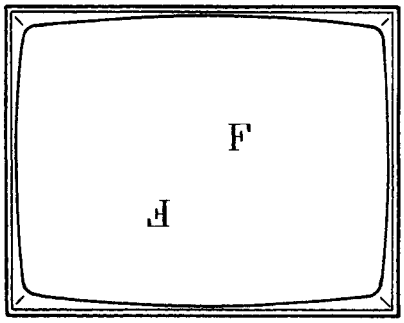
The maximum scale factor that allows a whole character to be displayed on the monitor screen depends on the display mode:

If the display is operating in 33 Hz mode, a y axis magnification factor of 47 (48X) results in a single character that fills the monitor screen.

If the display is operating in 60 Hz mode, a y axis magnification factor of 31 (32X) results in a single character that fills the monitor screen.

SETC ORN

Set the Character Orientation



Instruction Format

HEX: 59 d

ASCII: Y d

Input Arguments

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

Outputs

None

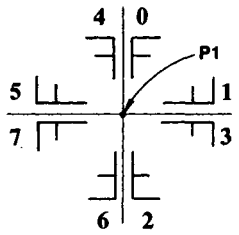
Description

The **Y** character (hexadecimal 59) is the Set the Character Orientation instruction. This selects whether the characters will be rotated in the x axis, the y axis, and whether the character will be mirrored. These conditions are specified by the eight possible combinations of the three least-significant bits. The three bits are assigned as follows:

bit 2	x rotation (0 = normal, 1 = rotated)
bit 1	y rotation (0 = normal, 1 = rotated)
bit 0	mirroring (0 = normal, 1 = mirrored)

These bits are detailed further on the following chart:

Number	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 x and y, and Mirror



Range

Does not apply

Special Considerations

When printing characters in normal mode, P1 moves to the lower left corner of the next character space after each character is printed. In other modes, it is best to set the origin point for each character, using a Pointer 1 Move instruction (MOVP1 or RMOVP1) prior to each Draw a Character instruction.

FSIZE

Set Font Size

Instruction Format

HEX: 49 w h

ASCII: I w h

Input Arguments

w	an 8-bit number representing font width
h	an 8-bit number representing font height

Outputs

None

Description

The I character (hexadecimal 49) is the Set Font Size command. This command invokes a viewport 8 pixels wide by h pixels high on the 8 x 16 character EPROM. This command is also used for the calculated default character spacing invoked by the SETCORN and the SETCSZ commands. In this regard, w and h specify character width and height.

Range

w: 1 to 8
h: 1 to 16

CSPACE

Set Character Spacing

Instruction Format

HEX: 48 Δx Δy

ASCII: H Δx Δy

Input Arguments

Δx	2 bytes that control the width of character auto spacing
Δy	2 bytes that control the height of character auto spacing

Outputs

None

Description

The **H** character (hexadecimal 48) is the Set Character Spacing instruction. This instruction controls the auto-incrementing of pointer P1 when character strings are used. The increment is applied after the character is drawn.

Range

$-2048 < \Delta x < 2047$
$-2048 < \Delta y < 2047$ (note that positive Δy is in a top to bottom direction)

Special Considerations

The Δx and Δy factors for character spacing are automatically adjusted according to the character orientation (see SETCORN) and the character size (see SETCSZ and FSIZE). For instance, the default (Δx=8, Δy=0) with a character orientation of 3 (see SETCORN) will write characters in a straight line down the screen. This might be used to label a vertical axis. Another solution would be a character orientation of 0 (normal) and a CSPACE of (Δx=0, Δy=16). This will label the vertical axis with normally-oriented characters which are located top to bottom.

The CSPACE command is over-ridden by the default action of the FSIZE, SETCORN and the SETCSZ instructions. CSPACE must follow any of these instructions to over-ride the default spacing. Note that the CSPACE must account for changes in SETCSZ. See example (e) for a character size of 1(x2).

Example

The following illustrates the results of using CSPACE and SETCORN together.

(a) SETCORN = 0

ab

(b) SETCORN = 3

(c) SETCORN = 0 CSPACE = 0,16

a

b

(d) SETCORN = 5 CSPACE = 16,0

(e) SETCSZ = 1 SETCORN = 0 CSPACE = 0, 32

a

b

Section 7

DISPLAY CONTROL INSTRUCTIONS

The Display Control instructions provide additional control over the display of graphics. The eight instructions in this group provide control of the available display colors, the magnification factor, and the location in memory to be mapped to the monitor screen.

The Display Control group includes the following instructions, described on the following pages:

- **CMAP - Load Color Map Address**
- **ZOOM - Select the Zoom Factor**
- **PPAN - Pan to Defined Origin**
- **CURS - Display the Cursor**
- **BLINK - Blink High-Order Bit Plane**
- **BLANK - Blank the Display**
- **SZCUR - Cursor Size**
- **CRTWR - CRT Write Register**

CMAP

Load Color Map Address

Instruction Format

HEX: 51 a rgb

ASCII: Q a rgb

Input Arguments

a	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 in the Ω440, to select an address in the range 0-255; only the least-significant four bits are used in the Ω420, selecting an address in the range 0-15
rgb	a three-byte sequence used to select the color from the Ω400's palette; the first byte selects the Red intensity, the second byte selects the Green intensity, and the third selects the Blue intensity; a value of zero (0) turns the color off; a value of FF (hex) sets full intensity for that color

Outputs

None

Description

The **Q** character (hexadecimal 51) is the operational code for the CMAP instruction. This instruction is used to load the colors into the color map addresses. Once loaded, these colors can be selected as the current draw color, using the Set the Drawing Color instruction. Three bytes allow color selection from a palette of 16.7 million colors. The Ω420 has 16 color map addresses into which the selected colors are placed. In the Ω420, each selected color is loaded into one of 256 color map addresses.

The instruction byte is followed by a one-byte argument for the address, then three bytes to define the color to be placed at the selected address.

Range

Ω420: color map addresses range from 0 through 15. There are 16.7 million possible colors in the palette.

Ω440: color map addresses range from 0 through 255. There are 16.7 million possible colors in the palette.

ZOOM

Select the Zoom Factor

Instruction Format

HEX: 5A n

ASCII: Z n

Input Arguments

n	the zoom factor byte; the four least-significant bits select a zoom factor in the range 0 (1X) → 15 (16X)

Outputs

None

Description

The **Z** character (hexadecimal 5A) is the operational code for the ZOOM instruction. The zoom factor is used to magnify the contents of graphics memory for display on the monitor. The least-significant four bits of the byte that follows the instruction code are interpreted as a binary representation of the zoom factor. A zoom factor of 0 is equivalent to 1X magnification, or normal viewing. A zoom factor of 15 is equivalent to 16X magnification; this is the greatest zoom factor allowable.

The selected zoom factor remains in effect until another zoom factor is selected, or until the system is initialized again, returning parameters to their default values. Initialization occurs at power-up, or when an INIT instruction occurs.

Range

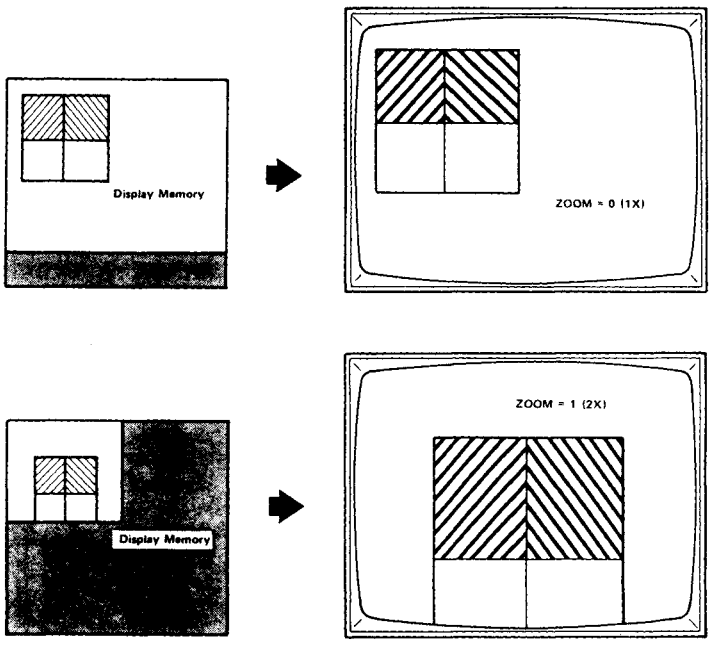
The zoom factor can range from 0 (1X - default) to 15 (16X - maximum).

Special Considerations

ZOOM is typically used in conjunction with the Pan instruction (PPAN) to zero in on a particular area of graphics memory. See **Special Considerations** under PPAN.

Example

Z XXXX0001 (Sets a 2X magnification factor)
or
Z XXXX0111 (Sets an 8X magnification factor)



PPAN

Pan to Defined Origin

Instruction Format

HEX: 5B

ASCII: [

Input Arguments

None

Outputs

None

Description

The [character (hexadecimal 5B) is the operational code for the PPAN instruction. 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 in conjunction with the ZOOM instruction, to move to and magnify a particular area of display memory.

Range

See the Display Pointer Move instructions.

Special Considerations

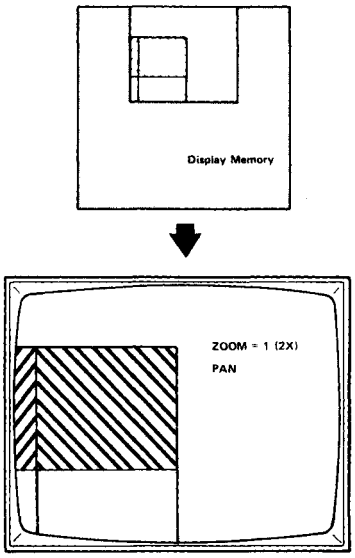
The PPAN instruction follows Pointer 1, which can usually be moved in one-pixel increments. The exception to this rule occurs when the zoom factor is set to zero (1X magnification), in which case the minimum horizontal move is 16 pixels. Vertical moves can still occur in one-pixel increments.

If P1 is moved far to the right (the exact value depends on the zoom factor), the display may "wrap around." That is, the display memory area to the right of the viewable area may wrap around and appear on the left area of the monitor screen.

Example

R 01100100 00000000 11001000 00000000 (Moves P1 to 100x,200y)
[

This example establishes 100x,200y as the memory origin point.



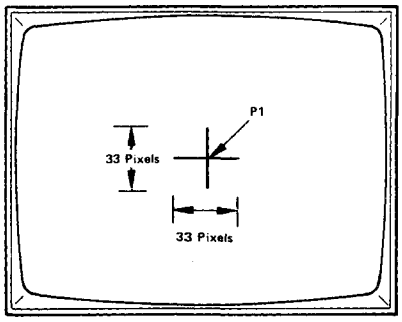
CURS

Display the Cursor

Instruction Format

HEX: 71

ASCII: q



Input Arguments

None

Outputs

None

Description

The **q** character (hexadecimal 71) is the operational code that enables the cursor display. When this instruction is used, a crosshair cursor appears at P1 on the display and remains on the display until another instruction is received by the Ω400 System. The cursor is displayed in the complement of existing pixel colors. The cursor is 33 pixels high and 33 pixels wide.

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.

BLINK

Blink High-Order Bit Plane

Instruction Format

HEX: 4D f

ASCII: M f

Input Arguments

f	the blink flag, following the op code; if this flag is 0, BLINK will be off; if the flag is a 1, the high-order bit plane is switched in and out at 2 Hz

Outputs

None

Description

The **M** character (hexadecimal 4D) is the operational code for the BLINK instruction; it is followed immediately by a flag byte. 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.

For the Ω420, this blinks the fourth bit plane (plane 3); for the Ω440 the eighth bit plane (plane 7) is blinked.

Range

Does not apply

Special Considerations

BLINK switches one bit plane in and out of the display. It can be used for emphasis, calling attention to a particular portion of a display. (This portion must also be written into the high-order bit plane.)

BLANK

Blank the Display

Instruction Format

HEX: 4B f

ASCII: K f

Input Arguments

f	the blank flag, following the op code, is the least-significant bit of the byte argument; if this flag is 0, blanking will be off; if the flag is a 1, screen blanking is enabled

Outputs

None

Description

The **K** character (hexadecimal 4B) is the operational code for the BLANK instruction. It is followed immediately by the blanking flag byte, which sets blanking to be on or off. When blanking is enabled, the screen is blanked until another BLANK instruction turns blanking off.

Blanking can be used to speed up writing of a new display into the Ω400 display memory by blanking the display while writing into it. Pixel writing is approximately four times faster when the screen is blanked. Instructions which perform significant computation as well as pixel writing, such as AFILL 1 (Random Area "Seed" Fill), will achieve a lesser increase in speed.

Range

Does not apply

SZCUR

Cursor Size

Instruction Format

HEX: 47 w h

ASCII: G w h

Input Arguments

w	two bytes that control cursor width; actual cursor width is 2w+1
h	two bytes that control cursor height; acutal cursor height is 2h+1

Outputs

None

Description

The **G** character (hexadecimal 47) is the Cursor Size instruction. This instruction programs the variable cursor size. The default cursor size is 33 pixels wide and 33 pixels high. A full screen cursor is achieved with arguments of 0200,0200 hex.

Range

w: 0001-0200 (hex)
h: 0001-0200 (hex)

CRTWR

CRT Write Register

Instruction Format

HEX: 46 addr data

ASCII: F addr data

Input Agruments

addr	register to be written to (range 0 through 9)
data	desired register contents

Outputs

None

Description

Direct write of 6845-1 CRT Controller chip. This capability is used for transferring between RS-170 compatibility and the 1024 x 768 33 Hz interlaced mode. Transition between 60 Hz non-interlaced and RS-170 modes requires use of this instruction combined with a hardware jumper change.

Range

See CRTWR Table

CRTWR TABLE

Register Number	RS-343 (33 Hz interlaced)		RS-170		RS-343* (60 Hz non-interlaced)	
	Hex	Decimal	Hex	Decimal	Hex	Decimal
0	4F	79	8F	143	3F	63
1	42	66	7E	126	30	48
2	44	68	81	129	32	50
3	36	54	3B	59	36	54
4	67	103	56	86	49	73
5	04	04	01	01	00	00
6	60	96	50	80	45	69
7	60	96	51	81	45	69
8	03	03	03	03	00	00
9	06	06	04	04	07	07

* NOTE: Changing to 60 Hz also requires moving the interlace jumper in hardware

Section 8

DATA TRANSFER INSTRUCTIONS

The Data Transfer instructions are used to move blocks of graphics data. Four of the instructions are used to transfer data between the Ω400 System and the host computer. The PIXBLT instruction is used to move a block of graphic information from one location in graphics memory to a new location in graphics memory. GRAFIN sets one of eight possible modes for Graphic Input and local cursor control by means of a graphics tablet.

The following are the Data Transfer instructions, described on the following pages:

- **RDR - Read a Rectangle**
- **WRR - Write a Rectangle**
- **RPIXEL - Read a Pixel**
- **WPIXEL - Write a Pixel**
- **PIXBLT - Pixel Block Transfer**
- **GRAFIN - Select Graphic Input Mode**

RDR

Read a Rectangle

Instruction Format

HEX: 6E

ASCII: n

Input Arguments

None

Outputs

A rectangle of pixels defined by P1 and P2

Description

The **n** character (hexadecimal 6E) is the operational code for the Read a Rectangle instruction. This instruction reads a rectangle from the Ω400 System's display memory and transfers the data to the host computer. The rectangle is defined by P1 (which sets the lower left corner), and P2, which sets the upper right corner. P1 and P2 are defined prior to issuing the Read a Rectangle 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. The pixels are read from left to right, top to bottom.

Range

Does not apply

Special Considerations

When reading a rectangle to the host, the read mask is disabled, as are the current color and pattern. Existing data in all planes is transferred to the host, in the defined rectangle area.

This instruction does not change Pointer 1 or Pointer 2. The current drawing color, drawing pattern, read mask and write mask also remain unchanged after the instruction is completed.

Special Considerations

If P1 and P2 are set to the same point, one byte will be returned.

To read the entire display memory, P1 should be set to (0,0) and P2 should be set to (1024,1024). Δx and Δy will then be 1023.

WRR

Write a Rectangle

Instruction Format

HEX: 6F

ASCII: o

Input Arguments

None (P1 and P2 define the rectangle)

Outputs

Δx times Δy bytes (corresponding to the rectangle of pixels)

Description

The **o** character (hexadecimal 6F) is the operational code for the Write a Rectangle instruction. This instruction writes a rectangle (defined by P1 and P2) from the host computer into the Ω400 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.) They are defined prior to issuing the Write a Rectangle 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. The pixels are loaded from left to right, top to bottom.

Range

Ω420: Pixel values range from 0-15

Ω440: Pixel values range from 0-255

Special Considerations

When writing a rectangle from the host, the write mask remains in effect, and no writing can occur in disabled memory planes. If you wish to write into all planes, the write mask must be so enabled prior to this instruction. The pattern register (line and fill patterns), however, does not affect the Write a Rectangle operation.

This instruction does not change Pointer 1 or Pointer 2. The current drawing color, drawing pattern, and write mask also remain unchanged after the instruction is completed.

If P1 and P2 are the same point, only 1 pixel is written.

If Δx or Δy is zero, a single pixel is written in the corresponding axis.

To fill the entire display memory, P1 should be set to (0,0) and P2 should be set to (1024,1024). Δx and Δy will then be 1023.

RPIXEL

Read a Pixel

Instruction Format

HEX: 6C

ASCII: l

Input Arguments

None

Outputs

One byte, for the pixel color at P1

Description

The l character (hexadecimal 6C) is the operational code for the Read a Pixel instruction, which causes the Ω400 System to return the color map address for the pixel at P1.

The Ω420 returns a single byte, the least-significant four bits of which are a binary representation of the color map address.

All eight bits of the byte returned by the Ω440 represent its color map address, in the range 0 through 255.

Range

Ω420: The color map address range is 0 through 15.

Ω440: The color map address range is 0 through 255.

Special Considerations

This instruction does not change Pointer 1 or Pointer 2. The current drawing color, drawing pattern, and write mask also remain unchanged after the instruction is completed.

When reading a pixel to the host, the read mask is disabled, as are the current color and pattern. Existing data in all planes is transferred to the host, for the defined pixel.

WPIXEL

Write a Pixel

Instruction Format

HEX: 6D

ASCII: m

Input Arguments

None (P1 points to the pixel)

Outputs

None

Description

The **m** character (hexadecimal 6D) is the operational code for the Write a Pixel instruction. When the Write a Pixel instruction is received, the Ω400 System writes a single pixel at P1. The pixel is written in the currently-selected draw color. P1 is defined previously, using a Display Pointer Move instruction.

Range

See the Display Pointer Move instructions for range of pointer moves.

Special Considerations

When writing a pixel from the host, the write mask remains in effect, and no writing can occur in disabled memory planes.

This instruction does not change Pointer 1 or Pointer 2. The current drawing color, drawing pattern, and write mask also remain unchanged after the instruction is completed.

PIXBLT

Pixel Block Transfer

Instruction Format

HEX: 70 ww hh f

ASCII: p ww hh f

Input Arguments

ww — these two bytes follow the operational code 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) deter- mine the 4 high-order width bits

The width range is 0 through 1023; blocks that exceed the distance from the pointer to the display memory boundary may "wrap around" and appear on the opposite side of the display

hh — 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.

The height range is 0 through 1023; blocks that exceed the distance from the pointer to the display memory boundary may "wrap around" and appear on the opposite side of the display

f	this byte determines the direction of the block from the pointers, in the source and destination; it is fully described under the heading **Description** below

Outputs

None

Description

The **p** character (hexadecimal 70) is the operational code for the PIXBLT instruction. This instruction moves a block of pixels of a specified width, height, and direction from one area in display memory to another location in display memory. Pointer 1 (P1) defines the initial point of the starting rectangle (to be moved); P2 defines the initial point of the destination.

The direction of the rectangle from the initial point is defined by the last byte of the input arguments (f, above). Five bits (0 through 4) of this byte are used to 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 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 (hh) 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 for the x axis, determining the direction (from the initial point) of the previously-specified width count.

The last two bits used, 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 and width ranges are 0 through 1023. Blocks that exceed the distance from the pointer to the display memory boundary may "wrap around" and appear on the opposite side of the display.

GRAFIN

Select Graphic Input Mode

Instruction Format

HEX: 4A f

ASCII: J f

Input Arguments

f — the GRAFIN mode select byte; this byte, which follows the op code, selects the GRAFIN mode; the selectable modes are listed below, and further defined under the heading "Description"

0	Software INIT
1	Local Cursor Control
2	Set Transparent Mode
3	Set Offset and Scale Factors (See Special Considerations)
4	Set Delimiter
5	Same Position - Screen Coordinates
6	Sample Position - Tablet Coordinates
7	Set Mode Register

Outputs

None

Description

The **J** character (hexadecimal 4A) is the operational code for the GRAFIN (Graphic Input) instruction. It is followed immediately by another byte, which defines the Graphic Input function. These modes are defined as follows:

0 Software INIT. Resets all GRAFIN attributes to their default values, and clears the coordinate queue. These attributes include the offset and scale factors. 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 8-1 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 Ω400 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 8-1
Ω400 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 = button depression
This allows detection of the "O" key on the GTCO tablet

2 Set Transparent Mode. The Ω400 is effectively removed from the data link between the Ω400 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 Ω400 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 fraction
4 - x multiplier integer
5 - y offset low byte
6 - y offset high byte
7 - y multiplier fraction
8 - y multiplier integer
(See **Special Considerations**)

4 Set Delimiter. The next byte specifies the delimiter, replacing the default delimiter. Transmitting the delimiter during Transparent Mode causes exit from Transparent Mode. The delimiter can range from 0 to FE(hex); the default delimiter is 80 (hex).

5 Sample Position - Screen Coordinates. The screen cursor position is relayed in the same format that is sent by GRAFIN 1.

6 Sample Position - Tablet 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	Report button depression during GRAFIN 1	No output during GRAFIN 1

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

Alternately, one may set or clear an individual bit without modifying the others by setting bit 7 to a 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).

Range

There are eight Graphic Input functions.

Special Considerations

The GRAFIN function allows the Ω400 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 Ω400. In Transparent Mode, however, the Ω400 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:

Scale * (tablet data + offset)

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

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

Summagraphics: 4A 03 00 00 70 00 D2 F6 90 FF

GTCO: 4A 03 00 00 0F 00 DE CC F1 FF

Section 9

UTILITY INSTRUCTIONS

The Utility instructions provide control over four system functions. These are essentially non-graphic or special functions that provide additional control over the Ω400 System.

The Utility instructions include these routines, described on the following pages:

- **o INIT - Initalize the System**
- **o SIG READ - Read Signature Analyzer Register**
- **o SYNCH - Wait for Vertical Retrace**
- **o READ CONF - Read and Return System Configuration**

INIT

Initialize the System

Instruction Format

HEX: 5E

ASCII: ↑

Input Arguments

None

Outputs

None

Description

The ↑ character (up arrow - hexadecimal 5E) is the operational code for the Initialize the System instruction. When the Ω400 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.

Initialize the System causes the Ω400 attributes to be set to the following states:

- o Pan = 0,0
- o Zoom = 0
- o Pattern Register = Solid lines, solid fills
- o Color Register = 0
- o Write Mask = All planes enabled
- o Read Mask = All planes enabled
- o Color Map = Default color selection
- o GRAFIN: Scale, Offset, and Delimiter = defaults
  (see the GRAFIN instruction)

SIG READ

Read Signature Analyzer Register

Instruction Format

HEX: 5C

ASCII: \

Input Arguments

None

Outputs

Two signature bytes

Description

The \ character (hexadecimal 5C) is the operational code for the Read Signature Analyzer Register instruction. When the Ω400 receives this instruction, it returns two bytes. These bytes represent data from an internal signature generator, and represent the data from the color map register, going to the video digital-to-analog converter.

Each display generates an individual signature. The signature received as a result of a Read Signature Analyzer Register instruction can be compared to those received during previous displays of the same image. These signatures should be identical. This function can be used during troubleshooting and system testing; it is also used internally during power-up self-test.

Special Considerations

Signatures are dependent upon Display Mode, and will change between 33 Hz mode and 60 Hz mode.

In 33 Hz mode, two signatures are returned, one for the odd field and one for the even field (33 Hz is interlaced). The Ω400 sorts them by size and sends the smaller one first.

In 60 Hz mode (non-interlaced), two bytes are also returned. However, since successive fields are identical, both signature bytes are the same.

SYNCH

Wait for Vertical Retrace

Instruction Format

HEX: 5F

ASCII: _ n

Input Arguments

n the wait count

Outputs

None

Description

The _ character (underscore - hexadecimal 5F) is the operational code for the Wait for Vertical Retrace instruction. This instruction is followed by a single byte that sets the wait count for the display synchronization. The eight bits of the wait byte specify a delay count from 0 to 255.

When the Ω400 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 will be executed after the next vertical synch pulse (0+1). If the count is set to 255, the Ω400 will wait 255 vertical synch pulses, then will execute another instruction after the next vertical synch pulse occurs (255+1).

The Wait for Vertical Retrace 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 to 255.

READ CONF

Read and Return System Configuration

Instruction Format

HEX: 5D

ASCII:]

Input Arguments

None

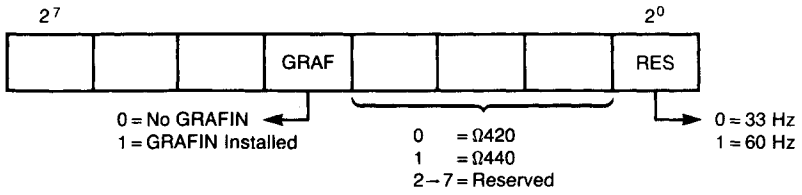
Outputs

byte 1	hardware configuration
byte 2	microcode version

Description

The] character (hexadecimal 5D) is the operational code for the Read and Return System Configuration instruction. When the Ω400 receives this instruction, it returns two bytes to the host system.

The first byte represents the hardware configuration.



The second byte represents the version of microcode installed in the Ω420. For example, an Ω440 set for 33 Hz, with version 3.5 microcode installed, would return the following (in hex):

byte 1 - 02
byte 2 - 35

Range

Does Not Apply

Section 10

INSTALLATION INSTRUCTIONS

Before Operating This Equipment

Read this section of the manual before you operate the Ω400. The installation instructions describe selection of 33/60 Hz Display Mode and GRAFIN data rate, interface installation, line voltage selection, connection to the monitor/computer/power source, and monitor adjustments when the Display Mode is changed.

CAUTION

Read and follow the installation instructions carefully. Failure to install the Ω400 properly could result in improper operation or equipment damage.

General Installation Procedure

To install the Ω400, first place the unit in a suitable location. A two-inch air space must be provided behind the unit to provide adequate air flow for cooling. After locating the unit, the general procedure for installing the Ω400 consists of the following actions:

1. Select the Display Mode.

2. Select GRAFIN data rate.

3. Install the interface, if not factory installed. (See Operator's Manual for proper interface.)

4. Check line voltage selection.

5. Connect the Ω400 to the monitor.

6. Connect the computer interface cable(s). (See Operator's Manual for appropriate interface.)

7. Connect to the power source.

Each of these procedures is described in this section, except those noted (interface-dependent information).

Note that, if the Display Mode is changed, the monitor adjustment procedure must be performed also, to obtain optimum display quality.

33/60 Hz Display Mode Selection

The Ω400 may be set for either 33 or 60 Hz frame refresh rate (Display Mode), as follows:

Remove the Ω400's top cover by removing the four screws that secure it. (See Fig. 10-1) Remove the GRAFIN and host interface boards, as shown in Fig. 10-2. The 33/60 Hz Display Mode jumpers are located in two places: under the GRAFIN board by U31, and near U291. Position the two jumpers as desired. Replace and secure the GRAFIN and host interface boards.

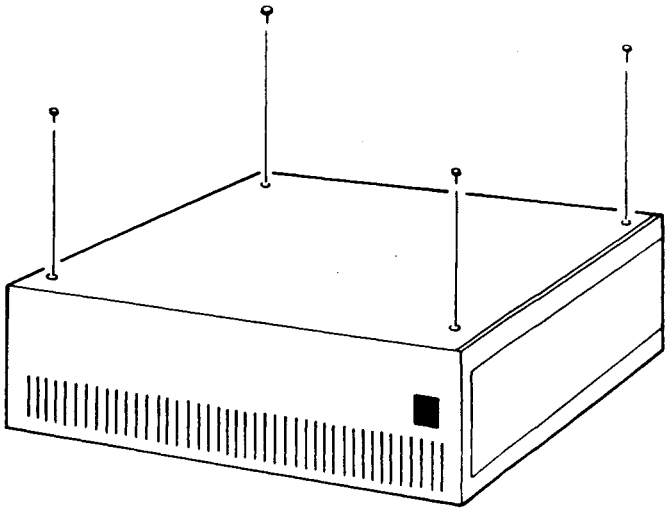


Fig. 10-1. Removing the Ω400 Top Cover

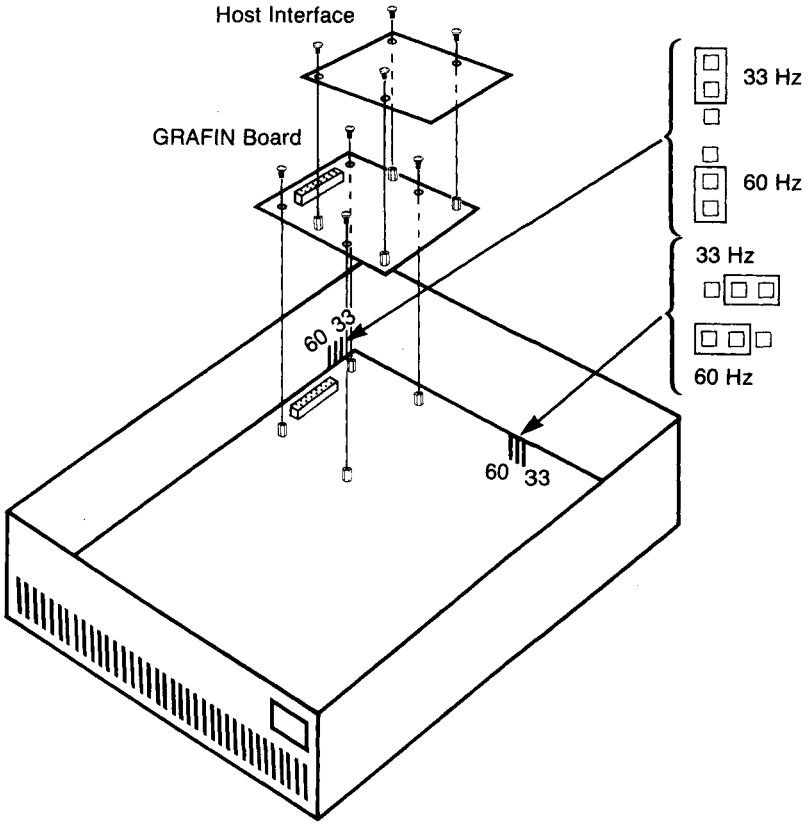


Fig. 10-2. Location of Display Mode Jumpers

Note that the monitor must also be set to match the Display Mode of the Ω400. If you are using the Ω400/GS, the instructions for setting and adjusting the monitor are located in this manual under the heading "Adjusting the Ω400/GS Monitor for 33/60 Hz Operation". If you are using the Ω400/DC with another monitor, refer to the monitor's manual for information on changing display modes.

Selecting GRAFIN Data Rate

The GRAFIN data rate is factory set for 2400 baud data transfer, but may be changed as required for a particular installation. GRAFIN data rate is selected using the four least-significant switches of the 8-switch package located near the center of the GRAFIN board (Fig. 10-3). Switch positions for the selectable transfer rates are shown in Table 10-1.

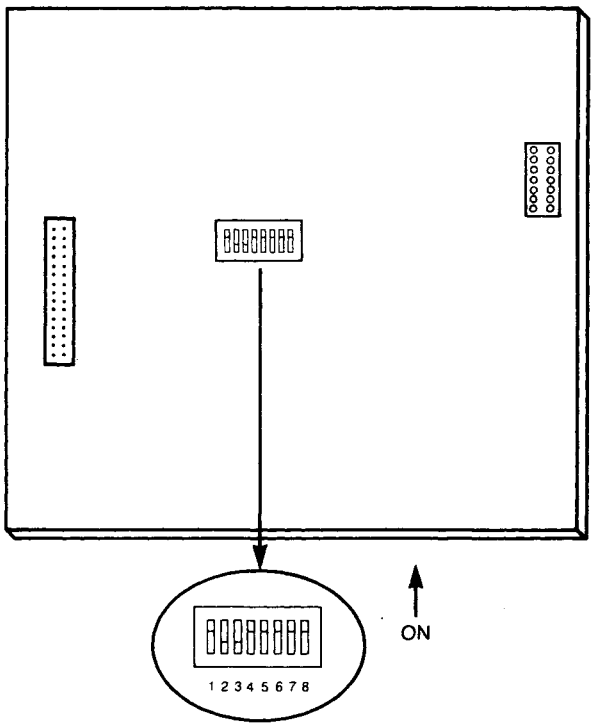


Fig. 10-3. Location of GRAFIN Data Transfer Rate Switches

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 tablet manual for compatibility information.) Switches 6, 7, and 8 should be set to 1.

Table 10-1
Data Rate Selection

baud	switches			
	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

0=Off, 1=On

After changing strap and switch positions, replace the top cover and secure it with the four attaching screws.

Installation Instructions

Installing the Interface

In most cases, the interface is installed at the factory. If it is necessary to install an interface after the Ω400 is received, refer to the interface's Operator's Manual for the interface for installation information.

Line Voltage Selection and Fuse Replacement

WARNING

Never change fuses or line voltage settings with the Ω400 connected to the power source. Always disconnect the power cord first, to prevent equipment damage and personal injury.

Changing Input Line Voltages. There are two line voltage selections available; these are 90-132 VAC and 180-264 VAC. To change line voltage selections, use the following procedure:

1. Disconnect the power cord from the AC power source.

2. Carefully turn the Ω400 over, resting it on its top surface.

3. Remove the four screws that attach the bottom cover. (See Fig. 10-4.) Remove the cover.

4. There are two power supplies located beneath the cover, -5V and +5V. The line voltage must be changed on each supply, as described in steps 5 and 6.

5. On the -5V supply (the smaller one), connect the two input jumpers to operate in the 90-132 VAC input voltage range, as shown in Fig. 10-5.

To operate in the 180-264 VAC input voltage range, only one jumper is connected, as shown in Fig. 10-5.

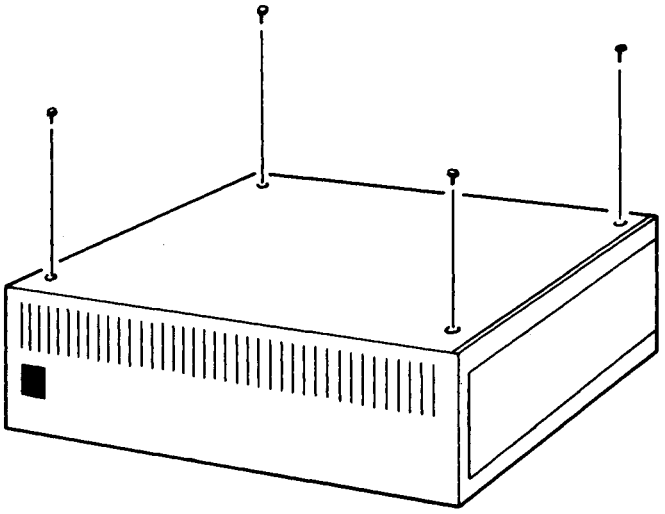


Fig. 10-4. Removing the Ω400 Bottom Cover

6. On the +5V supply, connect the input AC power to terminals 7 and 9, with a jumper from 7 to 8, for operation in the 90-132 VAC input voltage range. To operate in the 180-264 VAC input voltage range, remove the jumper between terminals 7 and 8, and connect the input AC power to terminals 7 and 8.

7. Replace the bottom cover, and secure with the four attaching screws. Turn the Ω400 back to the upright position.

IMPORTANT

Always use replacement fuses of the same value and type as the original. Follow instructions carefully.

Replacing Fuses. There is one line fuse on the back panel of the Ω400. Replace the fuse by first disconnecting power, then removing the fuse cover. Remove the fuse and replace it with one of the same value.

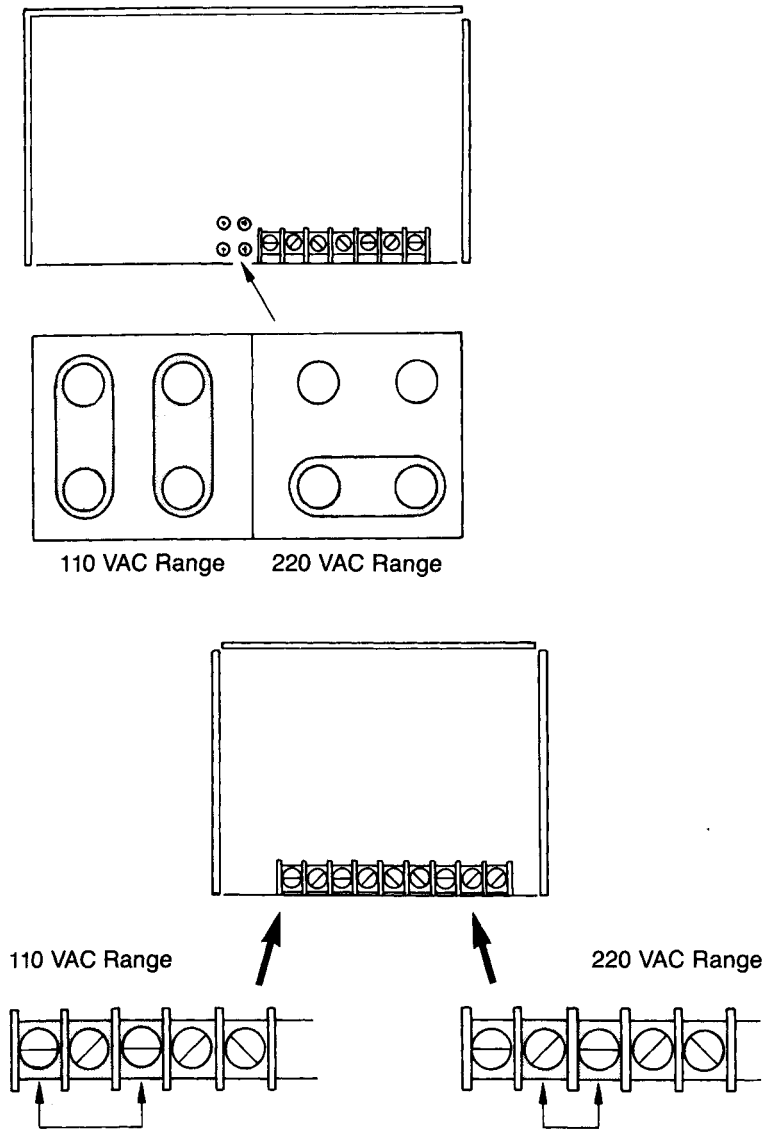


Fig. 10-5. Power Supply Jumpers

In addition to the line fuse, there is a fuse in each of the power supplies. Replace them as necessary, using fuses of the same value, as follows:

1. Disconnect the power cord from the AC power source.

2. Carefully turn the Ω400 over, resting it on its top surface.

3. Remove the four screws that attach the bottom cover. (See Fig. 10-4.) Remove the cover.

WARNING

Observe the wait period described in step 4. Voltage in the power supplies can cause personal injury if the fuse is replaced without observing the waiting period after disconnecting power.

4. Wait at least five minutes, then remove the defective fuse and replace with one of the same type and value.

5. Replace the bottom cover, and secure with the four attaching screws. Turn the Ω400 back to the upright position.

Connecting to the Monitor

There are three separate cables that connect the Ω400 to the display monitor. These connections are shown in Fig. 10-6. To connect the Ω400 to the monitor, first connect a cable from the R (red) connector on the Ω400 back panel to the Red input connector on the monitor.

Connect a second cable from the the G (green) connector on the Ω400 back panel to the Green input connector on the monitor. Connect a third cable from the B (blue) connector on the Ω400 back panel to the Blue input connector on the monitor.

Make certain that all connections are secure.

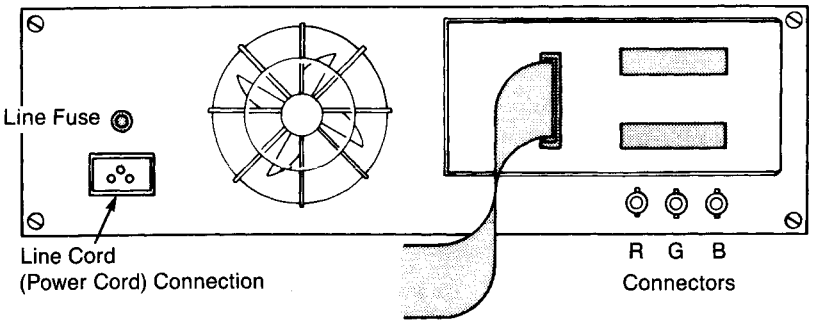


Fig. 10-6. The Ω400 Back Panel

IMPORTANT

In order to comply with FCC Class A operation requirements, fully-shielded connection cables must be used.

Connecting to the Computer

Connections to the computer are described in the Operator's Manual for the Ω400 interface that you are using. Refer to that manual for connection information.

Connecting to Power

After all installation procedures have been followed, connect a power cord to the Ω400 back panel connector, then connect to an appropriate power source. The Ω400 is ready to operate.

Adjusting the Ω400/GS Monitor for 33/60 Hz Operation

When the Display Controller's operating mode is changed from 33 Hz to 60 Hz, or vice-versa, the following adjustments must be performed on the monitor to maintain optimum display quality. This adjustment procedure applies to the Hitachi HM-3619 color monitor provided with Ω400/GS Graphics Subsystems. For further information on adjustments within the Hitachi HM-3619 monitor, refer to the manual supplied with it. Similar adjustments will be required for other types of monitors connected to Ω400/DC Display Controllers; for information on monitors other than the HM-3619, refer to the manual for the monitor.

1. Move the Horizontal Frequency Jumper on the video board to the appropriate position for the desired Display Mode (Fig. 10-7). The position marked SL1 is used for 33 Hz operation; the SL2 position is for 60 Hz.

2. Adjust the front panel's CONTRAST control (Fig. 10-8) to obtain the best display contrast.

3. On the video board, adjust the H-HOLD control to obtain horizontal stability. (See Fig. 10-7.) Then adjust the VIDEO PHASE control on the deflection board to bring the complete display back onto the screen (See Fig. 10-9).

4. Returning to the video board, adjust the H-CENT control to finish centering the display horizontally (Fig. 10-7).

5. Adjust the H-SPC control (Fig. 10-7) to eliminate jagged or bowed edges on the displayed image.

6. Adjust the V-CENT control as necessary to center the display vertically (Fig. 10-7).

7. Adjust the HEIGHT control (Fig. 10-7) to obtain the correct display size and ratio.

8. Repeat steps 3, 4, and 5 as necessary for the best display.

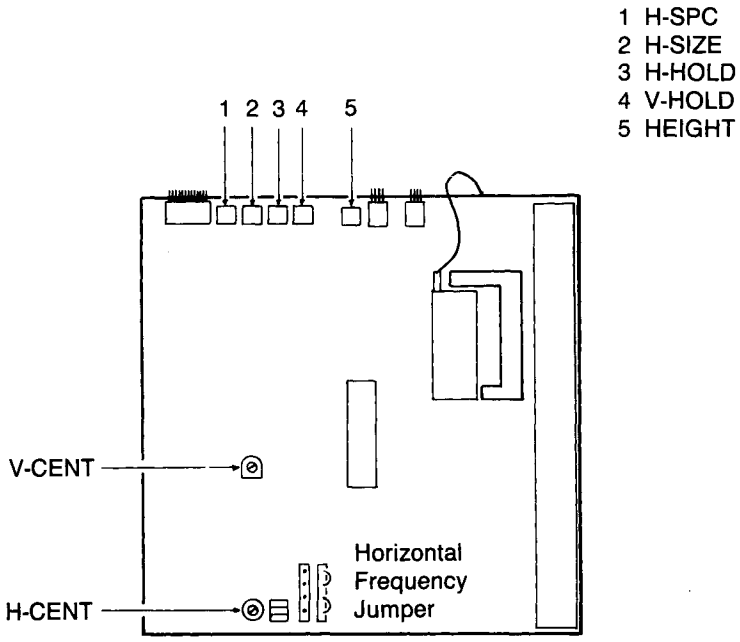


Fig. 10-7. Video Board Adjustment Locations

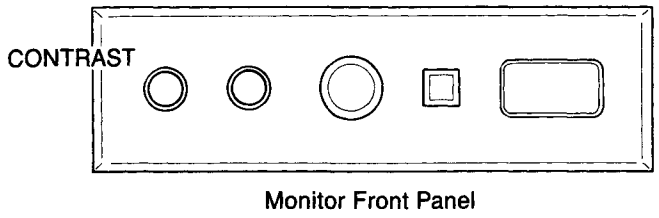


Fig. 10-8. The Contrast Control

9. Some of the above adjustments were "locked" prior to adjustment, using locking "paint." After performing this adjustment procedure, lock all previously-locked screws with an equivalent locking compound.

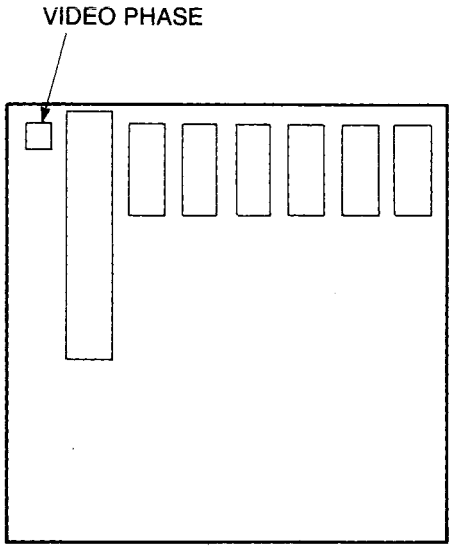


Fig. 10-9. Deflection Board Adjustment Locations

Appendix A

ASCII Code Conversion Chart

Appendix A

ASCII 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

Appendix A

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
.	46	56	2E
/	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
E	69	105	45

ASCII	Decimal	Octal	Hex
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
i	105	151	69
j	106	152	6A
k	107	153	6B

ASCII	Decimal	Octal	Hex
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