# RM-9460

# GRAPHIC DISPLAY SYSTEM

## Software Reference Manual

Publication No. 8000081-02, Revision D

June 1986

Ramtek

2211 Lawson Lane
Santa Clara, California 95050

# LIST OF EFFECTIVE PAGES

Changes, deletions, and additions to information in this manual are indicated by bars in the margins or by an asterisk near the page number if the entire page is affected.

| PAGE | | REVISION | PAGE | | REVISION |
|---|---|---|---|---|---|
| Title | | D | 2-210b | | C |
| ii | | A | 2-210c | | D |
| iii | through iv | D | 2-210d | through 2-210e | B |
| v | | C | 2-210f | | C |
| vi | through x | D | 2-210g | | D |
| 1-1 | through 1-11 | A | 2-210h | through 2-210o | B |
| 1-12 | through 1-13 | B | 2-210p | through 2-211 | C |
| 1-14 | through 1-24 | A | 2-211a | through 2-212 | D |
| 2-1 | through 2-4 | A | 2-212a | | C |
| 2-5 | through 2-10b | D | 2-213 | | D |
| 2-11 | through 2-22 | A | 2-213a | through 2-214 | C |
| 2-23 | | D | 2-215 | | A |
| 2-24 | through 2-34 | A | 2-216 | through 2-216b | C |
| 2-35 | | C | 2-217 | | A |
| 2-36 | through 2-44 | A | 2-217a | through 2-217c | C |
| 2-44a | through 2-48b | D | 2-218 | through 2-219 | A |
| 2-49 | through 2-51 | C | 2-219a | through 2-219c | C |
| 2-51a | | D | 2-220 | | A |
| 2-52 | | A | 2-221 | through 2-223a | C |
| 2-52a | | C | 2-224 | through 2-234 | A |
| 2-53 | through 2-107 | A | 2-235 | through 2-236a | C |
| 2-108 | | C | 2-237 | through 2-242 | A |
| 2-109 | through 2-115 | A | 2-243 | through 2-243b | C |
| 2-116 | | C | 2-243c | through 2-243e | D |
| 2-117 | through 2-144 | A | 2-243f | | C |
| 2-145 | | C | 2-244 | through 2-277 | A |
| 2-146 | through 2-147 | B | 2-278 | through 2-278b | C |
| 2-148 | through 2-185 | A | 2-279 | | A |
| 2-186 | through 2-187 | C | 2-280 | | D |
| 2-188 | through 2-195 | A | 2-281 | | C |
| 2-195a | through 2-195b | D | 2-281a | | D |
| 2-195c | through 2-195d | C | 2-281b | | C |
| 2-196 | through 2-198 | A | 2-282 | | D |
| 2-199 | | C | 2-283 | through 2-285 | A |
| 2-200 | through 2-201 | A | 2-286 | | D |
| 2-201a | through 2-201n | D | 2-287 | through 2-302 | A |
| 2-202 | through 2-210 | A | 2-302a | through 2-302b | C |
| 2-210a | | D | | | |

CONTENTS

Paragraph Page

## Appendix A. CONICS



## Appendix B. VIDEO GENERATORS



## Appendix C. INTERACTIVE PERIPHERALS

# FIGURES

# Chapter 1

# SYSTEM CONCEPTS

## 1.1 SYSTEM OVERVIEW

The RM-9460 Series Graphic Display System (figure 1-1) is a raster-scan display system compatible with either monochrome or color CRT monitors and large screen projectors. The system consists of a rackmount chassis, power supply, and solid-state electronics. The RM-9460 is capable of single or multichannel operation; it may be configured as an output peripheral or as an on-line interactive display system.

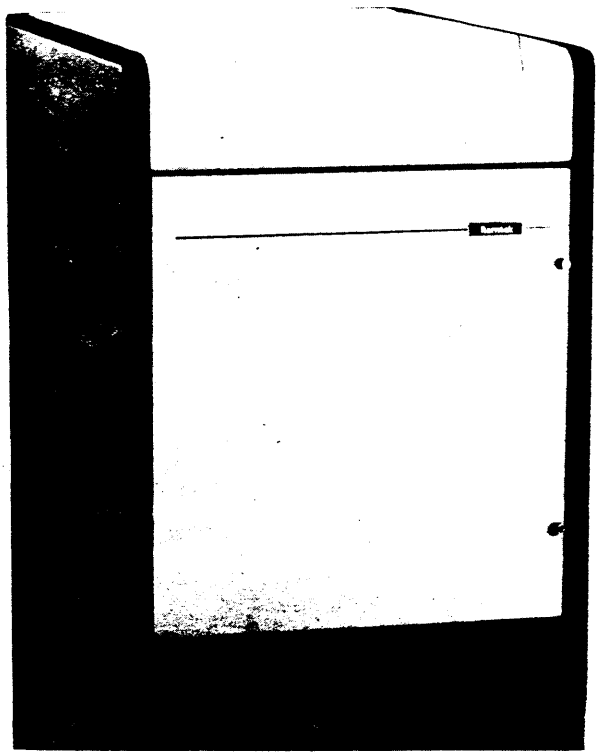


Figure 1-1. RM-9460 Series Graphic Display System

### 1.2.1 System Concepts

Chapter 1 functionally describes hardware elements and relates how these elements accomplish graphics and imaging techniques. The features of the RM-9460 system are elaborated and programming considerations are set forth.

### 1.2.2 Command Operands and Instructions

Chapter 2 provides a detailed discussion of command operands and instructions. Instructions processed by the RM-9460 and their page numbers are listed in tables 2-9 and 2-10.

### 1.2.3 Interface

Chapter 3 supplies hardware and software information needed to interface the RM-9460 to a host computer.

### 1.2.4 Coordinate System and Transformation Concepts

Chapter 4 describes the coordinate system and the associated digital and video addressing transformation concepts.

### 1.2.5 Conics

Appendix A describes three families of conics: the ellipse, the parabola, and the hyperbola.

### 1.2.6 Video Generators

Appendix B describes video generators encountered in RM-9460 systems.

### 1.2.7 Interactive Peripherals

Appendix C describes supporting peripherals and the ramifications of programming when these devices are supplied.

### 1.2.8 Instruction Timing

Appendix D describes the execution times of the RM-9460 instruction set.

### 1.2.9 Error Codes

Appendix E lists the error codes for the RM-9460 instruction set.

### 1.2.10 Glossary

Appendix F lists the glossary terms for the RM-9460 Software Reference Manual.

### 1.2.11 Index

Appendix G lists manual topics cross referenced to page numbers.

depending upon character size. The refresh memories incorporate a bulk erase feature that erases any combination of memory bit planes during a single refresh cycle (16, 20, 34, or 40 microseconds depending upon resolution and refresh frequency). This feature reduces response time when a new picture or perspective is to be displayed.

Interactive video functions include translate (through video lookup table), pan (of the refresh memories), and zoom (by pixel replication in integer steps between 1 and 16).

The optional serial link PCB contains a microprocessor that processes keyboard input/output, cursor control input, and cursor output. Interactive devices supported by the serial link/cursor option include a keyboard, joystick, trackball, light pen, and digitizing tablet.

### 1.3.1 Computer Interface

The computer interface provides a high-speed link between the host computer and the RM-9460 Graphic Display System. A general purpose TTL-compatible interface (GPIF) is provided on the display processor. The data exchange sequence is software controlled. One additional card slot is reserved for custom interfaces. Here, the data exchange sequence is hardware controlled. All interfaces are 16-bit parallel. Most incorporate or utilize direct memory access in the host computer.

### 1.3.2 DMA Sequencer

The TTL DMA sequencer performs high-speed, nonprocessor data transfers between multiple devices on the system bus, such as the computer interface and the display processor or memory control processor. The DMA sequence can involve as many as 14 device ports and 7 subloops. For example, the DMA sequencer can send one word from Port A to Port B, then six words from Port C to Port D before the sequence is repeated.

### 1.3.3 Display Processor

The display processor (also called the system processor PCB) controls each processing element within the graphic display system. It also decodes received instructions, stores subpictures (command lists) and fonts, performs coordinate transformations, and drives the memory control processor. The display processor can contain either a Z80 or an MC68000 microprocessor.

The Z80 system processor has 32K bytes each of EPROM and RAM, a GPIF interface, three serial ports, a timer, memory map, cycle-stealing DMA controller, and interrupt control logic. The memory map accommodates up to 512K memory bytes. When the Z80 microprocessor is installed on the system processor PCB, the processor expansion PCB is needed to perform coordinate transformations.

The MC68000 system processor has 128K bytes of EPROM and 256K bytes of RAM (of which 240K bytes are user RAM, a GPIF interface, three serial ports, a timer, cycle-stealing DMA controller, and interrupt logic.

### 1.3.7 Video Generators

The video generators transform the stored pictures into video signals that drive Ramtek and other CRT monitors, large screen projectors, and hardcopy printers.

Video generators process data on a pixel-by-pixel basis through PROM/RAM-defined lookup tables that assign output color and/or intensity. Each pixel indexes the lookup tables as the data is scanned from refresh memory. The contents of the addressed cell in the lookup table are passed to the digital-to-analog converters (DACs) and/or video amplifiers that produce the output video signals.

### 1.3.8 Serial Link PCB (Optional)

The optional serial link PCB processes operator input from keyboards and graphics input devices and generates cursors that can be moved about the face of the screen without affecting the data in refresh memory. It contains its own Z80 microprocessor with dedicated EPROM and RAM, four or eight serial ports, and two or four generators. The serial link PCB generally is not user programmable; however, support software is available for keyboard, trackball, light pen, and graphic tablet devices. The serial link PCB provides up to four keyboard ports, four cursor control ports, and four cursor generators. The cursor symbol is user programmable within a 32 X 32 pixel matrix.

## 1.4 CONCEPTS AND FEATURES

The RM-9460 system introduces many new features to raster scan display technology. The concepts upon which these features are based include context switching, internal display list processing, coordinate transformation, decluttering, and entity detection. The dual bus, dual microprocessor, architecture of the system and its support logic are specifically designed to perform these tasks. Figure 1-3 is a conceptual view of the RM-9460. As shown, graphics data can be defined by picture coordinates that are automatically transformed to display coordinates as the picture is processed from the display list to the refresh memory. Picture (local) coordinates differ from display (global) coordinates in that pictures may be defined against an arbitrary reference frame and then projected into display coordinate space. In this manner, pictures or picture segments may be enlarged, reduced, scaled, translated, and rotated without changing the display list that describes the picture of objects being displayed. Because the display coordinate space is much larger than the refresh memory, parts of extremely large or detailed pictures may be displayed, that is, a panning effect is created by repeating the display list after registering the refresh memory against the display coordinate space. Once stored in refresh memory, the picture may be enlarged further by pan and zoom hardware.

The display list may reside in either the host computer or the RM-9460 display list memory. One advantage of local storage is that it need not be retransmitted from the host computer as the perspective of the picture is changed. Another advantage is that any key or cursor movement may be linked to locally-stored display lists. For example, a complete menu may be displayed as the result of striking a single function key, or a predefined

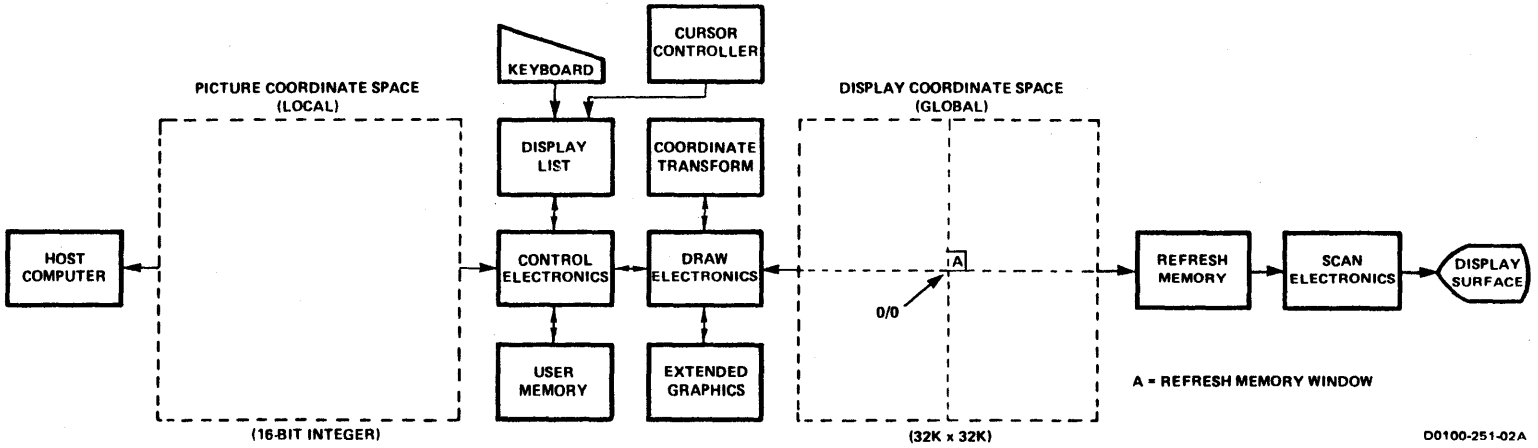



Figure 1-3. RM-9460 Graphing Concepts

### 1.4.3 Spatial Transformations

Figure 1-5 illustrates the spatial transformations that are performed by the system. These include internal coordinate transformation from picture coordinates to display coordinates, registration of the refresh memories against the larger display coordinate space, clipping, pan, and zoom. The pan and zoom features of the system are highly interactive functions that project the pictures from the bit-per-element refresh memories on to the screen surface(s) at video rates. Coordinate transformation, registration, and clipping functions consume more time because the picture is reprocessed from the display list and redrawn into the refresh memory.

**a.** Coordinate Transformations Internal coordinate transformations (figure 1-6) translate, rotate, and scale the image as picture segments are processed from the display list to the refresh memory. As shown in figure 1-5, the transformation occurs as the image is projected from the picture coordinate space to display coordinate space. Each is a virtual picture space in that no memory exists. Instead, the picture is stored in terms of graphic endpoints in the display list and bit-per-element raster data in the refresh memory, with the scan conversion being performed by the draw electronics. The coordinate transformation firmware adds logic to the draw electronics.

The translation feature of the display system allows pictures or picture segments to be projected to different points within the display coordinate space. For example, a single resistor, as defined within the display list, might be drawn at several locations within display coordinate space. The rotation feature allows these same objects to be rotated in one-degree increments.

Finally, the scaling feature allows these objects to be enlarged or reduced up to 255 times their original size. The coordinate transformation firmware controls picture or picture segment position, rotation, and size.

**b.** Registration. Whether or not coordinate transformations are performed, the refresh memory may be registered against the larger display coordinate space. While display increments in the display coordinate space are equivalent to pixels in the refresh memory, the display coordinate space is much larger than the refresh memory. Therefore, the refresh memory may pan about the display coordinate space as the display list is reprocessed by the display system.

**c.** Clipping The clipping feature (figure 1-7) constrains images to specified viewports (or windows) in the refresh memory. In this way the picture may be limited to particular areas on the display surface regardless of scale so that surrounding areas on the screen surface may be used for annotation or other purposes.

**d.** Decluttering. Decluttering provides visibility control for command strings within the display list(s) by comparing modal display class operands which appear in the display list against a table of displayable classes that is established by a command external to the display list. Display class is defined by a 16-bit integer value, while displayable classes are defined by 1 to 16 ranges of these values. Because decluttering implies controlled visibility as a function of scale, displayable classes typically are set to

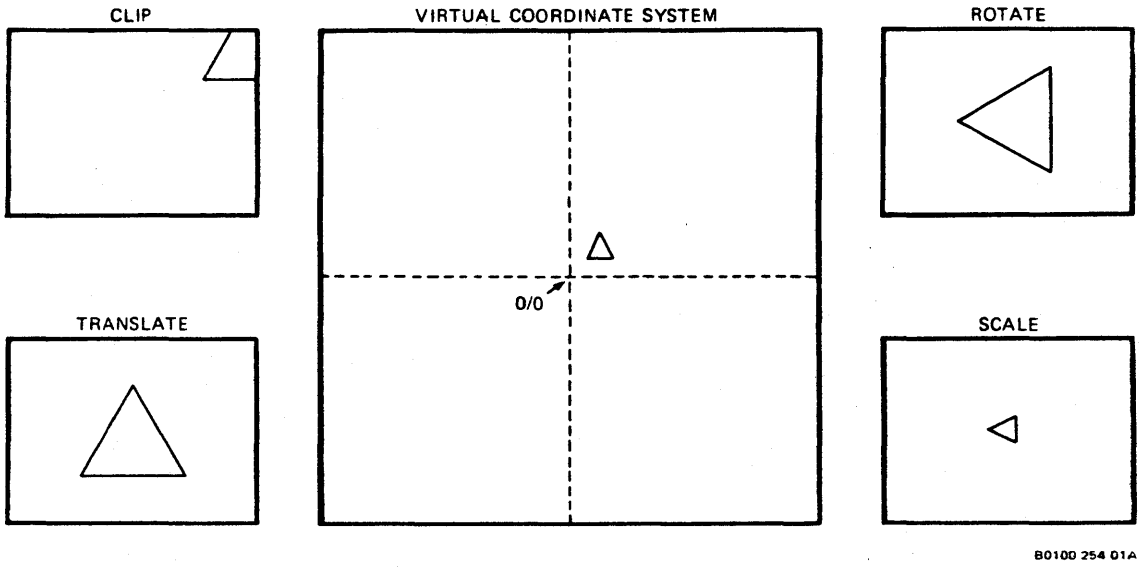



Figure 1-6. Coordinate Transformation

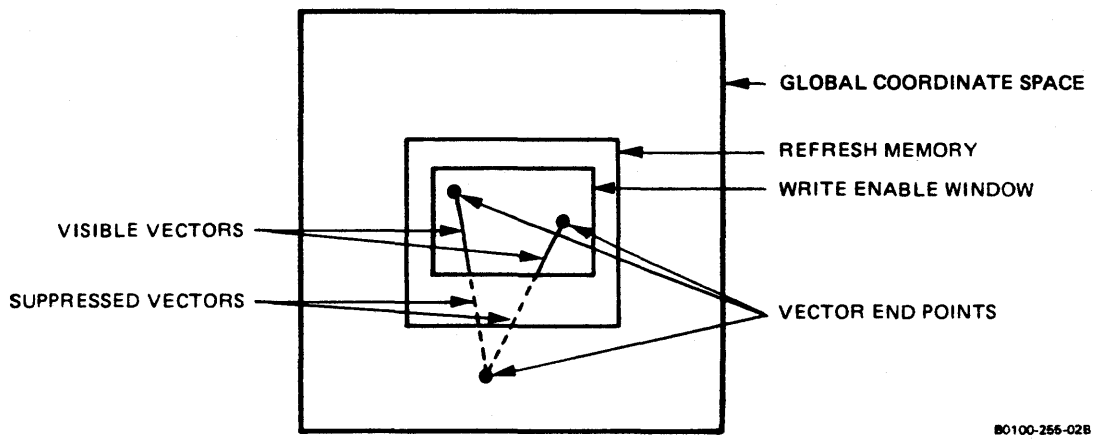



Figure 1-7. Clipping

indicate the minimum scale (or magnification ratio) at which information should become visible.

### 1.4.4 Entity Detection

The entity detect feature identifies graphics procedures and instructions that attempt to draw information within a specified target area on the display surface. The target area typically is established by cursor position; for example, using a cursor-control device such as a trackball, joystick, or light pen, the operator points the cursor at the object to be detected and interrupts the host computer. The host computer responds by enabling entity detect

Table 1-2. Programming Features

| FEATURE | DESCRIPTION |
|---|---|
| Primitives | Erase, point, solid rectangle, vector, conic, circle, arc, solid polygon (fill); text and special symbol, raster, plot, bar chart, and continuous tone image (8 or 16 bits/pixel). |
| Attributes | Foreground and background color/intensity, line texture, character size and orientation, window, reverse background, and additive write. |
| Fonts | Standard font containing 64 symbols defined within 7 x 9 element dot matrix. Up to sixteen additional 128 character fonts may be down-line loaded. Maximum font resolution is 16 x 20 elements; symbols may be enlarged by pixel replication. |
| Coordinate Transformations | Translate (two dimension), rotate, and scale. |
| Addressing Modes | Absolute, relative, and indexed. Separate instructions also provide for transfer of cursor coordinates to pen position (COP) or index registers. |
| Picture Resolution | 32K x 32K virtual picture. |
| Screen Resolution | 1024 x 1280. |
| Video Functions | Pan, zoom (in integer steps between 1:1 and 16:1), blink, and color/intensity translation. |
| Subpictures | Stored locally as graphic subroutines in 4K to 16K byte memory segments. |
| Declutter | Controls visibility of graphic procedures and instructions; usually as a function of scale. |
| Entity Detect | Identifies procedures and instructions that attempt to generate data within a prescribed rectangular window (target). |
| Clip Window | Constrains pictures to arbitrary viewports on the display surface. |
| Format Window | Begins a new line or page when the present line or page is filled to capacity while writing font, raster, or image data. |
| Rotation | Graphic rotation in one degree increments; font, raster, and image data rotation in 90 degree increments. |

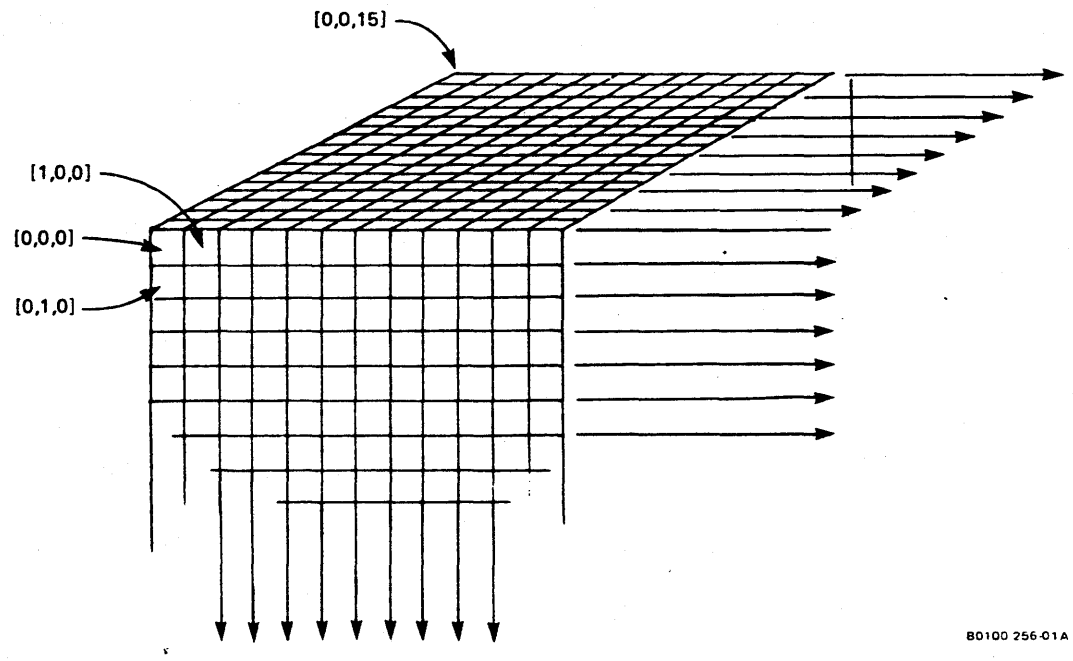



Figure 1-8. Classical X-Y-Z Coordinate System

These categorize all data that is transferred to the RM-9460 through a normal-format instruction directly or indirectly into display refresh memory. Image data is transferred directly into refresh memory. Text, raster, and graphics data are transferred indirectly in the sense that data is interpreted by the RM-9460 microprocessor firmware and generated to refresh memory as a result of this interpretation.

**a.** Image Data. Image data is loaded directly into refresh memory using the WRITE IMAGE normal-format instructions. Each 16-bit image data word is loaded into the refresh memory associated with successive picture elements (pixels) in the system; that is, one image data word or byte is written to one display pixel. Image data is always written into a rectangular region (pixel array) within display refresh memory; this region, or window, is defined by a parameter operand called FORMAT WINDOW that may be set in any normal-format instruction. The pixel-to-pixel updating direction for successive words of image data is defined by the parameter operand SCAN, which also defines the action to be performed at the window boundaries. Image data can also be written to arbitrary pixels using the WRITE RANDOM PIXEL normal-format instruction.

Image data in standard configurations defines the displayed image in one of two ways. In a Type I Video standard configuration, each subchannel (bit) of refresh memory can be displayed separately to a black-and-white CRT monitor. In a Type II Video standard configuration, the low order 11 (or 12) subchannels are used as an address into a video lookup table (VLT) that contains the color definition information. Any pixel that has the same data value in the

## 1.6 DISPLAY INTERACTION

The RM-9460 can be configured to interact with the host processor through several devices. Up to eight keyboards and eight cursor controllers are possible in a system.

## 1.7 INTERACTIVE PERIPHERALS

Joystick, trackball, light pen, and graphic tablet controllers are steering devices that can automatically change the position of a cursor on the display CRT monitor when connected to the serial link PCB. The joystick or trackball interrupts the host processor when a momentary-action switch (ENTER) on the device is depressed or whenever the position (TRACK) of the display cursor is changed. This interrupt can be used by the host processor to signal some action to be taken, based on the position of the cursor. The interpretation is completely flexible, based on the needs of the host processor.

The keyboard transmits eight-bit ASCII codes from the RM-9460 to the host processor. The keyboard is attached to the serial link card or to the processor card and is handled by the microprocessor. Each keyboard is buffered on input up to 16 characters. Entry of a character through the keyboard generates an interrupt request to the host processor if interrupts have been enabled at the interface by the host processor. The keyboard allows user interaction through text input. (See appendix C for further information on interactive peripherals.)

## 1.8 INSTRUCTION FORMATS

The RM-9460 processes three categories of instructions that differ in format. The first includes those instructions that normally are used to draw information into the refresh memories and therefore onto the display screen(s). Called normal-format instructions, they are application-oriented, macro-level commands whose functions vary, depending upon operands provided as arguments of calls to the host-resident graphics package or I/O driver. These arguments include channel selection, color or intensity, scale, line texture, etc.

The second category consists of commands that are more closely machine-oriented. These instructions perform such functions as loading programmable fonts and video lookup tables. They are termed special-format instructions because the instruction format varies according to the function being performed.

The third category of instructions is user-defined. Here, the user downline-loads control software into the RM-9460 to interpret instructions. For example, these instructions directly interpret the host computer's data base or perform other high-level, application-oriented functions. These are called user-format instructions.

### 1.8.1 Normal-Format Instruction

The normal-format instruction (figure 1-9) is extremely flexible. Operands and data are optional in any given instruction; many instructions contain a single 16-bit control word.

AD - when nonzero, selects additive write mode which suppresses the writing of zero state font, raster and graphics data into refresh memory. When zero, replacement mode is selected.

BK - when nonzero, inverts the polarity of font, raster and graphics data being generated into the refresh memory. Thus, foreground and background color/intensity are reversed.

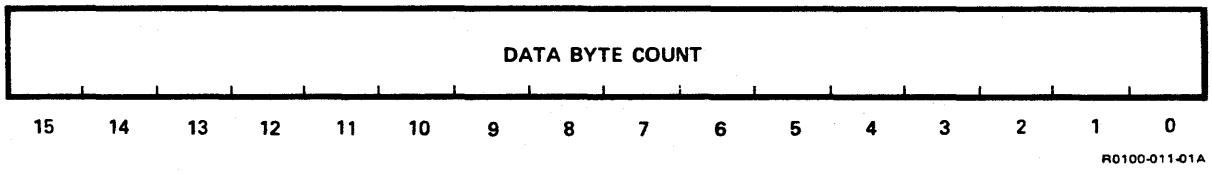
RP - when nonzero, reverses the order in which byte-oriented data is unpacked and processed. The least significant information byte is processed first.

OF1 and OF2 - when nonzero, indicates the presence of corresponding operand flag words in the instruction format.

DF - when nonzero, indicates the presence of the DATA BYTE COUNT word in the instruction format.

**b.** Operand Flag Words. Detailed descriptions of operand flag words and operand parameters are contained in chapter 2.

**c.** Data Byte Count Word.

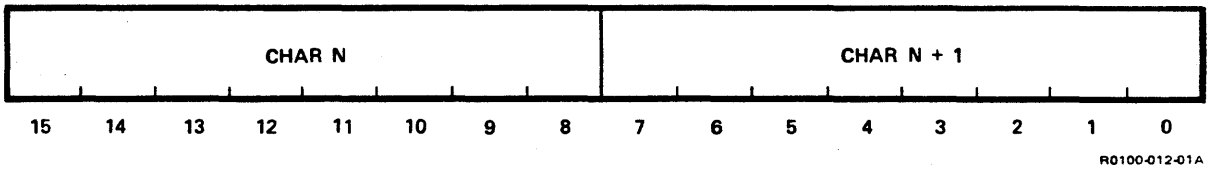



The DATA BYTE COUNT word specifies the number of data bytes to be transferred to or from the display system.

### 1.8.2 Data Formats

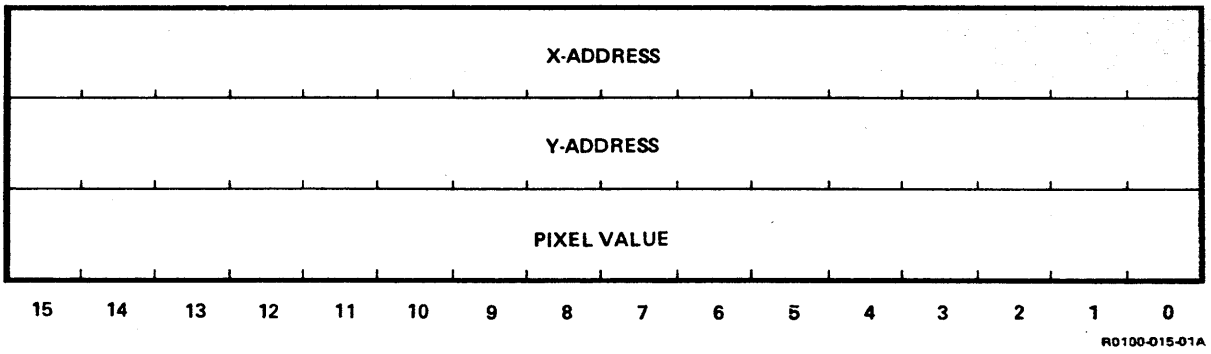
Data elements are either byte, word, or multiword oriented, depending on the function being performed by the display system. Following are descriptions of the various data types and the applicable instruction arguments and control bits.

**a.** Font Data (Character or Symbol).



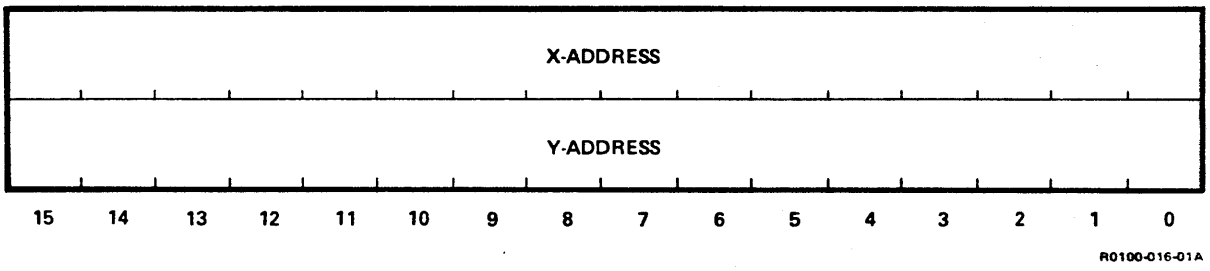
units from scanning devices such as television cameras or scanners (IR, X-ray, radar, etc.). Data is loaded sequentially into a pixel array defined by the FORMAT WINDOW and SCAN operands. New lines of data are started at the near boundary after the far boundary is reached. Any of eight possible scan sequences may be specified. An image being loaded may be logically or arithmetically combined with an image already contained in refresh memory as defined by the FUNCTION operand. The reverse packing (RP) control bit defines the order of byte processing.

**d.** Random Point Data.




Each multiword data element describes a randomly-located picture element in terms of X-Y coordinate address and color and/or intensity, as translated by a video lookup table.

**e.** Graphics Endpoint Data.




Each multiword data element describes a single graphics endpoint in terms of an X-Y coordinate address. Graphics instructions draw lines between sequential endpoints. For example, a polyline is specified by a single startpoint and a series of endpoints. The display system processes this data by drawing vectors between the startpoint and first endpoint, the first and second endpoints, the second and third endpoints, etc. Disconnected lines are specified by a separate instruction that interprets alternate endpoints as new startpoints. For example, vectors are drawn between the first and second endpoints, the third and fourth endpoints, the fifth and sixth endpoints, etc.

# Chapter 2

# COMMAND OPERANDS AND INSTRUCTIONS

## 2.1 OPERAND FLAG WORDS

Operand flag words indicate the status of the 22 currently-defined parameter operands for all normal-format instructions (see figure 2-1). There are two operand flag words (1 and 2) flagged in the parameter byte by OF1 and OF2, respectively. Operand flag word 1 (figure 2-2) flags the first 16 parameter operands for the RM-9460 and operand flag word 2 (figure 2-3) flags the 6 RM-9460 extension operands. Each bit flags a particular parameter operand. A zero bit indicates that the corresponding parameter operand does not exist. Each bit is interpreted from bit position 0 to bit position 15, with the bits in operand flag word 1 (if OF1 = 1) being evaluated before the bits in operand flag word 2.

## 2.2 PARAMETER OPERANDS

Parameter operands are the various internal values that further define the operation for all normal-format instructions. Any set or subset of parameter operands is indicated by the status of the 22 bits in the operand flag words. Format of the operand flag words is defined in figure 2-1. The default values for all parameters are defined in table 2-1. The effects of the coordinate transformations on the normal-format parameters are listed in table 2-2. The use of parameters in instructions is detailed in table 2-3.

**OPERAND FLAG WORD 1**

| COP | SCR | BAS | CON | LAF | SIZ | SPC | DIM | SCN | WIN | ORG | IX 2 | IX 1 | BGD | FGD | WMSK |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| OPERAND WORD | BIT | MNEMONIC | FUNCTION |
|---|---|---|---|
| 1 | 0 | WMSK | WRITE MASK |
| 1 | 1 | FGD | FOREGROUND |
| 1 | 2 | BGD | BACKGROUND |
| 1 | 3 | IX 1 | INDEX 1 |
| 1 | 4 | IX 2 | INDEX 2 |
| 1 | 5 | ORG | ORIGIN |
| 1 | 6 | WIN | FORMAT WINDOW |
| 1 | 7 | SCN | SCAN |
| 1 | 8 | DIM | DIMENSIONS |
| 1 | 9 | SPC | SPACING |
| 1 | 10 | SIZ | SIZE |
| 1 | 11 | LAF | LOGICAL/ARITHMETIC FUNCTION |
| 1 | 12 | CON | CONIC EQUATION |
| 1 | 13 | BAS | BASELINE |
| 1 | 14 | SCR | SCROLL COUNT |
| 1 | 15 | COP | START POINT |

B0100 260 01A

Figure 2-2. Operand Flag Words Format - Word 1

**OPERAND FLAG WORD 2**

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | IMG | DCL | VTX | WOF | WEW | RMSK |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| OPERAND WORD | BIT | MNEMONIC | FUNCTION |
|---|---|---|---|
| 2 | 0 | RMSK | READ MASK |
| 2 | 1 | WEW | WRITE-ENABLE WINDOW |
| 2 | 2 | WOF | WRITE-ENABLE WINDOW OFFSET |
| 2 | 3 | VTX | VECTOR TEXTURE |
| 2 | 4 | DCL | DISPLAY CLASS |
| 2 | 5 | IMG | IMAGE MODE |

B0100 261 01A

Figure 2-3. Operand Flag Words Format - Word 2

Table 2-2. Specifications of Normal-Format Parameters

| NAME | DATA TYPE | COORDINATE SYSTEM | TRANSFORMED |
|---|---|---|---|
| WRITE MASK | MASK | N/A | N/A |
| FOREGROUND | MASK | N/A | N/A |
| BACKGROUND | MASK | N/A | N/A |
| INDEX 1 | COORDINATES | LOCAL | YES |
| INDEX 2 | COORDINATES | LOCAL | YES |
| ORIGIN | COORDINATES | REFRESH | NO |
| FORMAT WINDOW | COORDINATES | LOCAL | YES |
| SCAN | MODE | N/A | N/A |
| DIMENSION | COORDINATES | REFRESH | NO |
| SPACING | COORDINATES | REFRESH | NO |
| SIZE | VALUES | N/A | N/A |
| FUNCTION | MODE | N/A | N/A |
| CONIC | COEFFICIENTS | N/A | N/A |
| BASELINE | COORDINATES | LOCAL | YES |
| SCROLL | VALUES | REFRESH | NO |
| START-POINT | COORDINATES | LOCAL | YES |
| READ MASK | MASK | N/A | N/A |
| WRITE-ENABLE WINDOW | COORDINATES | GLOBAL | NO |
| WRITE-ENABLE WINDOW OFFSET | COORDINATES | REFRESH | NO |
| TEXTURED LINES | VALUES | N/A | N/A |
| DISPLAY CLASS | VALUES | N/A | N/A |
| IMAGE MODE | MODE | N/A | N/A |

Table 2-3. RM-9460 Instruction Repertoire (Continued)

| OP CODE | INSTRUCTION | MNE-MONIC | KEY F | KEY P | IX | AD | BK | RP | OF2 | OF1 | DF | COP | SCR | BAS | CON | LAF | SIZ | SPC | DIM | SCN | WIN | ORG | IX2 | IX1 | BGD | FGD | WMSK | IMG | DCL | VTX | WOF | WEW | RMSK |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 | LOAD PROGRAMMABLE FONT REVERSE PACKING | LPFRP | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | SELECT MCP/GROUP | SELMG | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 23 | SET DISPLAY CLASS RANGES | SETDC | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 24 | READ NORMAL PARAMETERS | READP | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 | READ ERROR STATUS | RERR | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 26 | ZOOM | ZOOM | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 27 | SELECT VIDEO ORIENTATION | SELVO | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 28 | LOAD SUBCHANNEL ORIGIN | LOADSO | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 29 | WAIT FOR VERTICAL RETRACE | WAITVR | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2A | WAIT FOR VIDEO LINE | WAITL | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2B | BULK ERASE | BERS | N | - | X | X | X | S | X | X | S | S | S | S | S | S | S | S | S | S | S | S | S | S | X | X | X | S | S | S | S | S | S |
| 2C | WRITE CURSOR STATE PIXEL | WCSP | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0C | WRITE CURSOR STATE GLOBAL | WCSG | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2E | READ CURSOR STATUS PIXEL | RCSP | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2F | READ CURSOR STATUS GLOBAL | RCSG | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30 | LOAD CURSOR FONT | LCF | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | LOAD CURSOR TO COP/INDEX/ORIGIN | LCCIO | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 32 | CENTER WINDOW AT CURSOR | CWC | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 33 | DEFINE DEVICE/CURSOR CONFIGURATION | DDCC | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 34 | DEFINE CURSOR/VIDEO CONFIGURATION | DCVC | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 35 | WRITE VECTOR UNLINKED | WVU | N | - | X | X | X | S | X | X | X | - | S | S | S | S | S | S | S | S | S | S | X | X | X | X | X | S | X | X | X | X | S |
| 36 | WRITE PLOT POINT | WPP | N | - | X | X | X | S | X | X | X | X | S | S | S | S | S | X | S | X | S | S | X | X | X | X | X | S | X | S | X | X | S |
| 37 | WRITE PLOT VECTOR | WPV | N | - | X | X | X | S | X | X | X | X | S | S | S | S | S | X | S | X | S | S | X | X | X | X | X | S | X | X | X | X | S |
| 38 | WRITE POINT | WPT | N | - | X | X | X | S | X | X | X | - | S | S | S | S | S | S | S | S | S | S | X | X | X | X | X | S | X | S | X | X | S |
| 39 | WRITE RANDOM PIXEL | WRP | N | - | X | S | S | S | X | X | X | - | S | S | S | S | S | S | S | S | S | S | X | X | S | S | X | S | X | S | X | X | S |
| 3A | FILL | FILL | N | - | X | X | X | S | X | X | X | X | S | S | S | S | S | S | S | S | S | S | X | X | X | X | X | S | X | S | X | X | X |
| 3B | ALLOCATE DISPLAY LIST | ALDL | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3C | DEALLOCATE DISPLAY LIST | DEDL | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3D | RESET CONTEXT | RESCON | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3E | WRITE KEYBOARD BLOCK | WBLK | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3F | WRITE KEYBOARD BLOCK REVERSE PACKING | WBLKRP | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 40 | READ TABLET COORDINATES | RTC | S | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

X0100-326#2-02D

Table 2-3. RM-9460 Instruction Repertoire (Continued)

| OP CODE | INSTRUCTION | MNE-MONIC | KEY F P | IX | AD | BK | RP | OF2 | OF1 | DF | COP | SCR | BAS | CON | LAF | SIZ | SPC | DIM | SCN | WIN | ORG | IX2 | IX1 | BGD | FGD | WMSK | IMG | DCL | VTX | WOF | WEW | RMSK |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 61 | SET DETECT PARAMETERS | SDP | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 62 | ENABLE DETECT | ED | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 63 | DISABLE DETECT | DD | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64 | SUSPEND DETECT | SD | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 65 | RESUME DETECT | RD | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 66 | SET DETECT DATA | SDD | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 67 | SENSE DETECT STATUS | SDS | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 68 | READ BACK DETECT BUFFER | RDB | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 69 | COPY IMAGE | COPY | N - | X | S | S | X | X | X | X | X | S | S | S | X | S | S | S | X | X | S | X | X | S | S | X | X | X | S | X | X | S |
| 6A | COPY IMAGE TRIGGERED | COPYT | N - | X | S | S | X | X | X | X | X | S | S | S | X | S | S | S | X | X | S | X | X | S | S | X | X | X | S | X | X | S |
| 6B | COMBINE IMAGE | COMBI | N - | X | S | S | X | X | X | X | X | S | S | S | X | S | S | S | X | X | S | X | X | S | S | X | X | X | S | X | X | S |
| 6C | COMBINE IMAGE TRIGGERED | COMBT | N - | X | S | S | X | X | X | X | X | S | S | S | X | S | S | S | X | X | S | X | X | S | S | X | X | X | S | X | X | S |
| 6D | WRITE CIRCLE | CIRC | N - | X | X | X | S | X | X | X | S | S | S | S | S | S | S | S | S | S | S | X | X | X | X | X | S | X | S | X | X | S |
| 6E | WRITE ARC TYPE I | ARC1 | N * | X | X | X | S | X | X | X | S | S | S | S | S | S | S | S | S | S | S | X | X | X | X | X | S | X | S | X | X | S |
| 6F | WRITE ARC TYPE 2 | ARC2 | N * | X | X | X | S | X | X | X | S | S | S | S | S | S | S | S | S | S | S | X | X | X | X | X | S | X | S | X | X | S |
| 70 | WRITE ARC TYPE 3 | ARC3 | N * | X | X | X | S | X | X | X | S | S | S | S | S | S | S | S | S | S | S | X | X | X | X | X | S | X | S | X | X | S |
| 71 | SET LOCAL FUNCTION STATE | SLFS | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 72 | SET LOCAL KEYBOARD FUNCTION | SLKF | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 73 | CLEAR LOCAL KEYBOARD FUNCTION | CLKF | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 74 | SET LOCAL CURSOR FUNCTION | SLCF | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 75 | CLEAR LOCAL CURSOR FUNCTION | CLCF | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 76 | SET KEYBOARD ECHO STATE | SKES | S - | | X | X | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 77 | CLEAR KEYBOARD ECHO STATE | CKES | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 78 | ALLOCATE TREND | ALTD | S T | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 79 | DEALLOCATE TREND | DETD | S T | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7A | INIT TREND | ITD | S T | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7B | LOAD TREND PATTERNS | LTDP | S T | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7C | UPDATE TREND | UDTD | S T | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7D | ERASE TREND | ERSTD | S T | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7E | DISPLAY TREND | DISTD | S T | | X | X | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7F | WRITE PACKED IMAGE | WPI | N † | X | S | S | S | X | X | X | X | S | S | S | S | S | S | S | S | X | X | S | X | X | S | S | X | S | X | S | X | X | S |
| 80 | WRITE IMAGE VECTORS | WIV | N † | X | S | S | S | X | X | X | S | S | S | S | S | S | S | S | S | S | S | S | X | X | S | S | X | S | X | S | X | X | S |

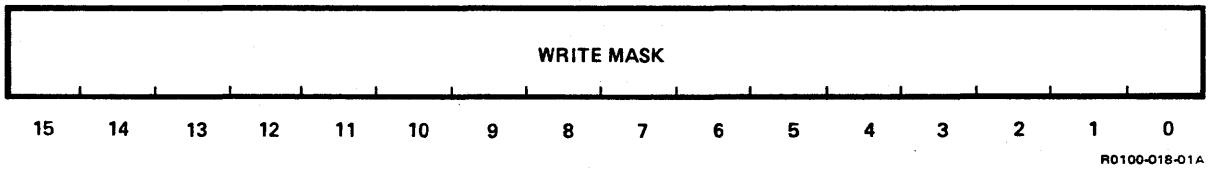
R0100-326#4-02D

Table 2-3. RM-9460 Instruction Repertoire (Continued)

| OP CODE | INSTRUCTION | MNE-MONIC | KEY F P | IX | AD | BK | RP | OF2 | OF1 | DF | COP | SCR | BAS | CON | LAF | SIZ | SPC | DIM | SCN | WIN | ORG | IX2 | IX1 | BGD | FGD | WMSK | IMG | DCL | VTX | WOF | WEW | RMSK |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A2 | SET ERRORS | SETER | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A3 | SET DISPLAY LIST MODE SWITCH | SDLMS | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A4 | EXTENDED LOAD DISPLAY LIST | ELDL | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A5 | EXTENDED READ DISPLAY LIST | ERDL | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A6 | EXTENDED EXECUTE INSTRUCTION MEMORY | EXIM | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A7 | EXTENDED LOAD DISPLAY LIST REVERSE PACKING | ELDLRP | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A8 | EXTENDED READ DISPLAY LIST REVERSE PACKING | ERDLRP | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A9 | EXTENDED CALL DISPLAY LIST | ECDL | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AA | EXTENDED LOAD DISPLAY LIST REGISTER | ELDLR | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AB | EXTENDED STORE DISPLAY LIST REGISTER | ESTDLR | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AC | EXTENDED JUMP CONDITIONAL UPON DISPLAY LIST REGISTER | EJDLR | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AD | SET LOCAL KEYBOARD FUNCTION EXTENDED | SLKFE | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AE | SET LOCAL CURSOR FUNCTION EXTENDED | SLCFE | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B0 | ARROW PRODUCT | AP | S - | | | | | | | | | | | | | | | | | | | | | | | X | | X | | X | X | |
| B1 | WIND BARB | WB | S - | | | | | | | | | | | | | | | | | | | | | | | X | | X | | X | X | |
| B2 | VARIABLE EXCEPTION VECTOR | VEV | S - | | | | | | | | | | | | | | | | | | | | | | | X | | X | | X | X | |
| B3 | KILL PRINTER PROCESS | KPP | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B4 | SENSE PRINTER PROCESS STATUS | SPPS | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B5 | SET PRINTER | SP | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B6 | PRINTACOLOR PRINT | PRNT | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BB | ALLOCATE STROKED FONT | ASF | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BC | DEALLOCATE STROKED FONT | DSF | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BD | DEFINE STROKED FONT | DEFSF | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BE | WRITE STROKED TEXT | WST | S - | | | | | | | | | | | | | | | | | | | | | | X | X | | | | X | X | |
| BF | INQUIRE TEXT EXTENT | ITE | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C0 | WRITE PROCESSOR BOARD PORT | WPBP | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C1 | WRITE PROCESSOR BOARD PORT REVERSE PACKING | WPBPRP | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C2 | WRITE SERIAL PORT CONFIGURATION | WSPC | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C3 | WRITE SLC CURSOR PORT | WSLCCP | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C4 | WRITE SLC CURSOR PORT REVERSE PACKING | WSLCCPRP | S - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

X0100-326#6-02D

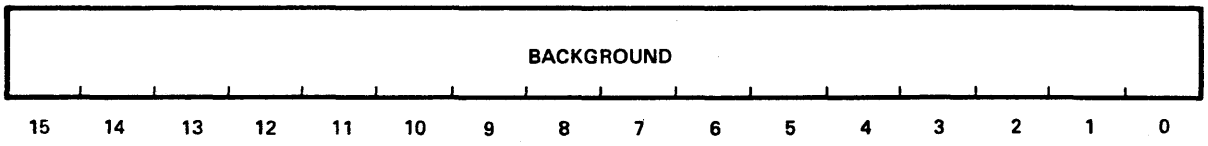
**WRITE MASK PARAMETER (Word 1, Bit 0)**




The WRITE MASK parameter may be set by any normal-format instruction except INOP. The operand is a 16-bit word used as a write-enable mask to generate any data in refresh memory groups. In any refresh memory group there can be up to 16 planes of memory; this parameter contains one bit per plane with bit 0 corresponding to plane 0, etc. When set to 1, the corresponding plane is write-enabled. This parameter has no affect on readback operations from refresh memory.

WRITE MASK default value = FFFF(H)

**BACKGROUND PARAMETER (Word 1, Bit 2)**



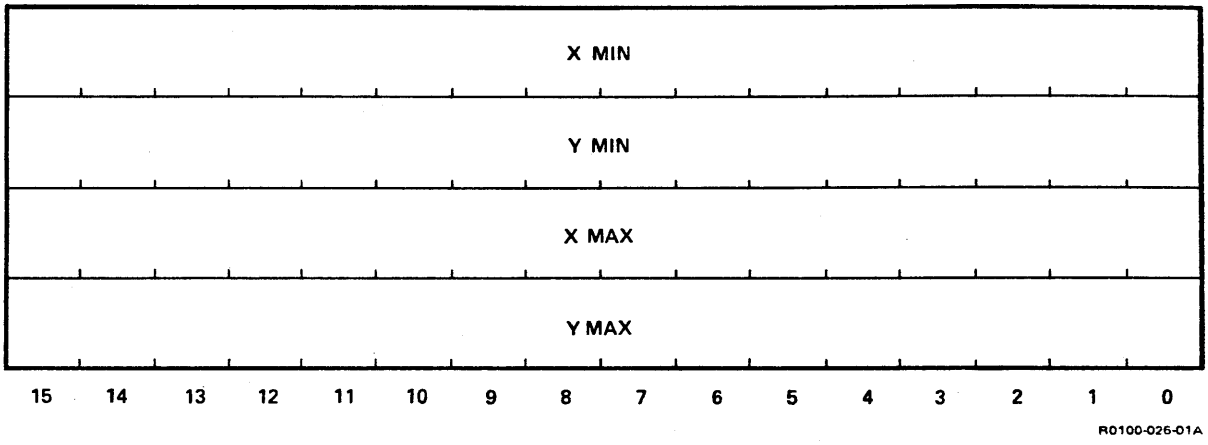

The BACKGROUND parameter may be set by any normal-format instruction except INOP. It is flagged by bit 2 of operand flag word 1. The operand is a single 16-bit word identical in format to the FOREGROUND argument. It establishes the color or intensity written for "zero" data bits during generation of font, raster, or graphics data. When writing reversed background data, BACKGROUND establishes color or intensity for "ones" data bits.

BACKGROUND default value = 0000(H)

| Parameters | Description |
|---|---|
| INDEX 1 ADDRESSING | The current value of the INDEX 1 parameter is added to the received X- and Y-coordinates to form the resultant absolute address; so INDEX 1 is set to the sum of the old INDEX 1 plus the new value, and all further coordinates have the new INDEX 1 value added to them before they are used. |
| INDEX 2 ADDRESSING | The current value of the INDEX 2 parameter is added to the received X- and Y-coordinates to form the resultant absolute address; so INDEX 1 is set to the sum of INDEX 2 plus the new value, and all further coordinates have the INDEX 2 value added to them before they are used. |
| RELATIVE ADDRESSING | The current value of the COP (as a result of the previous drawing operation) is added to the received X- and Y-coordinates to form the resultant absolute address; so INDEX 1 is set to the sum of the current COP plus the new value, and all further coordinates have the current COP value added to them before they are used. Note that as each coordinate is used in drawing, it can change the value of the current COP. |

| Parameters | Description |
|---|---|
| INDEX 1 ADDRESSING | The current value of the INDEX 1 parameter is added to the received X- and Y-coordinates to form the resultant absolute address; so INDEX 2 is set to the sum of INDEX 1 plus the new value, and all further coordinates have the INDEX 1 value added to them before they are used. |
| INDEX 2 ADDRESSING | The current value of the INDEX 2 parameter is added to the received X- and Y-coordinates to form the resultant absolute address; so INDEX 2 is set to the sum of the old INDEX 2 plus the new value, and all further coordinates have the new INDEX 2 value added to them before they are used. |
| RELATIVE ADDRESSING | The current value of the COP (as a result of the previous drawing operation) is added to the received X- and Y-coordinates to form the resultant absolute address; so INDEX 2 is set to the sum of the current COP plus the new value, and all further coordinates have the current COP value added to them before they are used. Note that as each coordinate is used in drawing, it can change the value of the current COP. |

**FORMAT WINDOW PARAMETER (WORD 1, BIT 6)**




The FORMAT WINDOW parameter may be set by any normal-format instruction except INOP. It is flagged by bit 6 of operand flag word 1. The parameter is four 16-bit words and specifies a rectangular region used by the ERS, WI, RI, WT, WR, SCRX, and SCRY instructions. The FORMAT WINDOW values are read as follows: XMIN, YMIN, XMAX, YMAX. They must obey the following conditions: XMIN $\leq$ XMAX and YMIN $\leq$ YMAX. Whenever the FORMAT WINDOW parameter is specified, the COP is set to the coordinates defined by table 2-4 as determined by the value of SCAN and the video orientation (set by the SELVO instruction) prior to the WINDOW setting instruction (figure 2-4).

Table 2-4. COP Placement after Format Window Setting

| SCAN | UPPER LEFT ORIGIN | | LOWER LEFT ORIGIN | |
|---|---|---|---|---|
| | XCOP | YCOP | XCOP | YCOP |
| 0 | XMIN | YMIN | XMAX | YMIN |
| 1 | XMAX | YMIN | XMAX | YMAX |
| 2 | XMIN | YMAX | XMIN | YMIN |
| 3 | XMAX | YMAX | XMIN | YMAX |
| 4 | XMIN | YMIN | XMAX | YMIN |
| 5 | XMIN | YMAX | XMIN | YMIN |
| 6 | XMAX | YMIN | XMAX | YMAX |
| 7 | XMAX | YMAX | XMIN | YMAX |

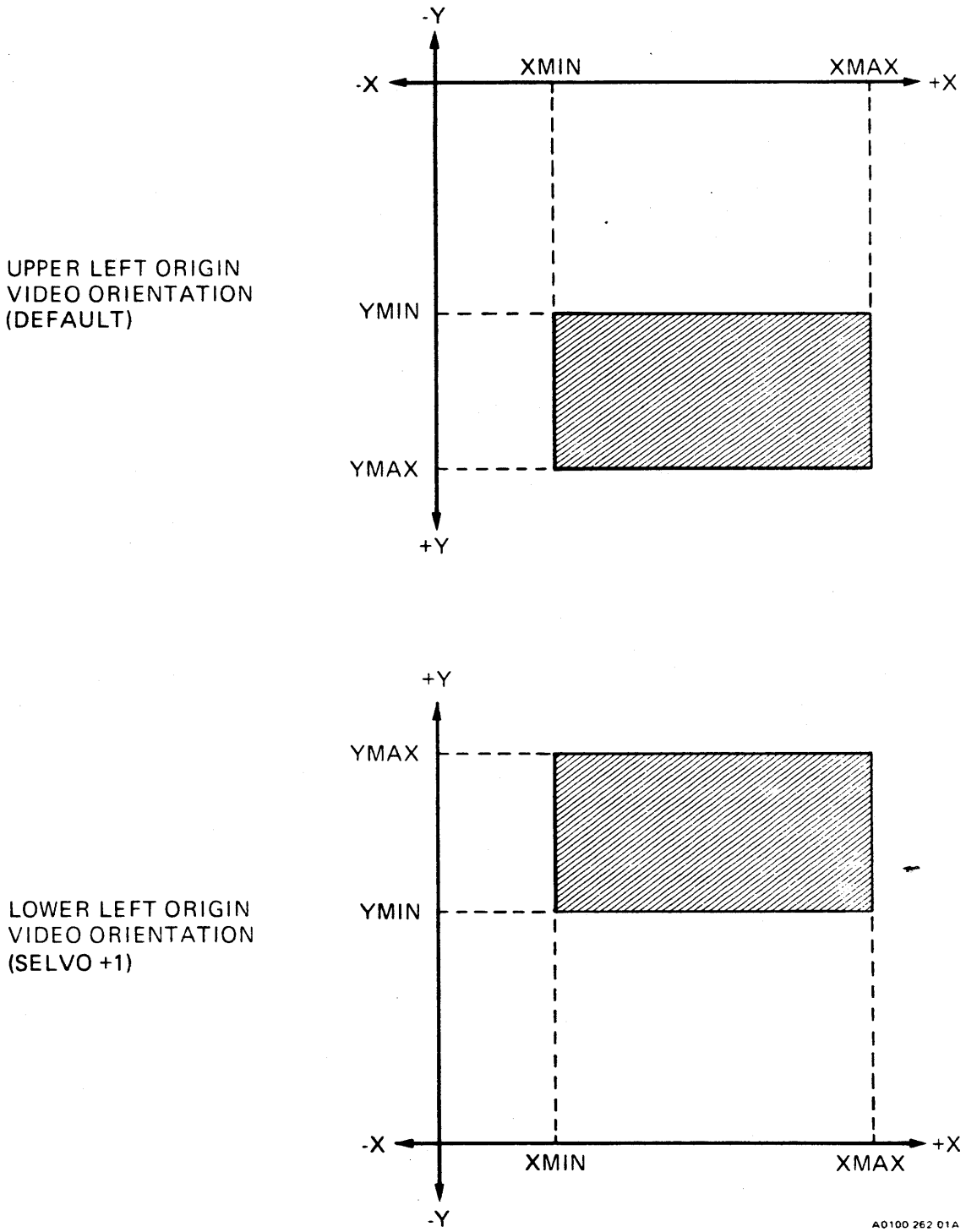



Figure 2-4. Format Window Definition

Table 2-5. Image and Raster Mode Scan Directions

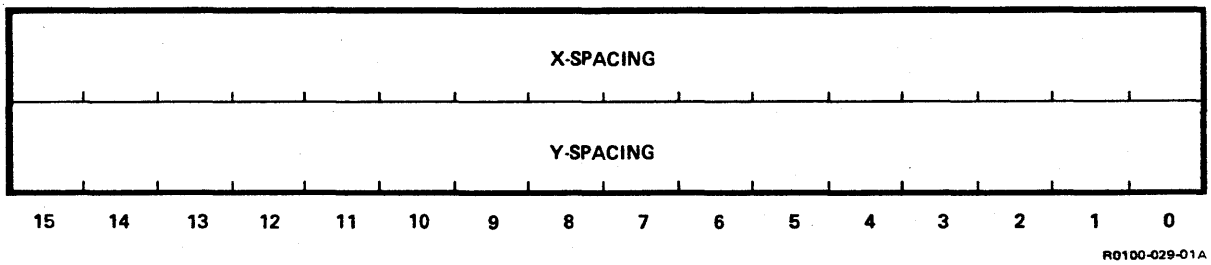
| SCAN | DIRECTION OF WRITING PROCESS | |
|---|---|---|
| | PRIMARY | SECONDARY |
| 0 | +X | +Y |
| 1 | -X | +Y |
| 2 | +X | -Y |
| 3 | -X | -Y |
| 4 | +Y | +X |
| 5 | -Y | +X |
| 6 | +Y | -X |
| 7 | -Y | -X |

**Table 2-6. Write Scan Update Directions and Character Orientation**

| | | | CHARACTER ORIENTATION | |
|---|---|---|---|---|
| SCAN MODE | PRIMARY SCAN UPDATE | SECONDARY SCAN UPDATE | UPPER LEFT ORIENTATION | LOWER LEFT ORIENTATION |
| 0 | + X—Spacing | + Y—Spacing | R | Я |
| 1 | - X—Spacing | + Y—Spacing | Я | R |
| 2 | + X—Spacing | - Y—Spacing | R | Я |
| 3 | - X—Spacing | - Y—Spacing | Я | R |
| 4 | + Y—Spacing | + X—Spacing | R | Я |
| 5 | - Y—Spacing | + X—Spacing | Я | R |
| 6 | + Y—Spacing | - X—Spacing | R | Я |
| 7 | - Y—Spacing | - X—Spacing | Я | R |

B0100-263-02A

**SPACING PARAMETER (WORD 1, BIT 9)**



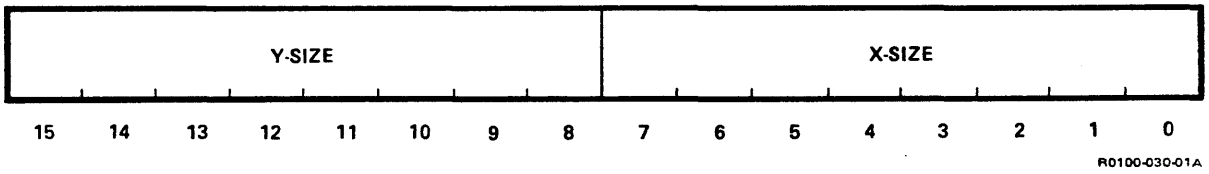

The SPACING parameter may be set by any normal-format instruction except INOP. It is flagged by bit 9 of operand flag word 1. The parameter is two 16-bit words, and negative spacing may be specified by using two's-complement notation.

For text generation, SPACING determines the distance between successive characters and the distance between successive character lines. X-SPACING is always in the horizontal axis and Y-SPACING is always in the vertical axis. The SCAN parameter defines which SPACING value is between characters and which is between lines (figure 2-5).

For WPB, WPV, and WPP, SPACING defines the increment from plot entity to plot entity along the plot axis. For horizontal plots, X-SPACING is used to define the plot axis increment, while for vertical plots, Y-SPACING is used to define the plot axis increment.

X-SPACING default value = 0007(H)
Y-SPACING default value = 0009(H)

## SIZE PARAMETER (WORD 1, BIT 10)




The SIZE parameter may be set by any normal-format instruction except INOP. It is flagged by bit 10 of operand flag word 1. The parameter is a single 16-bit word that specifies a sizing or repetition factor of generated picture elements to displayed picture elements for the WR, WT, and WRCT instructions. When sizing text, X-SIZE always refers to character width and Y-SIZE always refers to character height.

The received-to-displayed-picture ratios are defined in table 2-7.

X-SIZE default value = 0
Y-SIZE default value = 0

Table 2-7. Sized Picture Ratios

| SIZE | RATIO | PICTURE ELEMENTS | |
|---|---|---|---|
| | | RECEIVED | DISPLAYED |
| 0 | 1:1 | 1 | 1 |
| 1 | 1:2 | 1 | 2 |
| 2 | 1:3 | 1 | 3 |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| 14 | 1:15 | 1 | 15 |
| 15 | 1:16 | 1 | 16 |

**NOTE**

The internal spacing values used in WT and WRCT are automatically adjusted when SIZE values greater than 0 are specified so that character spacing remains proportional regardless of SIZE.

Table 2-8. Logical/Arithmetic Function Codes

| CODE(H) | FUNCTION |
|---|---|
| 0 | NONE (DATA REPLACEMENT) |
| 1 | LOGICAL OR |
| 2 | LOGICAL XOR |
| 3 | LOGICAL AND |
| 4 | ARITHMETIC SUM |
| 5 | ARITHMETIC DIFFERENCE |
| 6 | GREATEST VALUE |
| 7 | LEAST VALUE |
| 8 | AVERAGE COMPUTATION |
| 9 | INVERSE ARITHMETIC DIFFERENCE |
| A | SIGN MAGNITUDE SUM |
| B | SIGN MAGNITUDE DIFFERENCE |
| C | SIGN MAGNITUDE INVERSE ARITHMETIC DIFFERENCE |

Function 0 - Replacement

RP = NP .AND. WM

The RP is set equal to the NP (normal WI processing is performed). The OP is lost.

Function 1 - Logical Inclusive OR

RP = (NP .OR. OP) .AND. WM

The RP is the logical inclusive OR of the OP with the NP.

Function 2 - Logical Exclusive OR

RP = (NP .AND. WM) .XOR. (OP .AND. WM)

The RP is the logical exclusive OR of the OP with the NP.

Function 3 - Logical AND

RP = (NP .AND. OP) .AND. WM

The RP is the logical AND of OP and NP values.

sum of two positive numbers produces a carry into the sign bit, then the carry is lost; that is, the result remains positive. If the sum of two negative numbers produces a carry into the sign bit, then the result remains negative as outlined below:

WRITE MASK = 0000001111110000 = 03F0(H)

0020(H) + 0210(H) = 0010(H)2 + (-1) = 1
0220(H) + 0210(H) = 0230(H)-2 + (-1) = -3
0020(H) + 0240(H) = 0220(H)2 + (-4) = -2

WRITE MASK = 0000000100001111 = 010F(H)

0002(H) + 0101(H) = 0001(H)2 + (-1) = 1
0102(H) + 0101(H) = 0103(H)-2 + (-1) = -3
0002(H) + 0104(H) = 0102(H)2 + (-4) = -2

Function A - Arithmetic Sum (Sign Magnitude)

RP =(((OP .AND. WM) + (NP .AND. WM)) .AND. WM)

The RP is the sum of the OP and the NP. Both NP and OP are evaluated as sign-magnitude numbers using the WRITE MASK parameter to define sign bit.

Function B - Arithmetic Difference (Sign-Magnitude)

RP = (((OP .AND. WM) - (NP .AND. WM)) .AND. WM)

The RP is the difference of the OP minus the NP. Both NP and OP are evaluated as sign-magnitude numbers using the WRITE MASK parameter to define the sign bit.

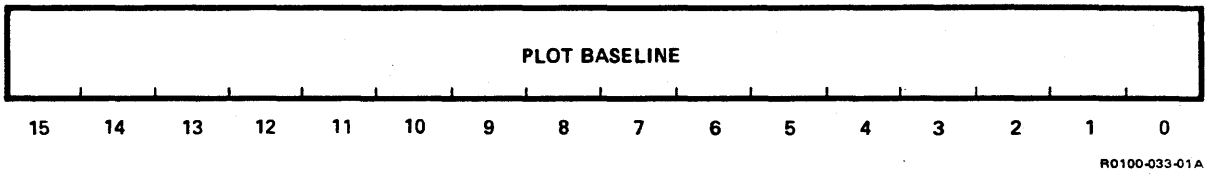
Function C - Inverse Arithmetic Difference (Sign-Magnitude)

RP = (((NP .AND. WM) - (OP .AND. WM)) .AND. WM)

The RP is the difference of the NP minus the OP. Both NP and OP are evaluated as sign-magnitude numbers using the WRITE MASK parameter to define the sign bit.

LOGICAL/ARITHMETIC FUNCTION default value = 0000(H) replacement mode

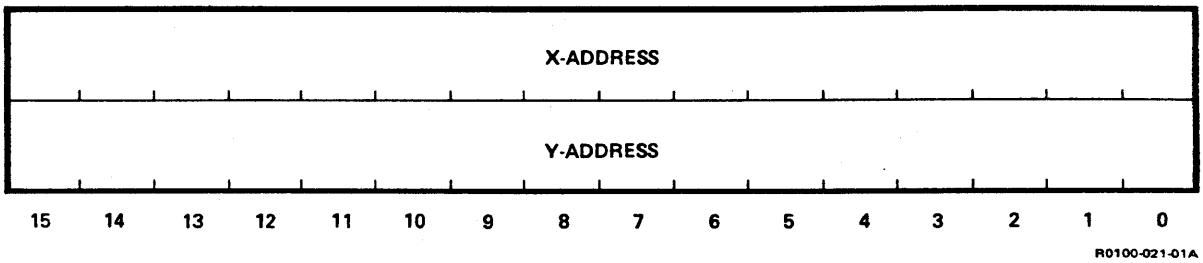
**BASELINE PARAMETER (WORD 1, BIT 13)**




The BASELINE parameter may be set through any normal-format instruction except INOP. It is flagged by bit 13 of operand flag word 1. The operand is a 16-bit word and specifies whether a baseline plot or a linked plot is to be drawn. When PLOT BASELINE is zero, a linked plot (a plot in which each end-point along the curve becomes the start-point for the succeeding plot segment) is drawn. When PLOT BASELINE is nonzero, it defines the startpoint for each plot segment. If SCAN is between 0 and 3, the BASELINE parameter defines the horizontal axis to which the filled-lot segments will be drawn; that is, PLOT BASELINE defines a Y-address. Similarly, if SCAN is between 4 and 7, then the BASELINE parameter defines the vertical axis to which the filled-plot segments will be drawn; that is, PLOT BASELINE defines an X-address.

PLOT BASELINE default value = 0000(H)

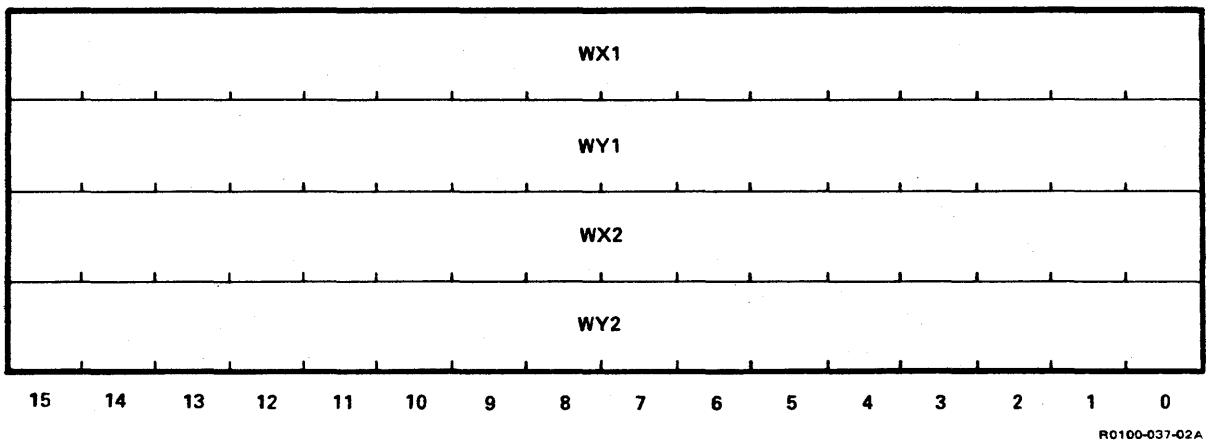
**START-POINT PARAMETER (WORD 1, BIT 15)**




The START-POINT parameter may be set by any normal-format instruction except INOP. It is flagged by bit 15 of operand flag word 1. The parameter is two 16-bit words that specify a start-point for WI, RI, WR, WT, WVL, WC, WC32, WPV, WPP, and WPB instructions. Note that when WINDOW or SCAN is specified, the appropriate startpoint is calculated automatically for the WI, RI, WR, and WT instructions and need not be specified unless the writing process is to begin in other than the appropriate corner of the WINDOW. Table 2-4 defines the default values for the START-POINT parameter, based upon the setting of WINDOW and SCAN. Because WINDOW is not pertinent to the WVL, WC, WC32, WPV, WPP, and WPB instructions, the START-POINT parameter must be specified. Otherwise, the last endpoint (current COP) is used as the new start-point. The START-POINT parameter explicitly sets the current operating point (COP) according to the addressing mode specified by the instruction containing the parameter. For window-oriented commands, the value specified must be within the area specified by the WINDOW parameter for proper operation.

X-ADDRESS default value = 0000(H)
Y-ADDRESS default value = 0000(H)

## WRITE-ENABLE WINDOW PARAMETER (WORD 2, BIT 1)

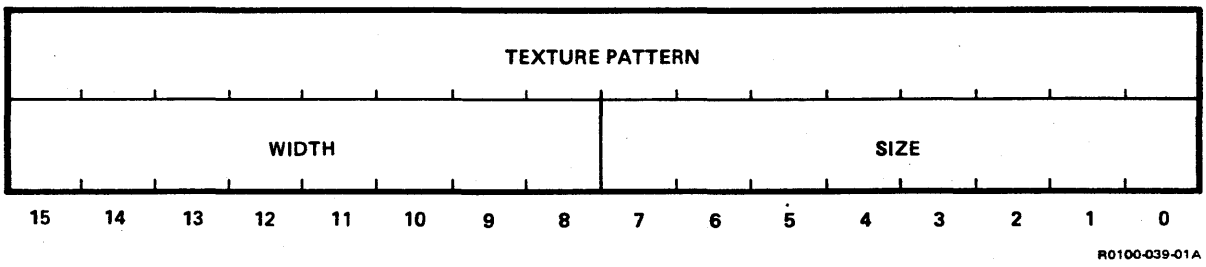



The WRITE-ENABLE WINDOW parameter may be set by any normal-format instruction except INOP. It is flagged by bit 1 of operand flag word 2. The parameter is four 16-bit words. At default, these values define the region of global address space where the writing of data to refresh memory is enabled. The following conditions must be observed:

All values must be in the range of -16384 to +16383, and WX 1 $\leq$ WX 2 and WY 1 $\leq$ WY 2.

WX 1 default value = 0000(H)
WY 1 default value = 0000(H)
WX 2 default value = XRES-1
WY 2 default value = YRES-1

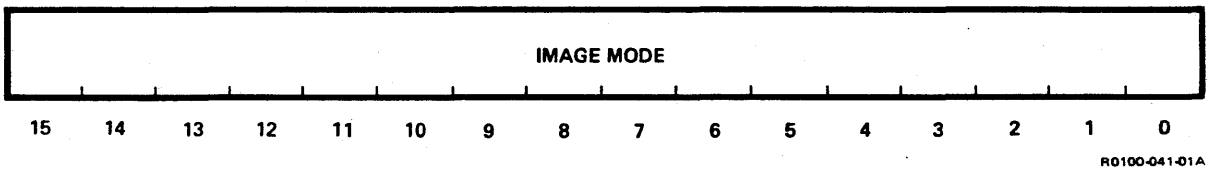
## VECTOR TEXTURE PARAMETER (WORD 2, BIT 3)




The VECTOR TEXTURE parameter may be set by any normal-format instruction except INOP. It is flagged by bit 3 of operand flag word 2. The parameter is two 16-bit words. The first word defines the vector pattern as a bit mask where the most significant bit (bit 15) is the first-used bit in vector generation. The second word defines the number of bits in the pattern mask to be used and how many times to repeat each pattern bit as a vector is being drawn. The range of values for WIDTH is from 0 to 16. A value of zero implies that a pattern with a value of FFFF is to be used (no vector texturing). The range of values of SIZE is from 0 to 15. A value of 0 implies a pattern bit is written once as a vector is drawn. A value from 1 to 15 implies the number of extra times each pattern bit is repeated along the vector; that is, 15 means each pattern bit is repeated (stored into refresh memory) 16 times.

TEXTURE PATTERN default value = FFFF(H)
WIDTH/SIZE default value = 0000(H)

## IMAGE MODE PARAMETER (WORD 2, BIT 5)




The IMAGE MODE parameter may be set by any normal-format instruction except INOP. It is flagged by bit 5 of operand flag word 2. The operand is a single 16-bit word and specifies one of three possible image loading/reading modes as outlined below.

0 Word Mode -- Image data received from/returned to the host is in word format and stored/read as one word per pixel. The significance of the bytes within each word is defined by the state of the reverse packing flag (RP) set within the WI or RI instruction. If RP = 1, the bytes are reversed before storing/reading the word to/from refresh memory (figure 2-6).

1 Low-Byte Mode -- Image data received from/returned to the host is in byte format packed two bytes per word, and stored/read one byte per pixel. In low byte mode all data is considered to reside in the low-order eight bits of each pixel. When a word of image data is received from the host, two pixels in refresh memory are modified. Here each pixel's low-order eight bits are updated and the high-order eight bits remain unchanged. When the host requests a word of image data, two pixels in refresh memory are read, the low-order bytes are packed into a word, and the word is returned to the host. The order in which the bytes are packed (for storing or reading) is defined by the state of the reverse packing flag (RP) set within the WI or RI instruction (figures 2-7, 2-8, and 2-9).

2 High-Byte Mode -- Image data received from/returned to the host is in byte format, packed two bytes per word, and stored/read one byte per pixel. In the high-byte mode, all data is considered to reside in the high-order eight bits of each pixel. When a word of image data is received from the host, two pixels in refresh memory are modified. Here each pixel's high-order eight bits are updated and the low-order eight bits remain unchanged. When the host requests a word of image data, two pixels in refresh memory are read, the high-order bytes are packed into a word, and the word is returned to the host. The order in which the bytes are packed (for storing or reading) is defined by the state of the reverse packing flag (RP) set within the WI or RI instruction (figures 2-10 and 2-11).

IMAGE MODE default value = 0000(H)

READ IMAGE - LOW-BYTE MODE

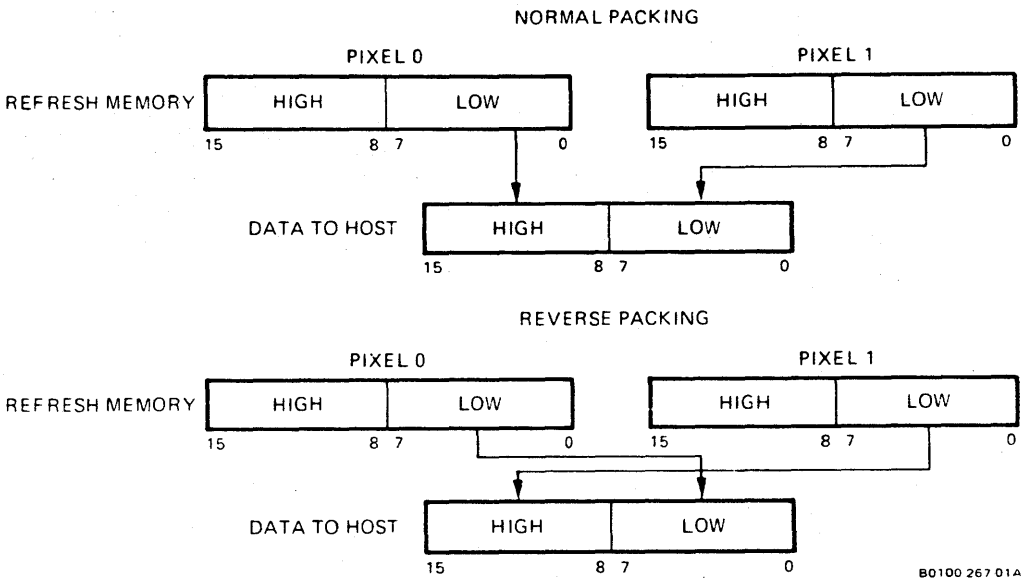



Figure 2-8. Low-Byte Mode Effects on Read Image Data

IMAGE MODE VALUES

| IMAGE MODE | PARAMETER VALUE |
|---|---|
| WORD | 0000 |
| LOW BYTE | 0001 |
| HIGH BYTE | 0002 |

B0100-233-01A

Figure 2-9. Image Mode Values

## 2.3 INSTRUCTIONS

This paragraph describes the RM-9460 instruction set and presents formats, parameters, and errors for each instruction. Table 2-9 lists instructions by mnemonic cross referenced to opcode and manual page number. Table 2-10 lists instructions by opcode cross referenced to mnemonic and manual page number.

Table 2-9. Instruction Mnemonics

| MNEMONIC | INSTRUCTION | OPCODE | PAGE |
|---|---|---|---|
| ACS | Allocate Control Store | 91 | 2-204 |
| ADLR | Add Display List Registers | 47 | 2-230 |
| ALCON | Allocate Context | 5C | 2-137 |
| ALDL | Allocate Display List | 3B | 2-213 |
| ALPF | Allocate Programmable Font | 4C | 2-245 |
| ALTD | Allocate Trend | 78 | 2-304 |
| ANDDLR | AND Display List Register | 87 | 2-232 |
| AP | Arrow Product | B0 | 2-302f |
| ARC1 | Write Arc Type 1 | 6E | 2-294 |
| ARC2 | Write Arc Type 2 | 6F | 2-297 |
| ARC3 | Write Arc Type 3 | 70 | 2-300 |
| ASF | Allocate Stroked Font | BB | 2-351 |
| ATTPF | Attach Programmable Font | 4E | 2-247 |
| BERS | Bulk Erase | 2B | 2-119 |
| CCM | Call Control Memory | 1D | 2-218 |
| CCS | Call Control Store | 93 | 2-207 |
| CDL | Call Display List | 41 | 2-223 |
| CIRC | Write Circle | 6D | 2-291 |
| CKES | Clear Keyboard Echo State | 77 | 2-285 |
| CLCF | Clear Local Cursor Function | 75 | 2-282 |
| CLKF | Clear Local Keyboard Function | 73 | 2-279 |
| COMBI | Combine Image | 6B | 2-335 |
| COMBT | Combine Image Triggered | 6C | 2-345 |
| COPY | Copy Image | 69 | 2-331 |
| COPYIM | Copy Image and Magnify | 96 | 2-210e |
| COPYIR | Copy Image and Rotate | 95 | 2-210a |
| COPYT | Copy Image Triggered | 6A | 2-341 |
| CWC | Center Window at Cursor | 32 | 2-173 |
| DCVC | Define Cursor/Video Configuration | 34 | 2-176 |
| DD | Disable Detect | 63 | 2-189 |
| DDCC | Define Device/Cursor Configuration | 33 | 2-175 |
| DDLR | Decrement Display List Register | 44 | 2-226 |
| DECON | Deallocate Context | 5D | 2-138 |
| DEDL | Deallocate Display List | 3C | 2-214 |
| DEFSF | Define Stroked Font | BD | 2-354 |
| DEPF | Deallocate Programmable Font | 4D | 2-246 |
| DETD | Deallocate Trend | 79 | 2-316 |
| DISTD | Display Trend | 7E | 2-313 |
| DIVDLR | Divide Display List Registers | 85 | 2-229 |
| DOP | Delete Opcode | 9D | 2-210q |

Table 2-9. Instruction Mnemonics (Continued)

| MNEMONIC | INSTRUCTION | OPCODE | PAGE |
|---|---|---|---|
| PUSHE | Save Environment | 13 | 2-105 |
| PUSHM | Push Matrix | 50 | 2-258 |
| RAM | Read Auxiliary Memory | 04 | 2-56 |
| RCFG | Read Configuration | 90 | 2-202 |
| RCON | Read Context Associations | 60 | 2-143 |
| RCSG | Read Cursor Status Global | 2F | 2-166 |
| RCSL | Read Cursor Status Local | 83 | 2-168 |
| RCSP | Read Cursor Status Pixel | 2E | 2-164 |
| RCSS | Read Cursor Status Screen | 17 | 2-162 |
| RD | Resume Detect | 65 | 2-191 |
| RDB | Read Back Detect Buffer | 68 | 2-194 |
| RDL | Read Display List | 1C | 2-217 |
| RDLRP | Read Display List Reverse Packing | 20 | 2-222 |
| READ | Read Soft Register | 02 | 2-53 |
| READAS | Read Allocation State | 5F | 2-141 |
| READM | Read Matrix | 5B | 2-271 |
| READP | Read Normal Parameters | 24 | 2-111 |
| REAERR | Read Errors | A1 | 2-195b |
| RERR | Read Error Status | 25 | 2-112 |
| RESCON | Reset Context | 3D | 2-140 |
| RETDL | Return from Display List | 42 | 2-224 |
| RI | Read Image | 0B | 2-70 |
| RKB | Read Keyboard | 19 | 2-148 |
| ROTAT | Rotate Matrix | 5A | 2-273 |
| RSET | Reset | 05 | 2-58 |
| RSM | Reset MCPs | 94 | 2-209 |
| RTC | Read Tablet Coordinates | 40 | 2-177 |
| RTCS | Read Tablet Coordinates and Status | 8C | 2-183 |
| SCALE | Scale Matrix | 58 | 2-268 |
| SCON | Select Context | 5E | 2-139 |
| SCRX | Scroll X | 11 | 2-99 |
| SCRY | Scroll Y | 12 | 2-102 |
| SCW | Set Cursor Window | 81 | 2-179 |
| SD | Suspend Detect | 64 | 2-190 |
| SDD | Set Detect Data | 66 | 2-192 |
| SDLMS | Set Display List Mode Switch | A3 | 2-243e |
| SDLR | Set Display List Register | 43 | 2-225 |
| SDP | Set Detect Parameters | 61 | 2-186 |
| SDS | Sense Detect Status | 67 | 2-193 |
| SELMG | Select MCP/Group | 22 | 2-107 |
| SELVO | Select Video Orientation | 27 | 2-114 |
| SERRC | Sense Error Count | A0 | 2-195a |
| SET | Set Parameter | 08 | 2-61 |
| SETDC | Set Display Class Ranges | 23 | 2-109 |
| SETER | Set Errors | A2 | 2-195c |
| SETM | Set Matrix | 52 | 2-260 |
| SKES | Set Keyboard Echo State | 76 | 2-283 |
| SLCF | Set Local Cursor Function | 74 | 2-280 |

Table 2-9. Instruction Mnemonics (Continued)

| MNEMONIC | INSTRUCTION | OPCODE | PAGE |
|---|---|---|---|
| WST | Write Stroked Text | BE | 2-359 |
| WT | Write Text | 0C | 2-75 |
| WVL | Write Vector Linked | 0E | 2-87 |
| WVU | Write Vector Unlinked | 35 | 2-121 |
| XIM | Execute Instruction Memory | 1E | 2-219 |
| XORDLR | XOR Display List Registers | 89 | 2-234 |
| ZOOM | Zoom | 26 | 2-113 |

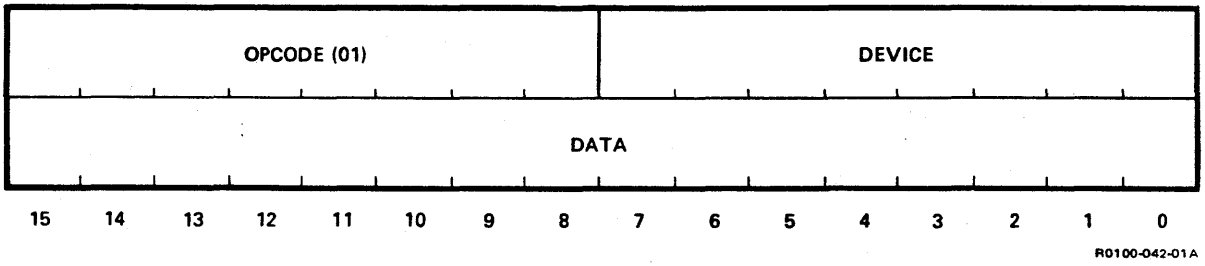
Table 2-10. Instruction Opcodes

| OPCODE | MNEMONIC | INSTRUCTION | PAGE |
|---|---|---|---|
| 01 | LOAD | Load Hard Register | 2-52 |
| 02 | READ | Read Soft Register | 2-53 |
| 03 | LAM | Load Auxiliary Memory | 2-54 |
| 04 | RAM | Read Auxiliary Memory | 2-56 |
| 05 | RSET | Reset | 2-58 |
| 06 | INIT | Initialize | 2-59 |
| 07 | INOP | No Operation | 2-60 |
| 08 | SET | Set Parameter | 2-61 |
| 09 | ERS | Erase | 2-62 |
| 0A | WI | Write Image | 2-64 |
| 0B | RI | Read Image | 2-70 |
| 0C | WT | Write Text | 2-75 |
| 0D | WR | Write Raster | 2-84 |
| 0E | WVL | Write Vector Linked | 2-87 |
| 0F | WC | Write Conic | 2-90 |
| 10 | WPB | Write Plot Box | 2-96 |
| 11 | SCRX | Scroll X | 2-99 |
| 12 | SCRY | Scroll Y | 2-102 |
| 13 | PUSHE | Save Environment | 2-105 |
| 14 | POPE | Restore Environment | 2-106 |
| 15 | LPF | Load Programmable Font | 2-248 |
| 16 | WCSS | Write Cursor State Screen | 2-154 |
| 17 | RCSS | Read Cursor Status Screen | 2-162 |
| 18 | WKB | Write Keyboard | 2-146 |
| 19 | RKB | Read Keyboard | 2-148 |
| 1A | SPS | Sense Peripheral Status | 2-144 |
| 1B | LDL | Load Display List | 2-215 |
| 1C | RDL | Read Display List | 2-217 |
| 1D | CCM | Call Control Memory | 2-218 |
| 1E | XIM | Execute Instruction Memory | 2-219 |
| 1F | LDLRP | Load Display List Reverse Packing | 2-220 |
| 20 | RDLRP | Read Display List Reverse Packing | 2-222 |
| 21 | LPFRP | Load Programmable Font Reverse Packing | 2-251 |
| 22 | SELMG | Select MCP/Group | 2-107 |
| 23 | SETDC | Set Display Class Ranges | 2-109 |
| 24 | READP | Read Normal Parameters | 2-111 |
| 25 | RERR | Read Error Status | 2-112 |
| 26 | ZOOM | Zoom | 2-113 |
| 27 | SELVO | Select Video Orientation | 2-114 |
| 28 | LOADSO | Load Subchannel Origins | 2-115 |
| 29 | WAITVR | Wait for Vertical Retrace | 2-117 |
| 2A | WAITL | Wait for Video Line | 2-118 |
| 2B | BERS | Bulk Erase | 2-119 |
| 2C | WCSP | Write Cursor State Pixel | 2-156 |
| 2D | WCSG | Write Cursor State Global | 2-158 |
| 2E | RCSP | Read Cursor Status Pixel | 2-164 |
| 2F | RCSG | Read Cursor Status Global | 2-166 |
| 30 | LCF | Load Cursor Font | 2-170 |

Table 2-10. Instruction Opcodes (Continued)

| OPCODE | MNEMONIC | INSTRUCTION | PAGE |
|---|---|---|---|
| 61 | SDP | Set Detect Parameters | 2-186 |
| 62 | ED | Enable Detect | 2-188 |
| 63 | DD | Disable Detect | 2-189 |
| 64 | SD | Suspend Detect | 2-190 |
| 65 | RD | Resume Detect | 2-191 |
| 66 | SDD | Set Detect Data | 2-192 |
| 67 | SDS | Sense Detect Status | 2-193 |
| 68 | RDB | Read Back Detect Buffer | 2-194 |
| 69 | COPY | Copy Image | 2-331 |
| 6A | COPYT | Copy Image Triggered | 2-341 |
| 6B | COMBI | Combine Image | 2-335 |
| 6C | COMBT | Combine Image Triggered | 2-345 |
| 6D | CIRC | Write Circle | 2-291 |
| 6E | ARC1 | Write Arc Type 1 | 2-294 |
| 6F | ARC2 | Write Arc Type 2 | 2-297 |
| 70 | ARC3 | Write Arc Type 3 | 2-300 |
| 71 | SLFS | Set Local Function State | 2-275 |
| 72 | SLKF | Set Local Keyboard Function | 2-277 |
| 73 | CLKF | Clear Local Keyboard Function | 2-279 |
| 74 | SLCF | Set Local Cursor Function | 2-280 |
| 75 | CLCF | Clear Local Cursor Function | 2-282 |
| 76 | SKES | Set Keyboard Echo State | 2-283 |
| 77 | CKES | Clear Keyboard Echo State | 2-285 |
| 78 | ALTD | Allocate Trend | 2-304 |
| 79 | DETD | Deallocate Trend | 2-316 |
| 7A | ITD | Init Trend | 2-306 |
| 7B | LTDP | Load Trend Patterns | 2-307 |
| 7C | UDTD | Update Trend | 2-310 |
| 7D | ERSTD | Erase Trend | 2-312 |
| 7E | DISTD | Display Trend | 2-313 |
| 7F | WPI | Write Packed Image | 2-318 |
| 80 | WIV | Write Image Vectors | 2-324 |
| 81 | SCW | Set Cursor Window | 2-179 |
| 82 | WRCT | Write Random Colored Text | 2-79 |
| 83 | RCSL | Read Cursor Status Local | 2-168 |
| 84 | WCSL | Write Cursor State Local | 2-160 |
| 85 | DIVDLR | Divide Display List Registers | 2-229 |
| 86 | MULDLR | Multiply Display List Registers | 2-228 |
| 87 | ANDDLR | AND Display List Register | 2-232 |
| 88 | ORDLR | OR Display List Registers | 2-233 |
| 89 | XORDLR | XOR Display List Registers | 2-234 |
| 8A | WC32 | Write Conic 32 Bits | 2-93 |
| 8B | STM | Set Tablet Mode | 2-181 |
| 8C | RTCS | Read Tablet Coordinates and Status | 2-183 |
| 8D | WCVU | Write Colored Vector Unlinked | 2-124 |
| 8E | V12LD | Load Video 12 PCB | 2-196 |
| 8F | MCPWT | MCP Write Through | 2-200 |
| 90 | RCFG | Read Configuration | 2-202 |

## LOAD HARD REGISTER (LOAD)

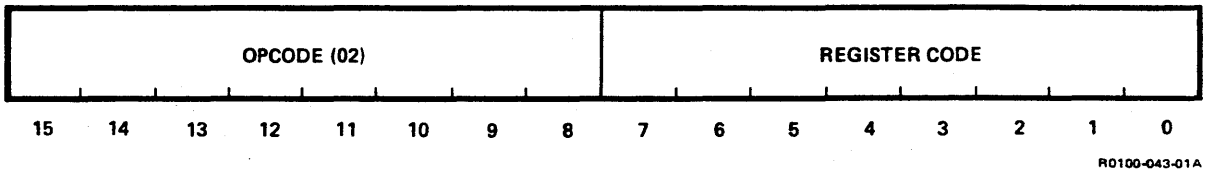



LOAD is a special-format instruction which provides the sequential integrity of RM-9000 instruction streams. It is treated as a two-word INOP. In the RM-9000 the internal display registers could be written directly. The internal architecture of the RM-9460 differs in that this capability is not possible; thus, compatible functions with the RM-9000 are not provided. The DEVICE address and the DATA for this instruction are discarded.

### Errors

None

## READ SOFT REGISTER (READ)




READ is a special-format instruction which returns the value of one of seven internal registers used by normal-format instructions for the current context (see table 2-11). It is provided in the RM-9460 mainly to provide compatibility with the RM-9000.

Following the output of the READ instruction, the host must read back exactly one word of data from the RM-9460. This instruction may not be executed from a display list.

| Parameters | Description |
|---|---|
| REGISTER CODE | Defines the register to be read back. |

| Errors | Description |
|---|---|
| Code = 0201 | Attempt to execute READ from within a display list. |

Table 2-11. Read Soft Register Addresses

| REGISTER CODE | REGISTER |
|---|---|
| 00H | WRITE MASK |
| 02H | X (ELEMENT) GLOBAL COP |
| 04H | Y (LINE) GLOBAL COP |
| 06H | X (ELEMENT) ORIGIN |
| 08H | Y (LINE) ORIGIN |
| 12H | FOREGROUND |
| 14H | BACKGROUND |

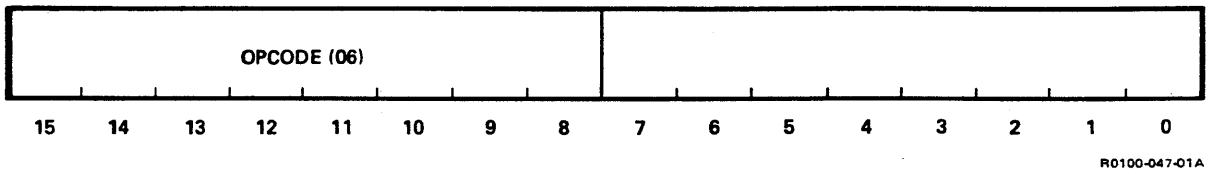
| Errors | Description |
|---|---|
| Code = 0307 | Illegal byte count. |
| Code = 0321 | No lookup table corresponding to device number. All data discarded. |

Data Format

The format of the data readback is dependent upon the nature of the auxiliary device assigned to DEVICE.

| Errors | Description |
|---|---|
| Code = 0401 | Attempt to execute RAM from within a display list. |
| Code = 0407 | Illegal byte count. |
| Code = 0421 | No lookup table corresponding to device number. |

## INITIALIZE (INIT)

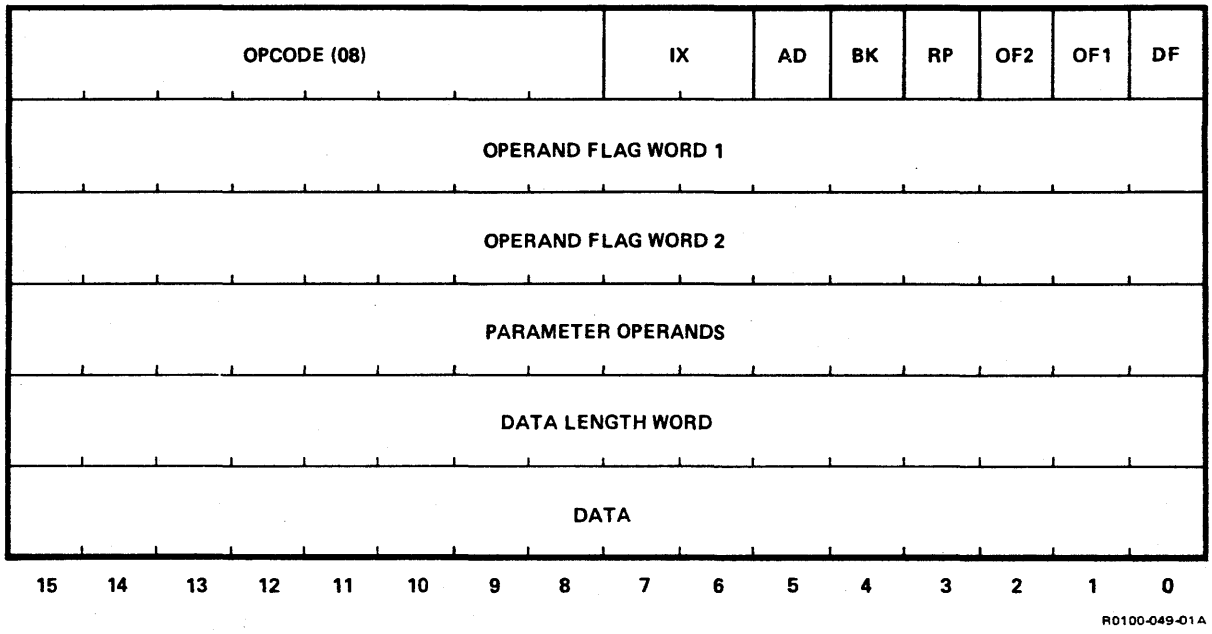



INIT is a special-format instruction which resets all normal-format parameters for the current context to the default values. It is equivalent to issuing a SET instruction with all parameter values set to the defaults. The environment stack is not affected. Previous PUSHE operations are available through POPE instruction.

### Errors

None

## SET PARAMETER (SET)




SET is a normal-format instruction which allows parameter operands to be defined as in any normal-format instruction. Data present in the instruction stream is ignored. This instruction is included to facilitate debugging of a display instruction stream. It allows a user to perform parameter operand processing where internal modifications could carry over to the subsequent instructions in the instruction stream while ignoring the received data.

| Flag | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, FORMAT WINDOW, BASELINE, and START POINT are evaluated. |

### Parameters

All normal-format parameters apply to the SET instructions in that the parameters may be redefined.

### Data Format

All data is discarded.

### Errors

None

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled to receive erase data. |
| FOREGROUND | Defines the color or intensity value to be stored in refresh memory when BK = 1. |
| BACKGROUND | Defines the color or intensity to be stored in refresh memory when BK = 0. |
| INDEX 1 | Displaces the values to be used for INDEX 2 and FORMAT WINDOW when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for FORMAT WINDOW when IX = 10(B). |
| FORMAT WINDOW | Defines the rectangular area to be erased. |
| WRITE-ENABLE WINDOW | Defines the area in global space where data may be written. |
| W.E.W OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, writing into refresh memory is enabled; otherwise, writing into refresh memory is disabled. |

## Data Format

All data is discarded.

## Errors

None

within the window, 16 instead of 15 pixels would be written on each line; and 14 full lines and 1 pixel on the last line are written (16 X 14) + 1 = 225. Therefore, in byte mode the FORMAT WINDOW parameter should always define a window that has an even number of pixels in the primary update direction.

| Flags | Description |
|---|---|
| IX | Defines the address mode in which INDEX 1, INDEX 2, FORMAT WINDOW, and START-POINT are evaluated. |
| RP | Defines the byte-accessing order in byte mode and the byte significance in word mode as follows:<br>Byte mode: RP = 0 - write high byte, then low byte<br>RP = 1 - write low byte, then high byte<br>Word mode: RP = 0 - store image word directly into refresh memory<br>RP = 1 - reverse bytes before storing image data into refresh memory |

**NOTE**

Since data to be written into refresh memory is supplied directly directly from the host processor and not from the FOREGROUND or BACKGROUND parameters, the AD and BK flags are ineffective when writing image data.

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled to receive image data. |
| INDEX 1 | Displaces the values to be used for INDEX 2, FORMAT WINDOW, and START-POINT when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for FORMAT WINDOW and START-POINT when IX = 10(B). |
| FORMAT WINDOW | Defines the rectangular area where image data is written. If START-POINT is not explicitly set in the WI instruction, FORMAT WINDOW and SCAN define the starting pixel coordinate. |

that is, either memory planes 0 through 7 are written in low-byte mode, or memory planes 8 through 15 are written in high-byte mode. Only those memory planes that have been write-enabled by the WRITE MASK are actually written. All other memory planes retain the original values.

COP Movement

The resulting current operating point (COP) after the completion of a WI instruction is the coordinate of the next pixel that would have been, written if (N + 1) image-data values had been passed in the instruction rather than N values. Since the primary and secondary update directions are defined by SCAN, the resulting COP position is a function of SCAN and the number of image data values written.

| Errors | Description |
|---|---|
| Code = 0A07 | Odd byte count in word mode. |

Example:

This example demonstrates the use of the WI instruction, word mode, and the operation of SCAN and FORMAT WINDOW with this instruction. The following instruction stream stores data in a rectangular window in refresh memory; the resulting displays are shown in figure 2-12, (a) through (h). The window is defined by the coordinate corners (100, 100) and (103, 103). Word mode is selected and 16 words of data (designated by the symbols D1 through D16) are written into the rectangular area defined by the FORMAT WINDOW parameter operand.

```
HEX         MNEMONIC            DESCRIPTION

0500        RSET                ;Clear Screen
0A07        WI+0F2+0F1+DF       ;Write Image Instruction
00C0        SCN+WIN             ;Operand Flag 1
0020        IMG                 ;Operand Flag 2
0064        100                 ;Element Left Margin = 100
0064        100                 ;Line Top Margin = 100
0067        103                 ;Element Right Margin = 103
0067        103                 ;Line Bottom Margin = 103
XXXX        XXXX                ;Scan Value from 0 through 7
0000        0                   ;Image Mode = Word Mode
0020        32                  ;Data Length Word = 32
            D1                  ;
            D2                  ;32 Bytes (16 words) of Data
            .
            .
            .
            D16
```

Where XXXX defines the scan mode 0 through 7

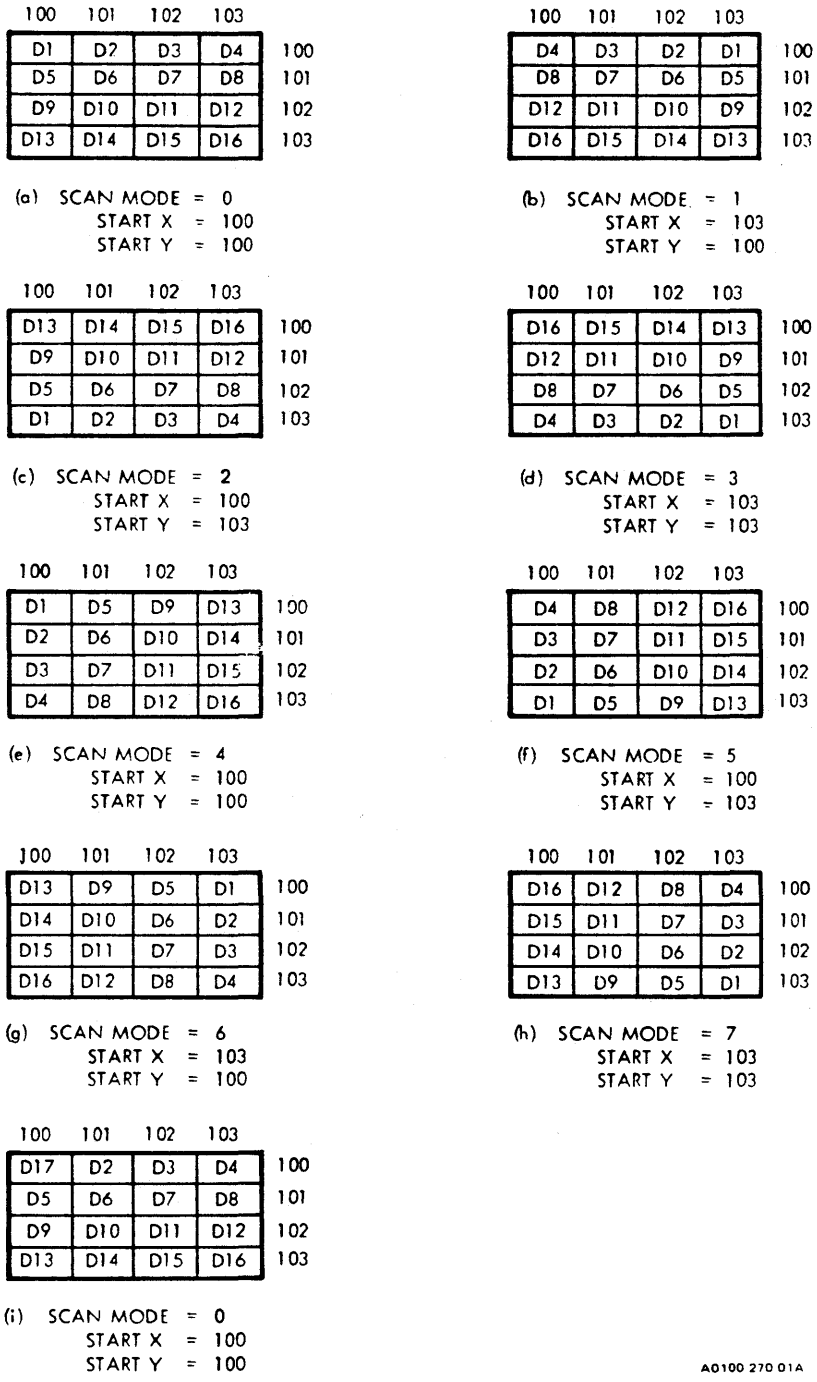



Figure 2-12. Imaging Scan Example

14 full lines and 1 pixel on the last line will be read (16 X 14) + 1 = 225. Therefore, in byte mode the FORMAT WINDOW parameter should always define a window that has an even number of pixels in the primary update direction.

| Flags | Description |
|---|---|
| IX | Defines the address mode in which INDEX 1, INDEX 2, FORMAT WINDOW, and START-POINT are evaluated. |
| RP | Defines the byte packing order in byte mode and the byte significance in word mode.<br><br>Byte Mode: RP = 0 - first read to high byte, second to low byte<br><br>RP = 1 - first read to low byte, second to high byte<br><br>Word Mode: RP = 0 - return image word to host directly<br><br>RP = 1 - reverse bytes of image word before returning to host |

| Parameters | Description |
|---|---|
| INDEX 1 | Displaces the values to be used for INDEX 2, FORMAT WINDOW, and START-POINT parameters when IX= 01(B). |
| INDEX 2 | Displaces the values to be used for FORMAT WINDOW and START-POINT parameters when IX = 10(B). |
| FORMAT WINDOW | Defines the rectangular area from which image data is read. If the START-POINT parameter is not explicitly set in the RI instruction, FORMAT WINDOW and SCAN define the starting pixel coordinate. |
| SCAN | Defines the primary and secondary update directions as well as the starting pixel coordinate when START-POINT is not defined in the RI instruction. |
| START-POINT | Defines the coordinate of the first pixel to be read from refresh memory. If neither START-POINT nor SCAN is defined, the COP resulting from the previous instruction is used as the starting pixel coordinate. If START-POINT defines a pixel outside refresh memory, then the data read with an RI instruction is indeterminate. |

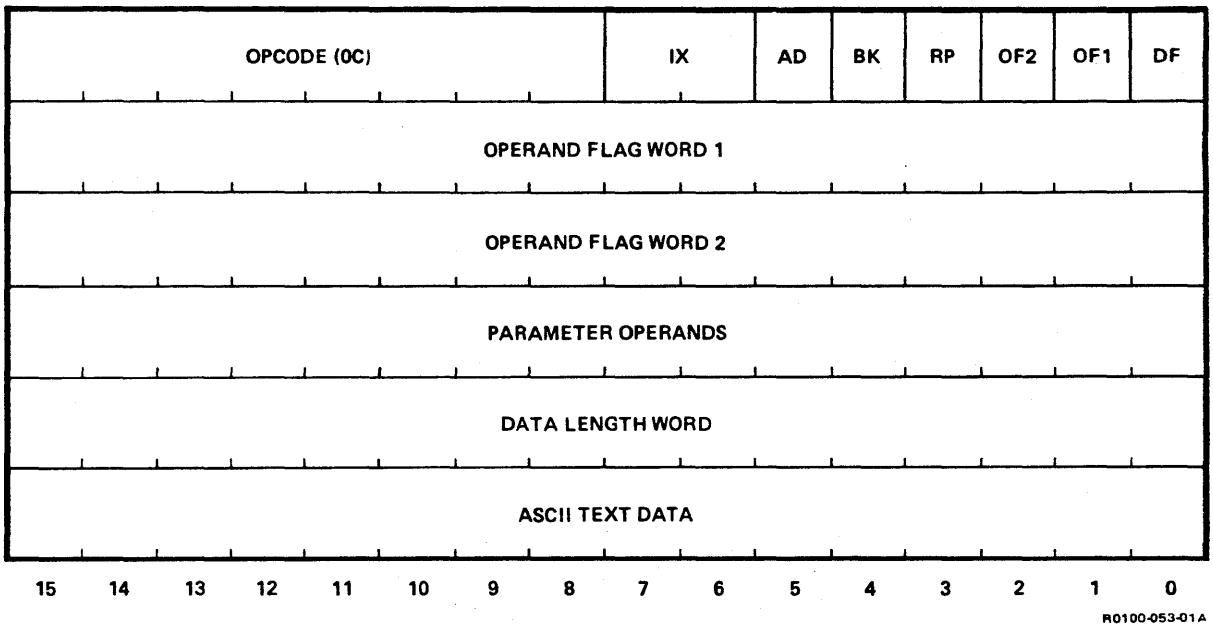
| Errors | Description |
|---|---|
| Code = 0B01 | Attempt to execute RI from within a display list. |
| Code = 0B07 | Odd byte count in word mode. |
| Code = 0B17 | No MPCs selected to read from. |
| Code = 0B18 | Too many MPCs selected to read from. |
| Code = 0B19 | No group selected to read from. |
| Code = 0B1A | Too many groups selected to read from. |

Example:

This example demonstrates use of the RI instruction and word mode. Assume that the data from the write image word mode example has been loaded into refresh memory. The instruction stream outlined below will read back that data.

| TRANSFER TYPE | HEX | MNEMONIC | DESCRIPTION |
|---|---|---|---|
| DATA | 0B07 | RI+OF2+OF1+DF | ;Read Image Instruction |
| DATA | 0040 | WIN | ;Operand Flag 1 |
| DATA | 0020 | IMG | ;Operand Flag 2 |
| DATA | 0064 | 100 | ;Re-transmit the |
| DATA | 0064 | 100 | ;Window Values |
| DATA | 0067 | 103 | ;To Set the COP to |
| DATA | 0067 | 103 | ;Correct SCAN Corner |
| DATA | 0000 | 0 | ;Image Mode = Word Mode |
| DATA | 0020 | 0032 | ;Data Length Word = 32 |
| INTERFACE COMMAND | 0400 | PREFETCH | ;Signal Readback to Interface |
| | | Read 16 Data Values, One per Word | |
| INTERFACE COMMAND | 0000 | | ;Clear Prefetch |

## WRITE TEXT (WT)




WT is a normal-format instruction which reads ASCII character codes from the host processor and generates the corresponding text characters into the rectangular area defined by FORMAT WINDOW. ASCII character codes 20(H) through 9F(H) are available and may be loaded with host processor-defined font patterns. (See figure C-10 for the ASCII characters and their associated codes.) Upon reset (either hardware reset or RM-9460 RSET instruction) programmable font 0 is allocated with size 8 X 12 and the character font descriptions for codes 20(H) through 9F(H) are copied to the programmable font RAM storage area. Three control character codes are recognized: carriage return 0D(H), line feed 0A(H) and backspace 08(H). Any character code received by the RM-9460 outside of the above-defined ranges generates a character of indeterminate font.

| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, FORMAT WINDOW, and START-POINT are evaluated. |
| AD | Affects the generation of text data in conjunction with the BK flag. Characters are generated by a font consisting of ones and zeros. The AD flag inhibits the writing of pixels within the character font in the manner specified in figure 2-13. |

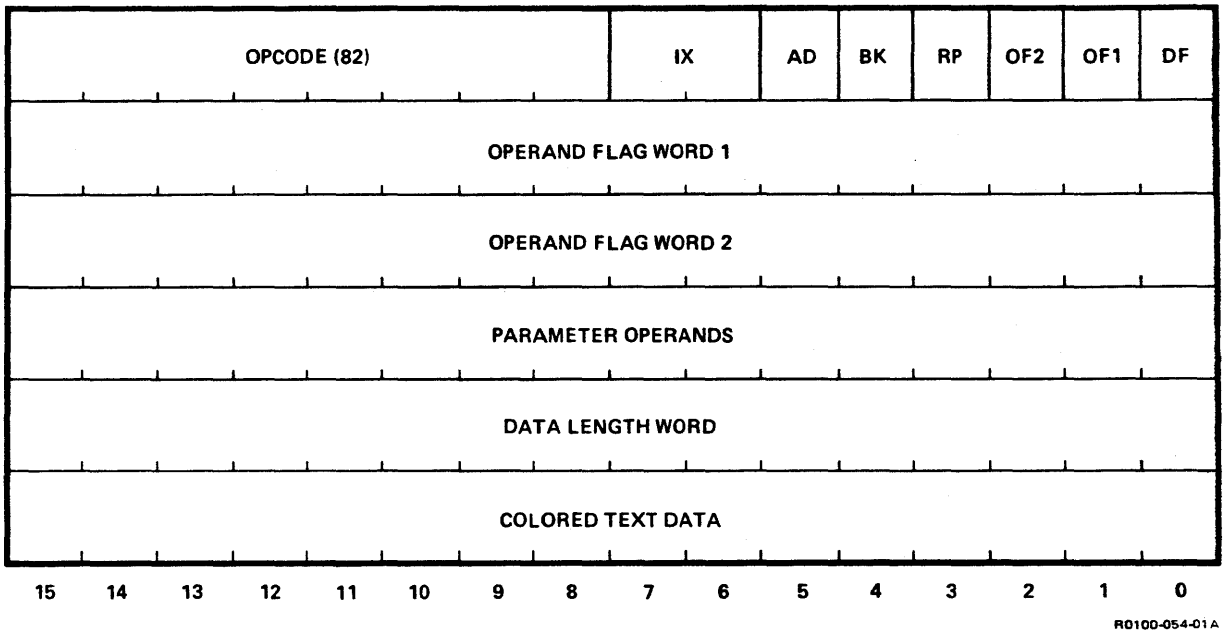
Although the FORMAT WINDOW defines the rectangular area for text generation, it is possible to generate part of a character outside this area. If the first pixel of any character to be generated is within the FORMAT WINDOW area, the entire character is generated. It is possible for up to (DIMENSION width * height - 1) pixels to be generated outside the FORMAT WINDOW.

It is also possible to generate single characters outside the FORMAT WINDOW by sending WRITE TEXT instruction with a COP and a single character. If this is attempted with more than one character in a WRITE TEXT instruction, the first character appears at the specified COP and the rest of the characters appear inside the FORMAT WINDOW.

The carriage return, OD(H), causes an immediate return to the window boundary in the direction opposite to the primary update direction as well as an update in the secondary update direction. The line feed code, OA(H), causes an immediate update in the secondary update direction. The backspace code causes an immediate update in the direction opposite to the primary update direction. If a window boundary is crossed during this update, the COP is set to that window boundary, but no secondary updating is performed. After a character has been generated and SPACING update applied, the actions for carriage return are performed if the COP is outside the FORMAT WINDOW. If the carriage return or line feed secondary updates cause the COP to be outside the FORMAT WINDOW, then the COP is returned to the window boundary in the direction opposite to the secondary update direction.

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled and will receive text data. |
| FOREGROUND | Defines the foreground color or intensity value to be written to refresh memory when selected by the text data bit value and the AD and BK flags. |
| BACKGROUND | Defines the background color or intensity value to be written to refresh memory when selected by the text data bit value and the AD and BK flags. |
| INDEX 1 | Displaces the values to be used for INDEX 2, FORMAT WINDOW, and START-POINT when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for FORMAT WINDOW and START-POINT when IX = 10(B). |
| FORMAT WINDOW | Defines the rectangular area where text data is to be written. If the first pixel of any character is within this area, the entire character is drawn, even though part of the character may exceed the FORMAT WINDOW boundaries. If START-POINT is not explicitly set in the WT instruction, FORMAT WINDOW and SCAN define the starting pixel coordinate. |

## WRITE RANDOM COLORED TEXT (WRCT)



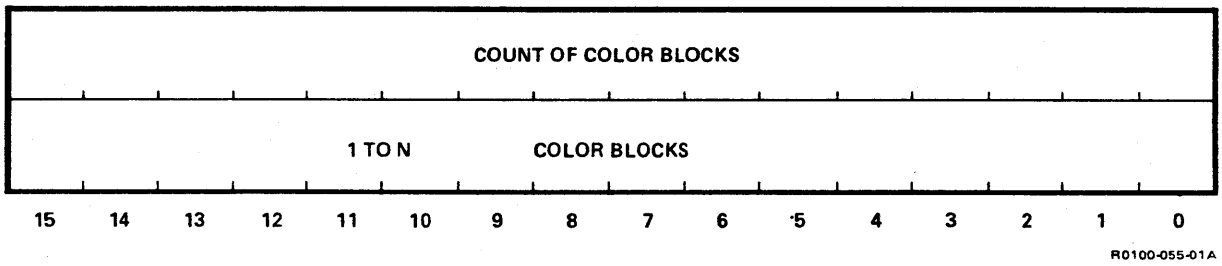

WRCT is a normal-format instruction which reads multiple ASCII character strings from the host processor and generates the corresponding text characters into the location specified in each text string. The FOREGROUND parameter may be changed for each text string, or multiple text strings can be generated using the same FOREGROUND value. The rectangular area, where the text strings can be written, is not defined by the FORMAT WINDOW; therefore, text strings can be generated anywhere. ASCII character codes 20(H) through 9F(H) are available and may be loaded with font patterns defined by the host processor. Upon reset (either hardware or through RSET instruction), programmable font 0 is allocated with a size 8 X 12 and the character font descriptions for codes 20(H) through 9F(H) are copied to the programmable font RAM storage area. Any character code received by the RM-9460 outside of the above defined ranges generates a character of indeterminate font.

The character return code, 0D(H), causes an immediate return to the left-most position supplied in the current text string X-START, Y-START in the direction opposite to the primary update direction, as well as an update in the secondary update direction. The line feed code, 0A(H), causes an immediate update in the secondary update direction. The backspace code causes an immediate update in the direction opposite the primary update direction. If the left-most position is crossed during this update, the COP is set to that left-most position, but no secondary updating is performed.

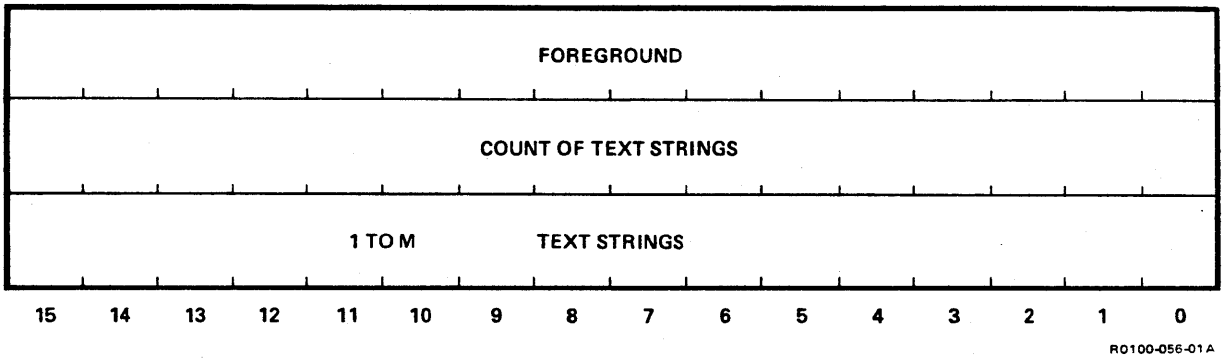
The difference between WRCT and WT instructions is that WRCT can write multiple text strings starting from the supplied locations while WT can write only a single text string starting from X-COP and Y-COP. Use a WRCT instruction instead of multiple WT instructions to write multiple text strings.

| Parameters | Description |
|---|---|
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, writing to refresh memory is enabled; otherwise, writing to refresh memory is disabled. |
| DATA LENGTH | Consists of the total number of bytes of data sent with the WRCT instruction. This data consists of a count of text color blocks, the FOREGROUND value for each color block, the number of text strings in each block, and the text string data. DATA LENGTH WORD may take on any value from 0 to 66535. |
| COUNT OF COLOR BLOCKS | Defines the number of color blocks sent from the host. |
| COLOR BLOCK DATA | Consists of a FOREGROUND value and one or more text strings. |
| FOREGROUND | Defines the FOREGROUND value to be used when generating the following text strings. The BACKGROUND value remains constant. |
| COUNT OF TEXT STRINGS | Defines the number of text strings sent from the host. |
| TEXT STRING DATA | Consists of one or multiple text strings. Each of these includes X-START, Y-START, CHARACTER COUNT, and ASCII text data. |
| X-START | Defines the X-coordinate of the first pixel of the first text character to be written into refresh memory. |
| Y-START | Defines the Y-coordinate of the first pixel of the first text character to be written into refresh memory. |
| CHARACTER COUNT | Represents the number of ASCII characters transmitted from the host processor to the RM-9460 within this text string. |

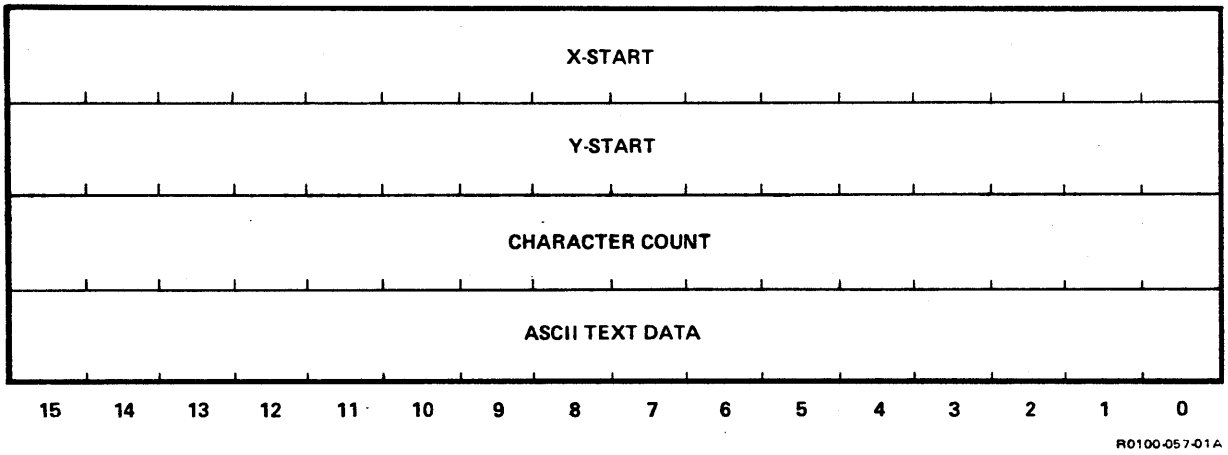
DATA FORMAT




COLOR BLOCK FORMAT



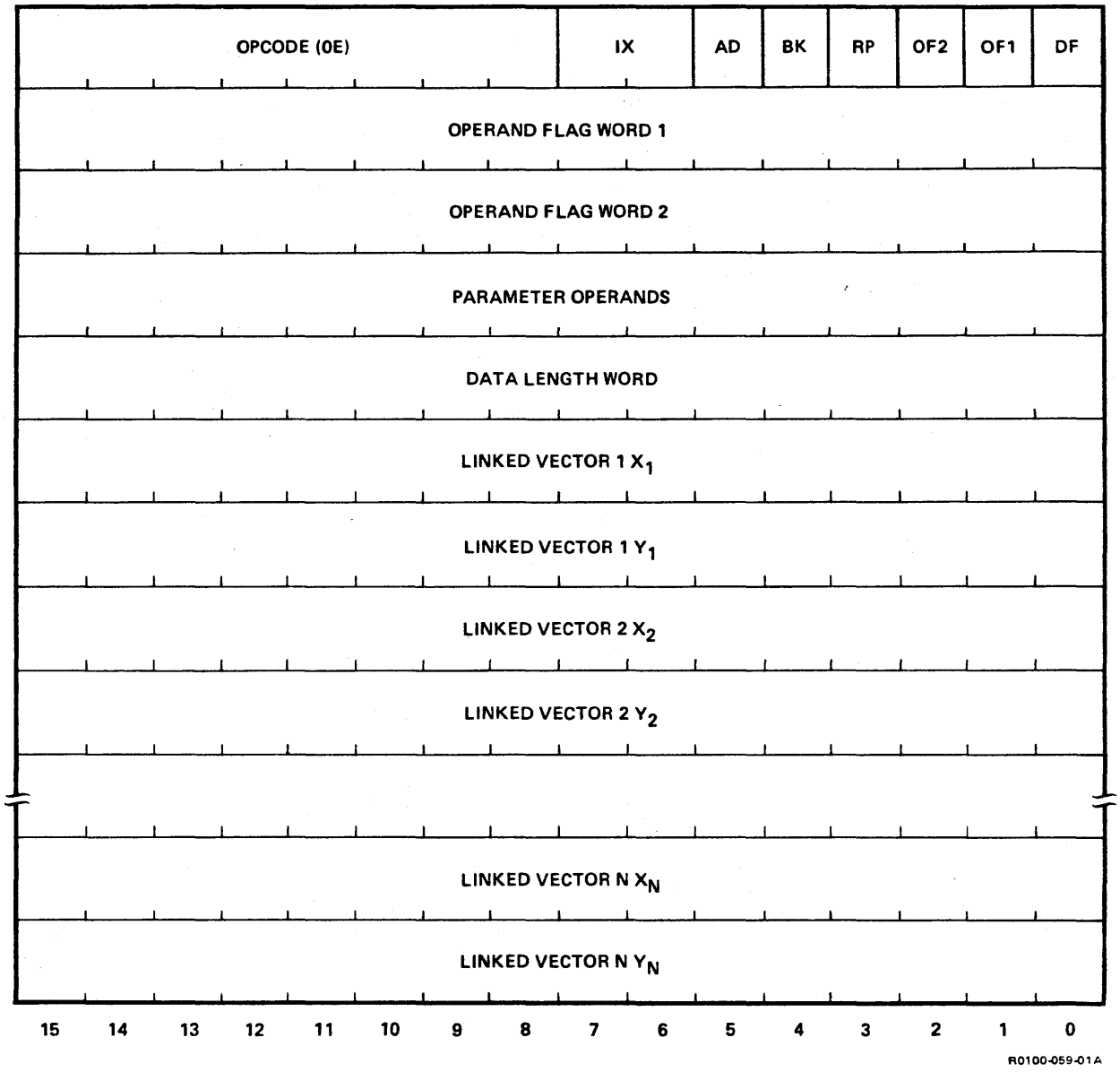

TEXT STRING FORMAT



| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, FORMAT WINDOW, and START-POINT are evaluated. |
| AD | Affects the generation of raster data in conjunction with the BK flag. When AD = 1, writing of zero raster data into refresh memory is inhibited. |
| BK | Defines the selection of foreground and background colors based on the ones/zeros data bits within each raster byte and on the value of the AD flag. |
| RP | Defines the byte-accessing order for each 16-bit word of raster data from the host processor. If RP = 0, the high byte is processed before the low byte; if RP = 1, the low byte is processed first. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled and will receive raster data. |
| FOREGROUND | Defines the foreground color or intensity value to be written to refresh memory when selected by the raster data bit value and the AD and BK flags. |
| BACKGROUND | Defines the background color or intensity value to be written to refresh memory when selected by the raster data bit value and the AD and BK flags. |
| INDEX 1 | Displaces the values to be used for INDEX 2, FORMAT WINDOW, and START-POINT when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for FORMAT WINDOW and START-POINT when IX = 10(B). |
| FORMAT WINDOW | Defines the rectangular area where raster data is written. No raster data is written outside this area. If START-POINT is not explicitly set in the WR instruction, FORMAT WINDOW and SCAN define the starting pixel coordinate. |
| SCAN | Defines the primary and secondary raster update directions as well as the starting pixel coordinate when START-POINT is not explicitly set in the WR instruction. |
| SIZE | Defines the received-bit-to-displayed-pixel ratio for raster data. |

## WRITE VECTOR LINKED (WVL)




WVL is a normal-format instruction which draws a series of linked vectors. In the linked vectors, each coordinate pair in the data stream represents the endpoint of a vector with an origin at the previous vector coordinate pair or at the current context's COP for the first vector in the instruction. The DATA LENGTH WORD must be four times the number of vectors to be drawn, since each vector requires four bytes to represent the endpoint coordinate.

Data Format

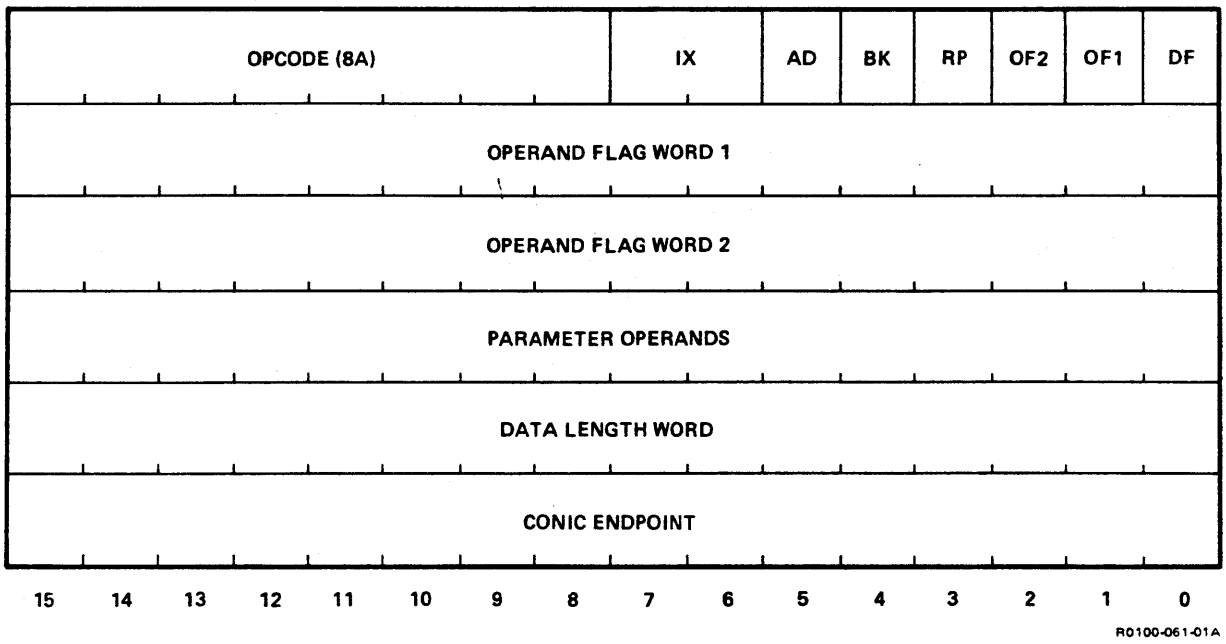
The vector endpoint data is organized into coordinate pairs, one per vector (four bytes per vector). The total amount of data associated with the WVL instruction (the value of the DATA LENGTH WORD) must be a multiple of four.

| Errors | Description |
|---|---|
| Code = 0E07 | Illegal value for DATA LENGTH WORD; not a multiple of four. |

| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, and START-POINT parameter operands set in the WC instruction are evaluated. Also, if a conic endpoint is specified as data in the WC instruction, IX affects the address mode where the coordinate is evaluated. |
| BK | Defines the selection of FOREGROUND or BACKGROUND value for the color or intensity value of the conic being drawn. If BK = 0, the conic is generated with the FOREGROUND value; otherwise the BACKGROUND value is used. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled to receive conic data. |
| FOREGROUND | Defines the color or intensity value of the conic when BK = 0. |
| BACKGROUND | Defines the color or intensity value of the conic when BK = 1. |
| INDEX 1 | Displaces the values to be used for INDEX 2, START-POINT, and the conic endpoint when IX = 01(B). |
| INDEX 2 | Displaces the value to be used for START-POINT and the conic endpoint when IX = 10(B). |
| CONIC-EQUATION | Specifies the conic-defining coefficients A, B, C, D, and E and the number of pixels to be drawn in the conic K. |
| START-POINT | Defines the origin of the conic to be generated. START-POINT is evaluated in the address mode defined by IX. |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DISPLAY CLASS | Defines the current DCL value. If this value is within one of the defined display-class ranges, writing into refresh memory is enabled; otherwise, writing into refresh memory is disabled. |

## WRITE CONIC 32 BITS (WC32)




WC32 is a normal-format instruction which draws circles, parabolas, ellipses, and hyperbolas, or sections thereof, based upon the CONIC-EQUATION parameter operands. The generalized conic equation used is $Ax^2 + By^2 + Cxy + Dx + Ey = 0$, where A, B, C, D, and E represent the first five CONIC-EQUATION parameter operands.

The last CONIC-EQUATION parameter operand, K, represents the total number of pixels to be generated by the WC32 instruction for the conic specified. A default value of 1280 is used for K.

The difference between WC and WC32 is that WC uses only the low 16 bits of each conic parameter (A, B, C, D, E) and all internal calculations are done in 16-bit arithmetic, while WC32 uses all 32 bits of the coefficients and calculates in 32-bit arithmetic. With some conic equations, 16-bit arithmetic does not have enough significance and underflow or overflow can occur, leading to incorrect results. WC32 can be used in these cases since full 32-bit arithmetic is used throughout.

The START-POINT parameter operand is used by the WC32 instruction to define the Cartesian origin of the conic or conic section to be drawn. All conics drawn pass through the Cartesian origin defined by START-POINT. If a conic endpoint is specified, the absolute coordinates of this pixel are compared to the coordinates of each pixel in the conic as it is being generated. If a pixel in the conic being drawn has the same coordinates as specified by the endpoint, the conic is terminated at that point.

The WC32 instruction is not affected by the FORMAT WINDOW parameter operand.

| Parameters | Description |
|---|---|
| DISPLAY CLASS | Defines the current DCL value. If this value is within one of the defined display-class ranges, writing into refresh memory is enabled; otherwise, writing into refresh memory is disabled. |
| DATA LENGTH WORD | Defines the total number of bytes contained in the WC32 following DATA LENGTH WORD. Since only one conic can be generated per WC32 instruction, it is suggested that DATA LENGTH WORD (if present due to DF = 1) always take on a value of four. This indicates that a single conic endpoint is present in the WC32 instruction data stream. If more than four bytes are indicated by DATA LENGTH WORD, all additional bytes are discarded. Since coordinates are composed of full words, it is necessary that this parameter always reflect an even number of bytes. If DATA LENGTH WORD is odd, an illegal instruction interrupt is generated and all data is discarded. |

## Data Format

The first 16-bit word pair is interpreted as the conic endpoint. All remaining data is discarded. The first 16-bit word is the element coordinate of the conic endpoint. The second is the line coordinate. The conic endpoint coordinate is interpreted in the address mode defined by IX.

## COP Movement

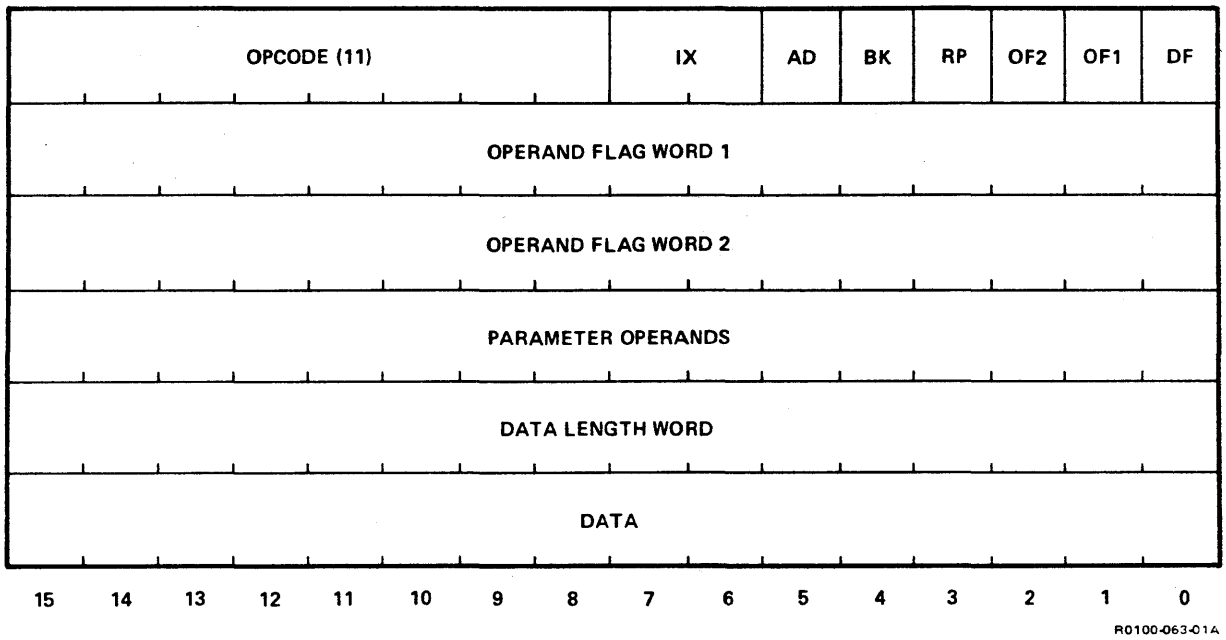
The current operating point after execution of the WC32 instruction is the coordinate of the last pixel of the conic generated.

| Errors | Description |
|---|---|
| Code = 8A07 | Odd byte count. |

| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, BASELINE, and START-POINT parameter operands are evaluated. |
| BK | Defines the color or intensity used for the plot segments drawn. If BK = 0, the FOREGROUND value is used; if BK = 1, the BACKGROUND value is used. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled to receive plot data. |
| FOREGROUND | Defines the color or intensity value for all plot segments when BK = 0. |
| BACKGROUND | Defines the color or intensity value for all plot segments when BK = 1. |
| INDEX 1 | Displaces the values to be used for INDEX 2, BASELINE, START-POINT, and all WPB data when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for BASELINE, START-POINT, and all WPB data when IX = 10(B). |
| SCAN | Defines the plot orientation. If SCAN $\leq$ 3, the plot axis is the Y-axis (horizontal). If SCAN $\geq$ 4, the plot axis is the X-axis (vertical). If SCAN is even, each plot segment is drawn in a positive direction (for horizontal plots, toward increasing X). If SCAN is odd, each plot segment is drawn in a negative direction (for horizontal plots, toward decreasing X). |
| DIMENSION | Defines the size of a plot segment parallel to the plot axis. If SCAN $\leq$ 3, DIMENSION width is used. If SCAN $\geq$ 4, DIMENSION height is used. |
| SPACING | Defines the distance between starting COP for successive plot segments along the plot axis. If SCAN $\leq$ 3, X-spacing is used. If SCAN $\geq$ 4, Y-spacing is used. |
| BASELINE | Defines the fixed-line plot axis if BASELINE is nonzero. If SCAN $\leq$ 3, BASELINE represents the Y-coordinate of the plot axis. If SCAN $\geq$ 4, BASELINE represents the X-coordinate of the plot axis. |

## SCROLL X (SCRX)




SCRX is a normal-format instruction which performs a scroll of image data within the rectangular area defined by the FORMAT WINDOW parameter operand along the X-axis. The sign of the SCROLL-COUNT parameter operand defines the direction of scroll. If SCROLL-COUNT is positive, scrolling is to the right toward increasing element coordinate values. If SCROLL-COUNT is negative, scrolling is to the left toward decreasing element coordinate values. The absolute value of SCROLL-COUNT defines the number of elements the image data within the FORMAT WINDOW area will be moved. A SCROLL-COUNT of zero results in no scrolling operation. Only those refresh memory planes selected by the WRITE MASK are scrolled.

Data scrolled out of the window is discarded. If BK = 0, the data scrolled into the FORMAT WINDOW area is taken from the BACKGROUND parameter operand. If BK = 1, fill data is taken from the FOREGROUND parameter operand. If AD = 1, no fill is performed. If the absolute value of SCROLL-COUNT is greater than the FORMAT WINDOW width, the result is an erasure of the FORMAT WINDOW area to either the FOREGROUND or BACKGROUND value, depending upon the value of BK.

| Parameters | Description |
|---|---|
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, writing to refresh memory is enabled; otherwise, writing into refresh memory is disabled. |
| DATA LENGTH WORD | Defines the number of bytes of data present in the SCRX instruction stream when DF = 1. |

Data Format

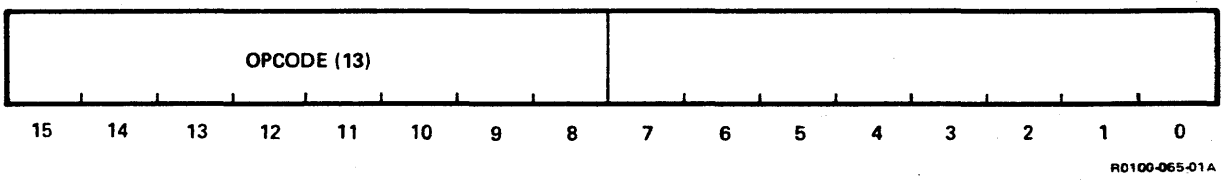
All data present in the SCRX is discarded.

| Errors | Description |
|---|---|
| Code = 1102 | No (scratch) RAM available. |

| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, and the FORMAT WINDOW parameter operands are evaluated. |
| AD | Allows the user to inhibit filling of the FORMAT WINDOW area where image data has been scrolled. If AD = 1, this will not be performed. |
| BK | Defines the color or intensity value to be used to fill the subregion of the FORMAT WINDOW region from which data is scrolled. If BK = 0, the BACKGROUND value is used; otherwise, the FOREGROUND value is used. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are scrolled; they receive fill data when AD = 0. |
| FOREGROUND | Defines the color or intensity value to be used as fill data when BK = 1 and AD = 0. |
| BACKGROUND | Defines the color or intensity value to be used as fill data when BK = 0 and AD = 0. |
| INDEX 1 | Displaces the values to be used for INDEX 2 and FORMAT WINDOW parameter operands when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for the FORMAT WINDOW parameter operand when IX = 10(B). |
| FORMAT WINDOW | Defines the rectangular area where image data is vertically scrolled. |
| SCROLL-COUNT | Defines the number of lines to scroll image data within the rectangular FORMAT WINDOW area. SCROLL-COUNT is defined as a two's-complement number whose absolute value is used as the number of scrolling lines. If SCROLL-COUNT is positive, scrolling is toward increasing line numbers. If SCROLL-COUNT is negative, scrolling is toward decreasing line numbers. A SCROLL-COUNT of 0 results in no scrolling operation. |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |

## SAVE ENVIRONMENT (PUSHE)



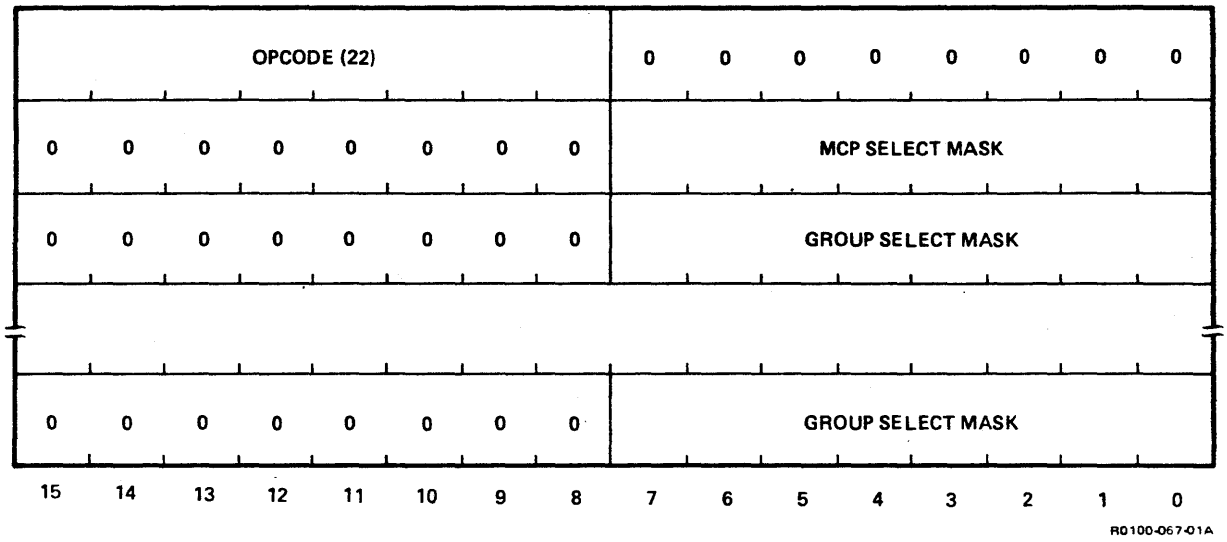

PUSHE is a special-format instruction which saves all parameter operands and their associated internal parameters on an internal stack. This internal stack uses a LIFO (last-in-first-out) structure. Up to five consecutive PUSHE instructions without intervening POPE instructions are allowed (up to five environments may be saved on the internal stack). If more than five saves are attempted by the user, an error interrupt is generated and the current environment is not saved on the internal stack.

The following values are stored on the stack with each PUSHE:

- All normal format parameters
- The current transformation matrix
- The current global X- and Y-COP values

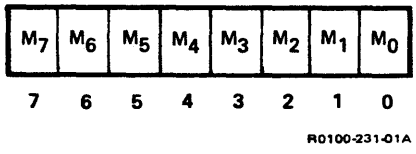
| Errors | Description |
|---|---|
| Code = 130A | No room on environment stack. |

## SELECT MCP/GROUP (SELMG)




SELMG is a special-format instruction which defines which refresh memory groups will be associated with the current context. This variable-length instruction specifies both the MCPs (through the MCP SELECT MASK) and the groups associated with each selected MCP (through the GROUP SELECT MASK). The default MCP/group selection at power-on reset is MCP 0 and group 0.

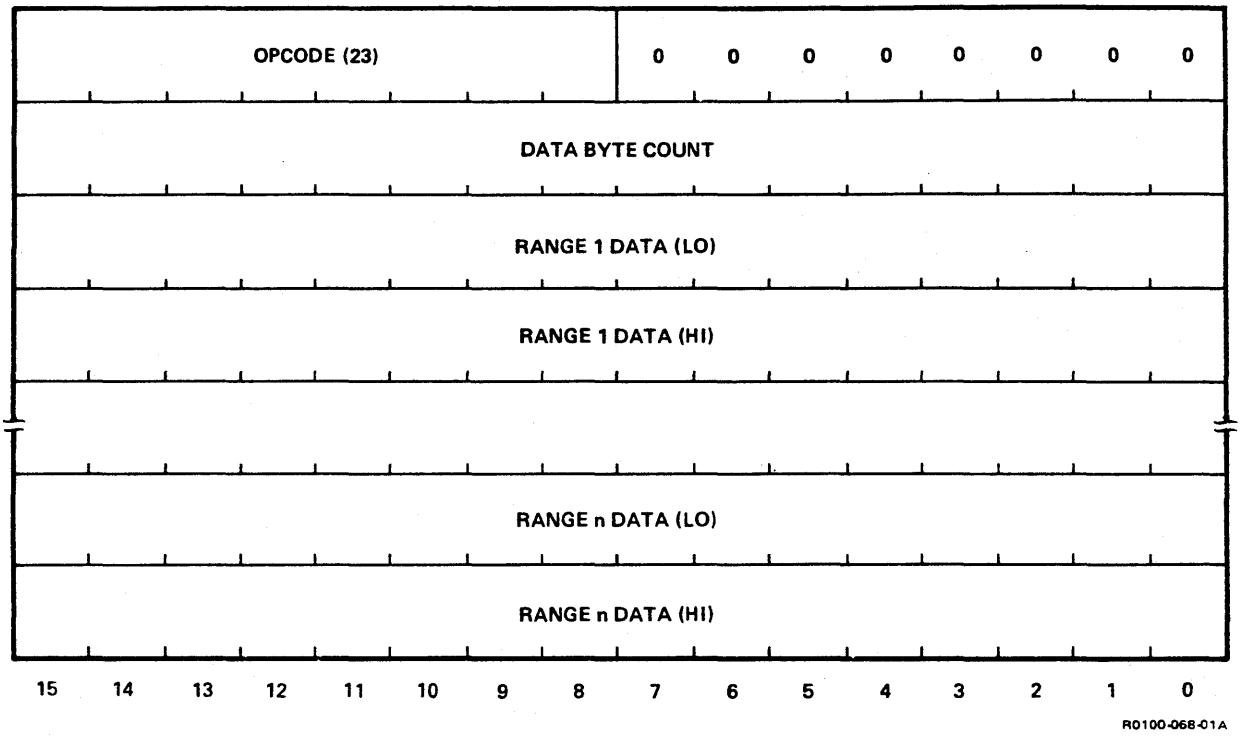
| Parameters | Description |
|---|---|
| MCP SELECT MASK | Defines the MCPs associated with the current context. The format of this mask is: |




where $M_i$ represents $MCP_i$

For each bit set in this mask, there is an accompanying GROUP SELECT MASK BYTE.

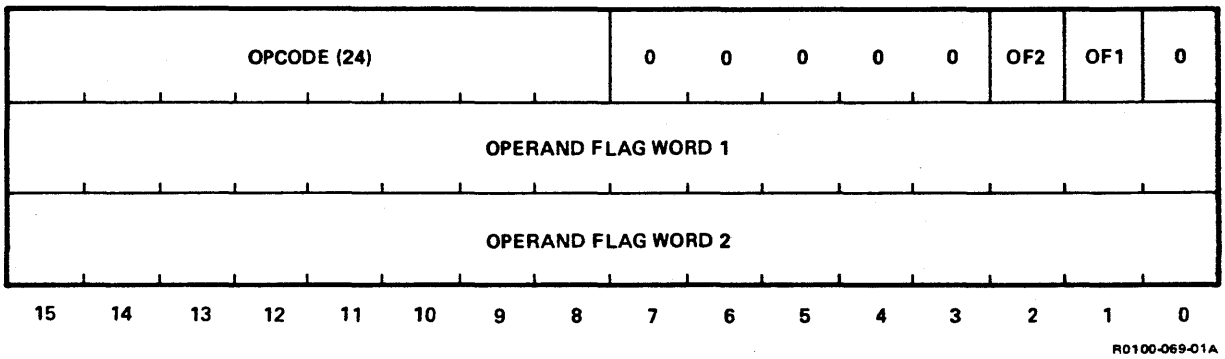
## SET DISPLAY CLASS RANGES (SETDC)




SETDC is a special-format instruction which defines the ranges of DISPLAY CLASS values that enable writing of data into refresh memory. Up to 16 ranges may be defined in this instruction. When the current value of the DISPLAY CLASS operand falls into one of these ranges, writing into refresh memory is permitted; otherwise, instruction execution continues normally but no data is written into refresh memory.

| Parameters | Description |
|---|---|
| DATA BYTE COUNT | Defines the number of range pairs to follow. This value must be a multiple of four, since there are four bytes of data per range pair. If the value is not a multiple of four, or is greater than 64, an error interrupt is generated and the data is discarded. |

## READ NORMAL PARAMETERS (READP)




READP is a special-format instruction which allows the user to read back the parameter values for the current context. The values are selected and returned to the user in the same manner and order as set in any normal-format instruction. There are currently 94 bytes of parameter data that can be read back. The user must make sure that the correct number of bytes is read back; otherwise, interprocessor communication will stop.

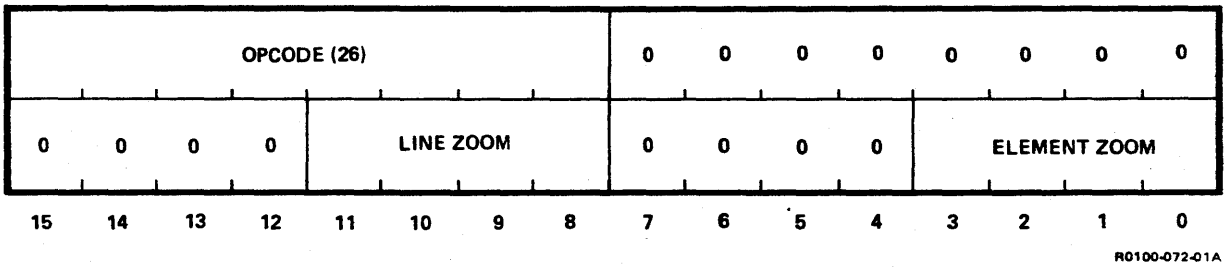
| Errors | Description |
|---|---|
| Code = 2401 | Attempt to execute READP from within a display list. |

Example:

The sequence READP + OF1
003F(H)

returns 18 bytes of parameter data to the host. This data represents the current values of WRITE MASK, FOREGROUND, BACKGROUND, INDEX 1, INDEX 2, and ORIGIN parameter operands.

## ZOOM (ZOOM)




ZOOM is a special-format instruction which sets the hardware zoom registers for all MCPs selected under the current context. The zoom feature is a hardware function using pixel replication by a factor of from 2 to 16. The replication is accomplished on an MCP basis. All groups under an MCP are zoomed, including those that may be associated with a context other than the current one.

| Parameters | Description |
|---|---|
| ELEMENT ZOOM | Defines the pixel replication factor in the horizontal axis. A value of 0 represents no replication, 1 represents a replication of two video pixels per actual pixel, ..., and 15 represents a replication of 16 video pixels per actual pixel. |
| LINE ZOOM | Defines the pixel replication factor in the axis. The values are analogous to the ELEMENT ZOOM. |

### Errors

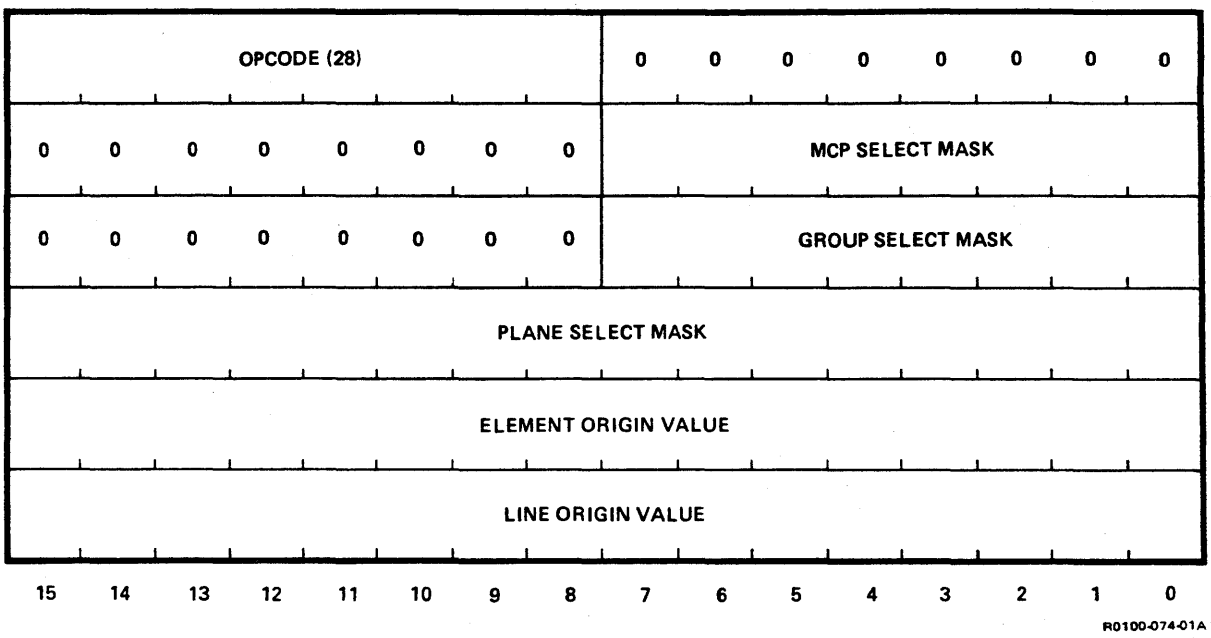
None

### Example:

The sequence
```
ZOOM
0203(H)
```

represents a pixel replication of 4:1 in the horizontal axis and 3:1 in the vertical axis.

## LOAD SUBCHANNEL ORIGINS (LOADSO)

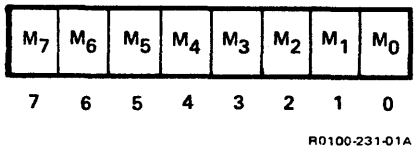



LOADSO is a special-format instruction which allows the setting of origins for individually-scrollable memory planes. If the specified plane is not individually assigned, instruction execution takes place but has no effect on the video.

**NOTE**

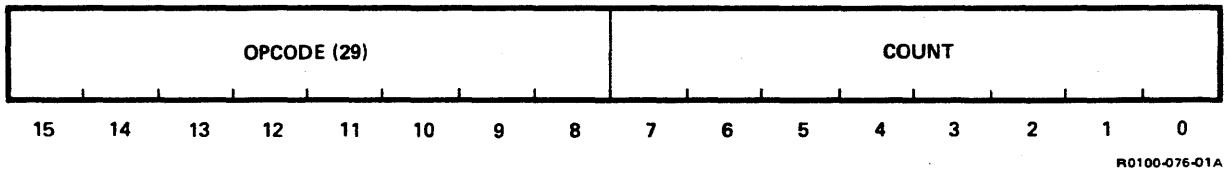
This instruction is for use with the INDEPENDENT SCROLLABLE REFRESH MEMORY PLANES only.

| Parameters | Description |
|---|---|
| MCP SELECT MASK | Defines the MCP(s) which contain(s) the planes to be loaded. The format is: |




where $M_i$ represents $MCP_i$

## WAIT FOR VERTICAL RETRACE (WAITVR)



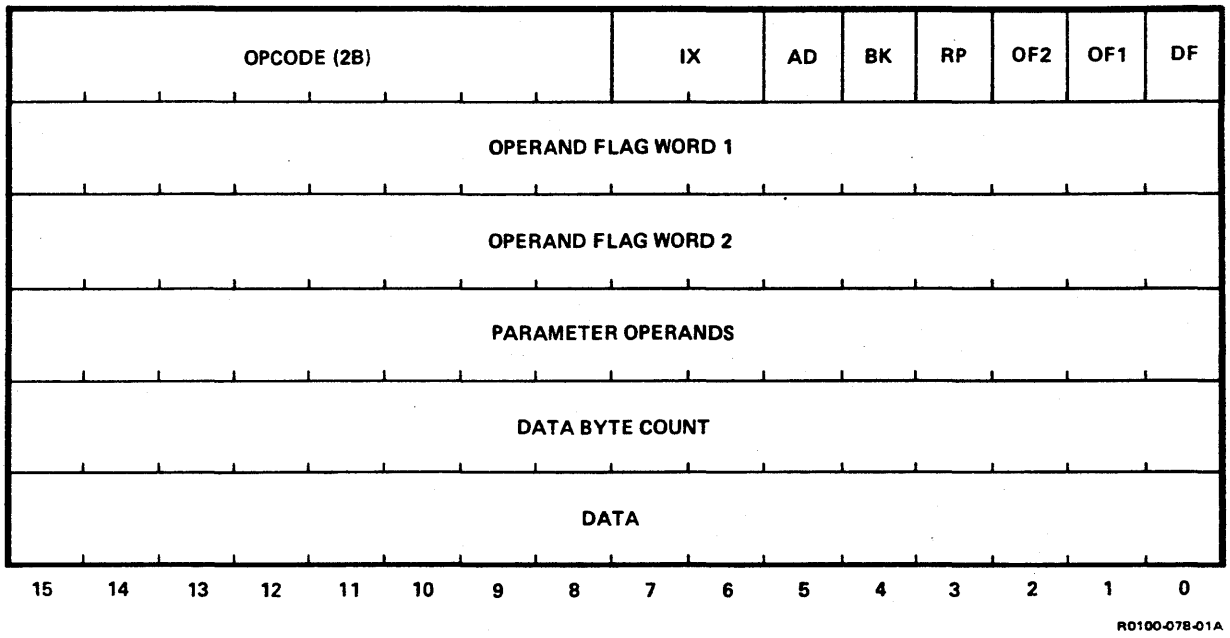

WAITVR is a special-format instruction which suspends execution of display instructions until the specified number of vertical retrace intervals have occurred.

| Parameter | Description |
|---|---|
| COUNT | Defines the number of vertical retrace intervals to wait. If COUNT = 0, no wait occurs. The value of COUNT must be less than 256. |

Errors

None

## BULK ERASE (BERS)

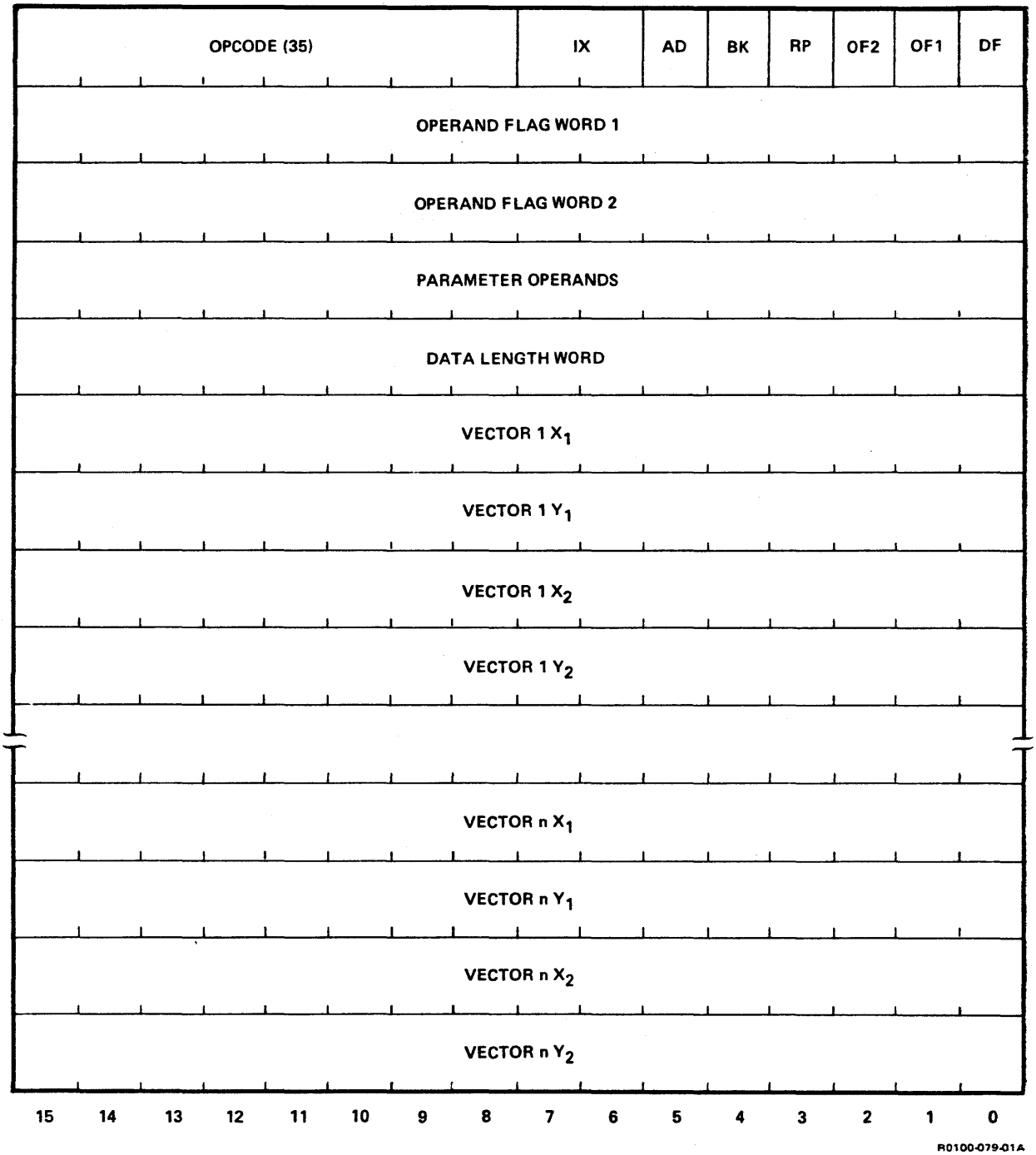



BERS is a normal-format instruction which sets all refresh memory in the groups selected under the current context to either the FOREGROUND or the BACKGROUND value based on the state of BK. If BK = 0, the BACKGROUND value is written. If BK = 1, the FOREGROUND value is written. If AD = 1, no data is written to refresh memory. This writing is under control of the WRITE MASK parameter operand. All data associated with the BERS instruction is discarded.

| Flags | Description |
|---|---|
| BK | Defines the value to be written into refresh memory. If BK = 0, the BACKGROUND value is used. If BK = 1, the FOREGROUND value is used. |
| AD | Affects the writing of data into refresh memory. If AD = 1, no writing is accomplished. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the memory plane write-enable mask. Only memory planes whose corresponding bit in the WRITE MASK is set will be erased. |
| FOREGROUND | Defines the data value written into refresh memory if BK = 1. |

## WRITE VECTOR UNLINKED (WVU)



| Parameters | Description |
|---|---|
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, writing into refresh memory is enabled; otherwise, writing into refresh memory is disabled. |

## Data Format

The vector endpoint data is organized into two sets of coordinates, or four words per vector. The total amount of data associated with any WVU instruction should always be a multiple of eight.

| Errors | Description |
|---|---|
| Code = 3507 | Illegal value for DATA LENGTH WORD; not a multiple of eight. |

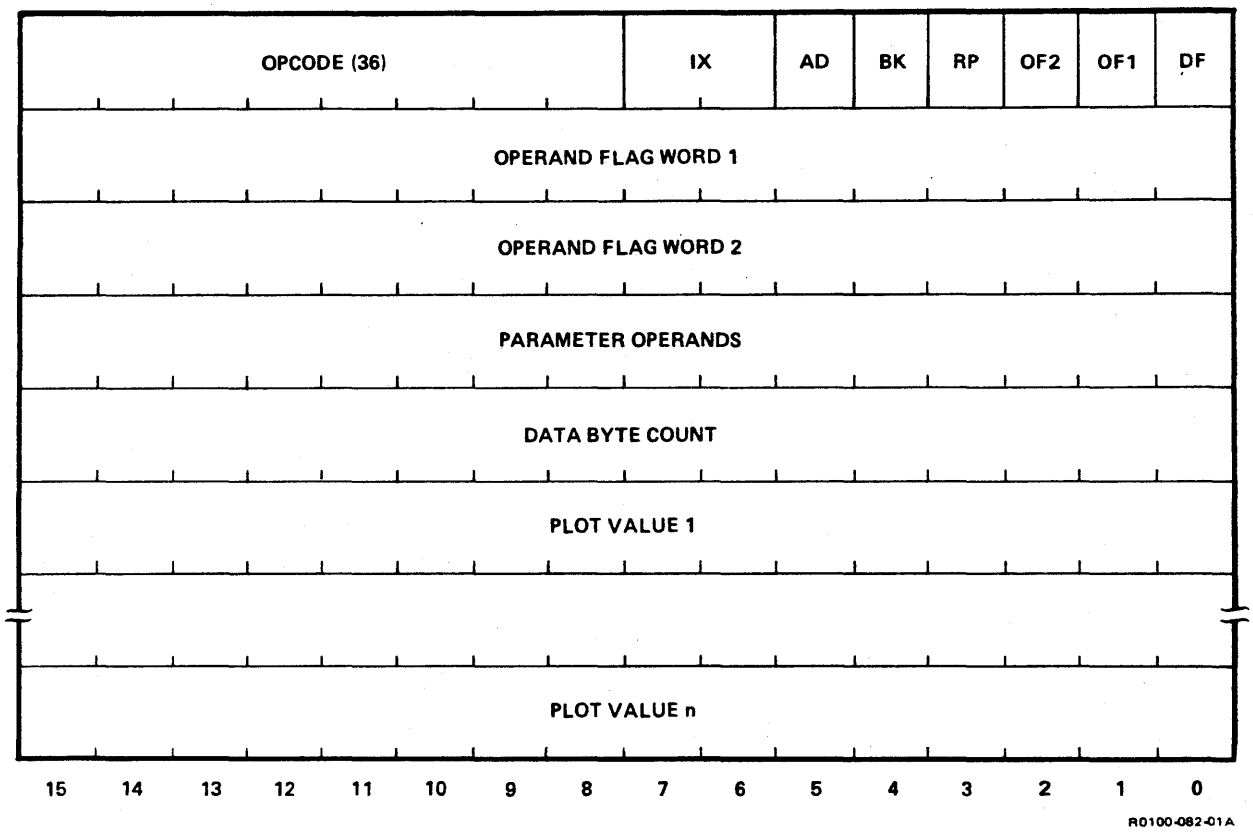
WCVU is a normal-format instruction which draws a series of random vectors, where each vector has specified the FOREGROUND color and the start and endpoint coordinates of the vector.

The DATA LENGTH WORD must be ten times the number of vectors to be drawn, since each vector requires ten bytes.

| Flags | Description |
|---|---|
| IX | Defines the addressing mode in which INDEX 1, INDEX 2, and all endpoints are evaluated. |
| AD | Affects the generation of vector data in conjunction with the BK flag and the VECTOR TEXTURE PATTERN parameter. When AD = 1, writing zero pattern data bits into refresh memory is inhibited. |
| BK | Affects the interpretation of bits in the VECTOR TEXTURE PATTERN parameter operand. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes enabled for the writing of data. |
| BACKGROUND | Defines the vector color/intensity data to be written into refresh memory for logical 0 data in the VECTOR TEXTURE PATTERN if BK = 0, and for logical 1 if BK = 1. |
| INDEX 1 | Displaces the values to be used for the INDEX 2 parameter operand and all vector endpoints when IX = 01(B). |
| INDEX 2 | Displaces the values of all vector endpoints when IX = 10(B). |
| WRITE-ENABLE | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| VECTOR TEXTURE PATTERN | Defines the pattern to be used to generate vector data. |
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, writing into refresh memory is enabled; otherwise, writing into refresh memory is disabled. |

## WRITE PLOT POINT (WPP)




WPP is a normal-format instruction which is used to generate single pixel plots, receiving a coordinate in one axis and automatically updating the coordinate for the next point in the other axis. This can reduce the number of words of data necessary for a pixel plot by a factor of two. The orientation of the plot (differentiating between X-axis and Y-axis plots) is defined by the SCAN parameter.

Initially the current COP is used, along with the first plot data word, to define the first pixel to be written. For example, (assuming IX = 00) if the current COP value is $(X_C,Y_C)$ and the first data word is $D_1$ then the first point written is $(X_C,D_1)$ for horizontal plots and $(D_1,Y_C)$ for vertical plots. After each pixel is drawn, the COP coordinate parallel to the plot axis is updated by adding the SPACING value for that axis.

The color or intensity of the plotted pixels is defined by either FOREGROUND (BK = 0) or BACKGROUND (BK = 1) and is masked by the WRITE MASK parameter.

| Parameters | Description |
|---|---|
| DATA LENGTH WORD | Defines the number of bytes of point-plot data to follow in the WPP instruction. Since each point-plot data value is a 16-bit word, the DATA LENGTH WORD must always reflect an even number of bytes. |

## Data Format

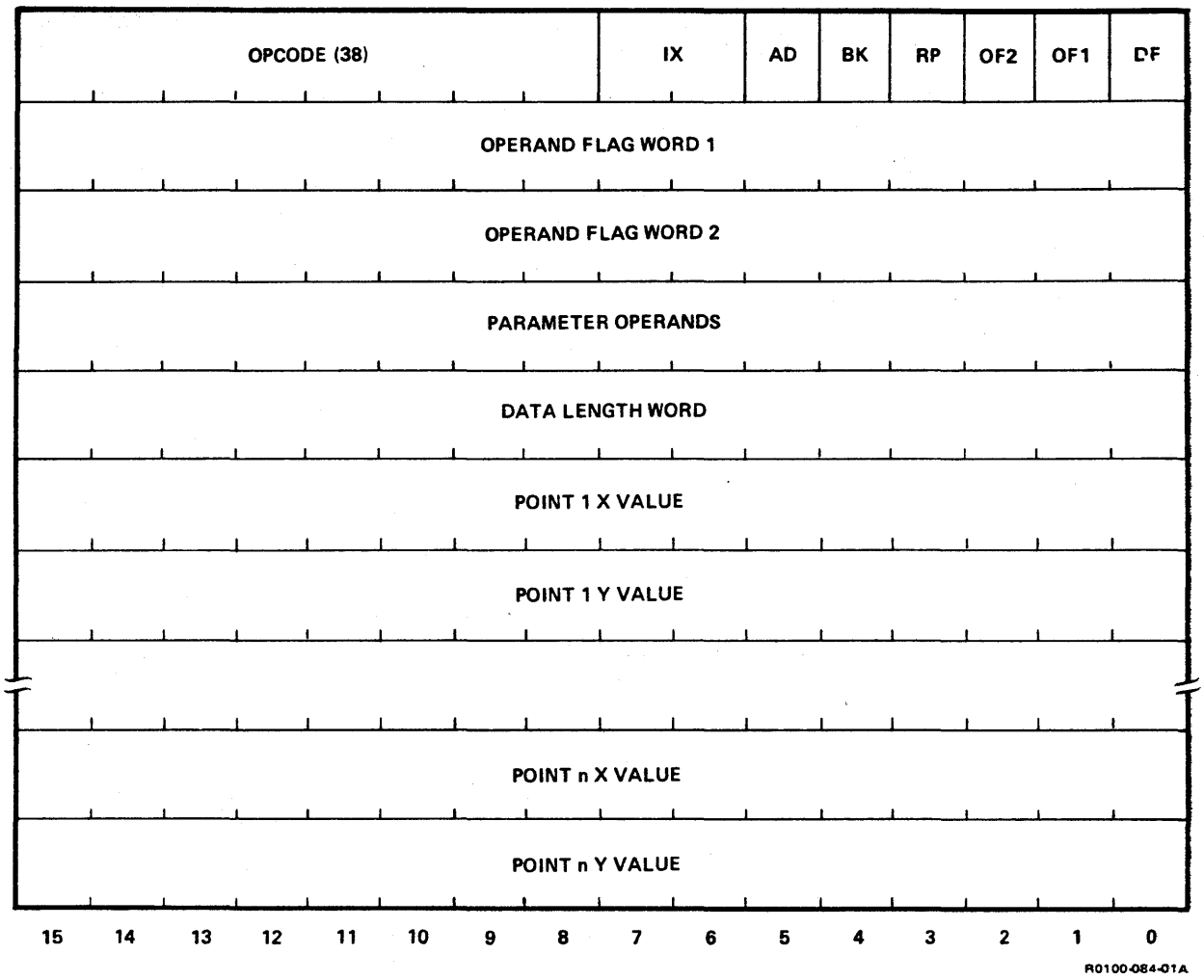
Point-plot data is interpreted on a word-per-point basis. Each word represents the coordinate of a plot-point perpendicular to the plot axis (as evaluated in light of the IX value).

| Errors | Description |
|---|---|
| Code = 3607 | DATA LENGTH WORD does not reflect an even number of bytes. |

| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, START-POINT, and the vector plot data endpoints are evaluated. |
| AD | Effects the generation of vector plot data in conjunction with the BK flag and the VECTOR TEXTURE PATTERN parameter. When AD = 1, writing into refresh memory of zero pattern data bits is inhibited. |
| BK | Defines the color or intensity for each vector write to refresh memory in combination with the VECTOR TEXTURE PATTERN and FOREGROUND or BACKGROUND parameters. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled to receive vector plot data. |
| FOREGROUND | Defines the vector plot data to be written into refresh memory for logical 1 data in the VECTOR TEXTURE PATTERN when BK = 0 and for logical 0 data when BK = 1. |
| BACKGROUND | Defines the vector plot data to be written into refresh memory for logical 0 data in the VECTOR TEXTURE PATTERN when BK = 0 and for logical 1 data when BK = 1. |
| INDEX 1 | Displaces the values to be used for INDEX 2 and START-POINT parameter operands and all plot endpoints when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for the START- POINT parameter operand and all plot endpoints when IX = 10(B). |
| SCAN | Defines the plot orientation. If SCAN $\leq$ 3, the plot axis is the Y-axis (horizontal plot). If SCAN $\geq$ 4, the plot axis is the X-axis (vertical plot). |
| SPACING | Defines the increment to be applied to the COP along the plot axis after each data vector is generated. If SCAN $\leq$ 3, X-SPACING is used. If SCAN $\geq$ 4, Y-SPACING is used. |
| START-POINT | Defines the starting coordinates of the first linked-vector endpoint. |

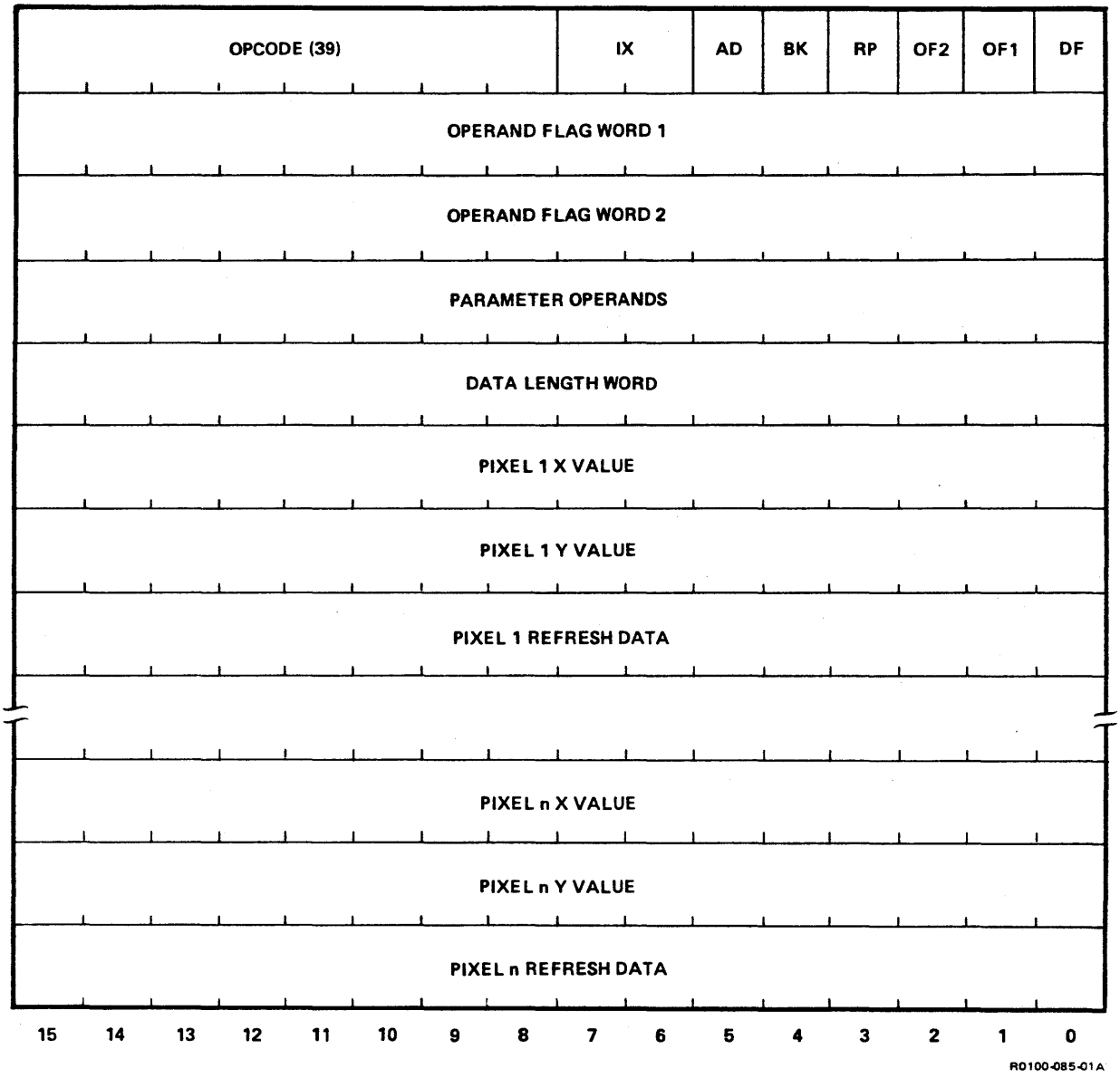
## WRITE POINT (WPT)




WPT is a normal-format instruction which is used to write random pixels with either the FOREGROUND or BACKGROUND parameter operand value based upon the setting of the BK flag. If BK = 0, the FOREGROUND value is written into refresh memory; if BK = 1, the BACKGROUND value is written. The data is used to specify the pixel location and consists of a two-word pair representing the X-Y coordinate of the pixel to be written.

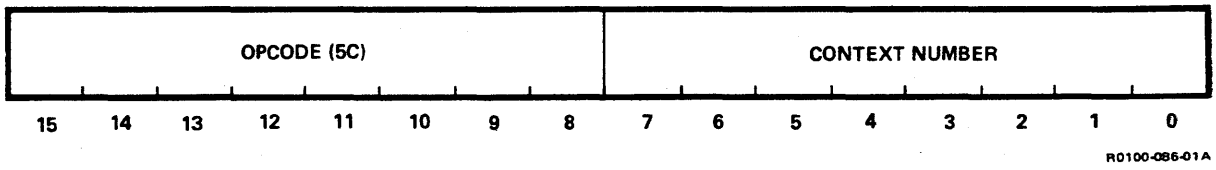
| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, and all pixel coordinate data points are evaluated. |

## WRITE RANDOM PIXEL (WRP)




WRP is a normal-format instruction which is used to write instruction-defined data at random coordinates in global space. The data to be written into refresh memory is supplied in the instruction data stream with each coordinate pair.

## ALLOCATE CONTEXT (ALCON)




ALCON is a special-format instruction which allocates a 4K-byte segment of RAM used to store all parameters that define a display-generation environment. The parameters and variables that are context-dependent are:

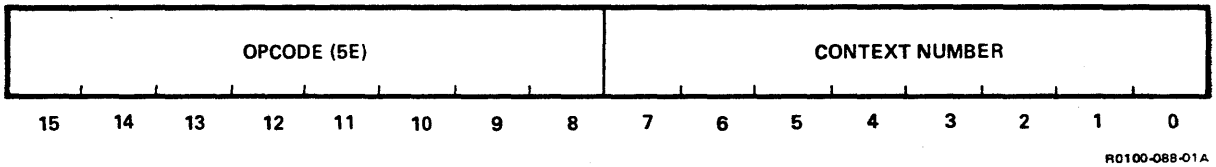
- All normal-format parameter operands
- MCP/group selection masks
- Environment stack used by PUSHE and POPE
- Programmable font number
- Display list number
- Display list registers
- All transformation matrices
- Transformation matrix stack
- Display class ranges

Context 0 is allocated at power-on and is the default display-generation environment. This instruction may not be executed from within a display list.

| Parameters | Description |
|---|---|
| CONTEXT NUMBER | Defines the context to be allocated. It can take on values from 1 to 0F(H). |

| Errors | Description |
|---|---|
| Code = 5C01 | Attempt to execute ALCON from within a display list. |
| Code = 5C02 | Not enough RAM available for allocation of a new context. |
| Code = 5C0E | Attempt to allocate a context already allocated. |
| Code = 5C16 | CONTEXT NUMBER out of range. |

## SELECT CONTEXT (SCON)




SCON is a special-format instruction which defines the context to be used at any given time in the course of processing the RM-9460 display instructions. If an attempt is made to select a context not previously allocated, there is no change of context and an error interrupt is generated. This instruction may not be executed from within a display list.

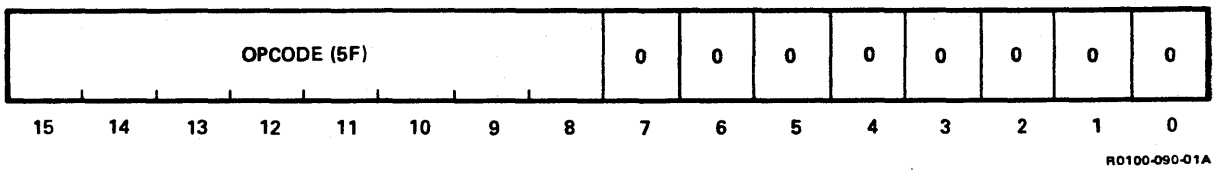
| Parameters | Description |
|---|---|
| CONTEXT NUMBER | Defines the context to be selected. It can take on values from 0 to 0F(H). |

| Errors | Description |
|---|---|
| Code = 5E01 | Attempt to execute SCON from within a display list. |
| Code = 5E05 | Attempt to select a context that has not been allocated. |
| Code = 5E16 | CONTEXT NUMBER out of range. |

**NOTE**

Upon selection of a newly allocated context, the MCP/group select must be set through the use of the SELMG instruction. Failure to do so will result in no display since no destination MCP/group has been selected to receive display data.

## READ ALLOCATION STATE (READAS)




READAS is a special-format instruction which allows the host computer to interrogate the RM-9460 about the current allocation state for contexts, display lists, and programmable fonts. After execution of the READAS, the host must read back ten data words. This instruction may not be executed from within a display list.

### Data Format

The format of the ten words read back by the host is shown in figure 2-14. The first word is a bit map that indicates which of the 16 possible contexts have been allocated. Bit 0 set to 1 indicates context 0 has been allocated, bit 1 represents context 1, etc. The next seven words are the number of 4K-byte segments of RAM allocated to each display list. Three bits are used to represent the number of 4K-byte RAM segments per display list in the following manner:

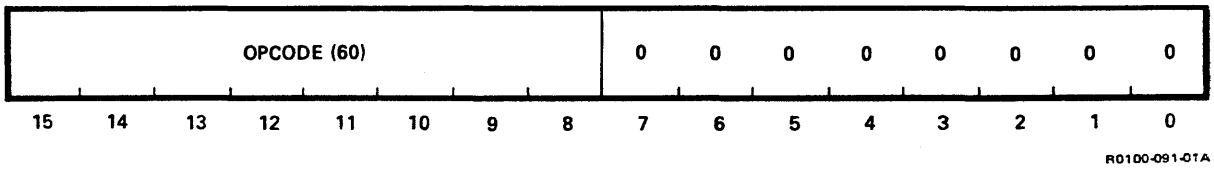
000 0 4K-byte RAM segments
001 1 4K-byte RAM segments
010 2 4K-byte RAM segments
011 3 4K-byte RAM segments
100 4 4K-byte RAM segments

The last word of these seven also indicates which context is currently selected. The ninth and tenth words are the number of 4K-byte RAM segments allocated for each programmable font. Two bits are used to represent the number of 4K-byte RAM segments per programmable font as outlined below:

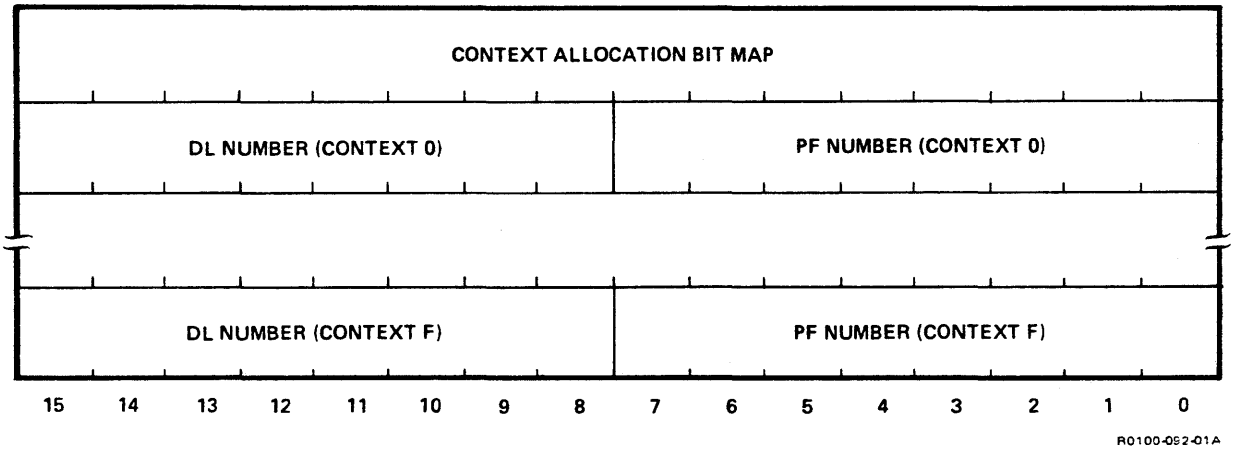
00 0 4K-byte RAM segments
01 1 4K-byte RAM segments
10 2 4K-byte RAM segments

| Errors | Description |
|---|---|
| Code = 5F01 | Attempt to execute READAS from within a display list. |

## READ CONTEXT ASSOCIATIONS (RCON)




RCON is a special-format instruction which allows the host computer to interrogate the RM-9460 about the current association state for contexts; that is, what programmable font and display list are currently attached to each allocated context. After execution of the RCON instruction, the host computer must read back 17 data words. This instruction may not be executed from within a display list.




The total number of words transferred will always be 17 (34 bytes). The format of the readback data is given above. The first word is a bit map that indicates which of the possible contexts has been allocated. Bit 0 set to 1 indicates context 0 has been allocated, bit 1 represents context 1, etc. The words following describe the display list and programmable font currently attached to each context.

For example, if the first word was 0301(H), it indicates that contexts 0, 8, and 9 are currently allocated, and the next 16 words describe the display list and programmable font associated with each context. The associations are read back from low-order context (0) to high-order (15). In all cases readback word 2 is associated with context 0, word 10 is associated with context 8, word 11 is associated with context 9, etc.

| Errors | Description |
|---|---|
| Code = 6001 | Attempt to execute RCON from within a display list. |

| Bit | Description |
|---|---|
| $C_n = 1$ | Indicates that one of the following two events has occurred:<br>1. The ENTER momentary action switch has been depressed with cursor n (where n = 0 through 7) selected on the joystick or trackball.<br>2. The coordinates of cursor n have changed, with cursor n selected on the joystick or trackball and with the TRACK switch enabled. $C_n$ remains set to one until the coordinates of cursor n have been read using the RCSS/RCSP/RCSG/RCSL/RTC/RTCS/PLDLR instruction for condition 6, 7, 8, 9, or A. At that time $C_n$ is reset to zero. |
| $X_n = 0$ | Indicates that a serial transmission initiated by a WKB instruction has not yet completed at output port n (where n = 0 through 7). $X_n$ is set to one when the transmission is completed. |
| $R_n = 1$ | Indicates that a data byte has been received by the serial input port n (where n = 0 through 7). $R_n$ is reset to zero when the data byte, along with its status, has been read back after issuing an RKB instruction. |

| Errors | Description |
|---|---|
| Code = 1A01 | Attempt to execute SPS from within a display list. |

| Parameters | Description |
|---|---|

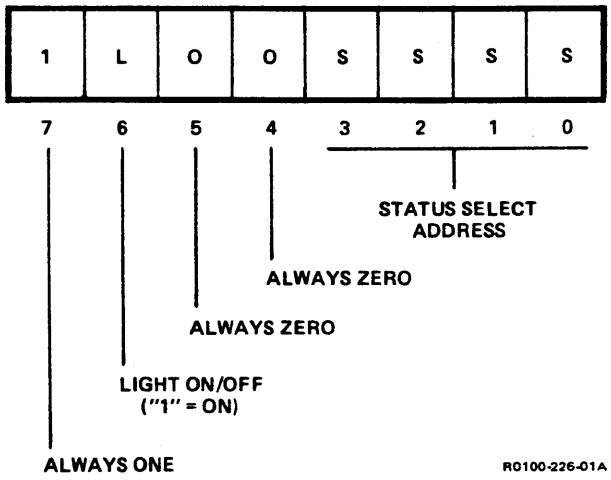
DEVICE

Specifies the serial output device where the CHARACTER CODE is to be output. The DEVICE code assignment is listed below.

| DEVICE | OUTPUT PORT |
|---|---|
| 0 | Serial output port #0 of serial link PCB #0 |
| 1 | Serial output port #1 of serial link PCB #0 |
| 2 | Serial output port #2 of serial link PCB #0 |
| 3 | Serial output port #3 of serial link PCB #0 |
| 4 | Serial output port #0 of serial link PCB #1 |
| 5 | Serial output port #1 of serial link PCB #1 |
| 6 | Serial output port #2 of serial link PCB #1 |
| 7 | Serial output port #3 of serial link PCB #1 |

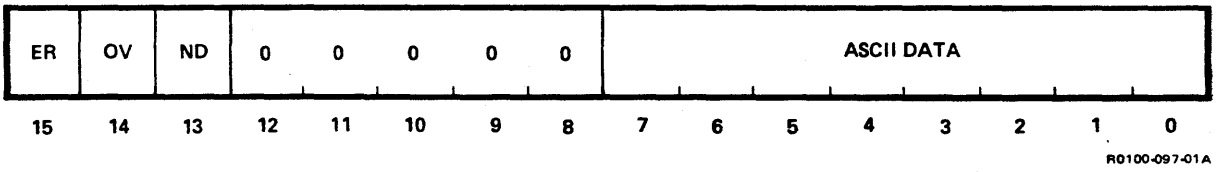
CHARACTER CODE

Defines the 8-bit byte of data that is output to the serial output port specified by DEVICE. All values from 0 through FF(H) are legal. The values 80(H) and above are reserved for setting the state of the keyboard status lights. To set the ON/OFF state of one of the 16 LEDs, set the high-order bit to one, and the remaining bits to the desired values, as described below.



Data Format

The format of the data word read back to the host processor is outlined below.




| Parameters | Description |
|---|---|
| DATA VALUE | Reflects the eight-bit data value input to the RM-9460 display processor from the external serial device. |
| ND | Reflects a no-data-available condition. If ND = 0 and ER = 0, the DATA VALUE represents a valid received data byte. If ND = 1, no data is present in the input buffer. |
| OV | Reflects a data-buffer-overflow condition. If OV = 1, data was received from the serial input port, but no room was available in the input buffer; that is, data has been lost. |
| ER | Reflects a hardware error condition. If ER = 1, a hard error has been detected by the input UART in attempting to obtain a character from the external device. |

| Errors | Description |
|---|---|
| Code = 1901 | Attempt to execute RKB from within a display list. |

| Parameters | Description |
| --- | --- |
| CHARACTERS | Define the eight-bit bytes of data that are output to the serial output port specified by DEVICE. All values from 0 through FF(H) are legal. The order in which the characters are output from each data word is high byte first, low byte second. |

| Parameters | Description |
|---|---|
| CHARACTERS | Define the eight-bit bytes of data that are output to the serial output port specified by DEVICE. All values from 0 through FF(H) are legal. The order in which the characters are output from each data word is low byte first, high byte second. |

| Parameters | Description |
|---|---|
| BL | Defines the blink state of the cursor specified by DEVICE. If BL = 0, the cursor is nonblinking. If BL = 1, the cursor blinks at a rate of one Hz. |

**NOTE**

The cursor's blink and visible states can be controlled manually by a cursor controller if such a device exists in a given configuration. They may be changed manually in this case without notifying the host processor.

| Parameters | Description |
|---|---|
| VI | Defines the visible or invisible state of the cursor specified by DEVICE. If VI = 1, the cursor is made visible. |
| BL | Defines the blink state of the cursor specified by DEVICE. If BL = 0, the cursor is non-blinking. If BL = 1, the cursor blinks at a rate of one Hz. |

**NOTE**

The cursor's blink and visible states can be controlled manually by a cursor controller if such a device exists in a given configuration. They may be changed manually in this case without notifying the host processor.

| Parameters | Description |
|---|---|
| Y-ADDRESS | Defines the line global coordinate where the cursor specified by DEVICE is to be positioned. |
| VI | Defines the visible or invisible state of the cursor specified by DEVICE. If VI = 1, the cursor is made visible. |
| BL | Defines the blink state of the cursor specified by DEVICE. If BL = 0, the cursor is non-blinking. If BL = 1, the cursor blinks at a rate of one Hz. |

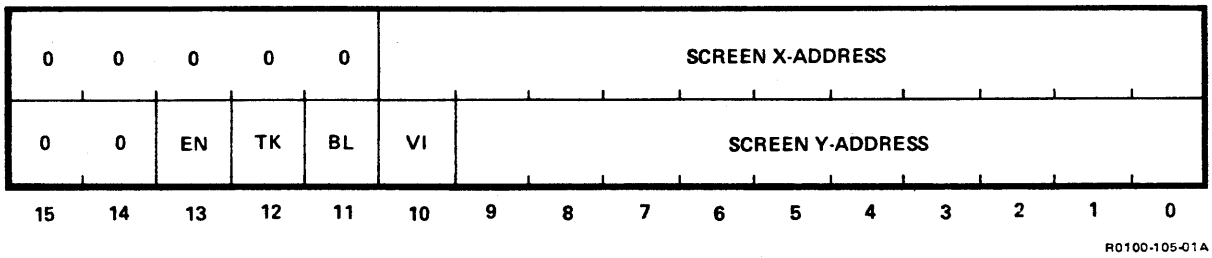
**NOTE**

The cursor's blink and visible states can be controlled manually by a cursor controller if such a device exists in a given configuration. They may be changed manually in this case without notifying the host processor.

| Parameters | Description |
|---|---|
| X-ADDRESS | Defines the element local coordinate where the cursor specified by DEVICE is to be positioned. |
| Y-ADDRESS | Defines the line local coordinate where the cursor specified by DEVICE is to be positioned. |
| VI | Defines the visible or invisible state of the cursor specified by DEVICE. If VI = 1, the cursor is made visible. |
| BL | Defines the blink state of the cursor specified by DEVICE. If BL = 0, the cursor is non-blinking. If BL = 1, the cursor blinks at a rate of one Hz. |

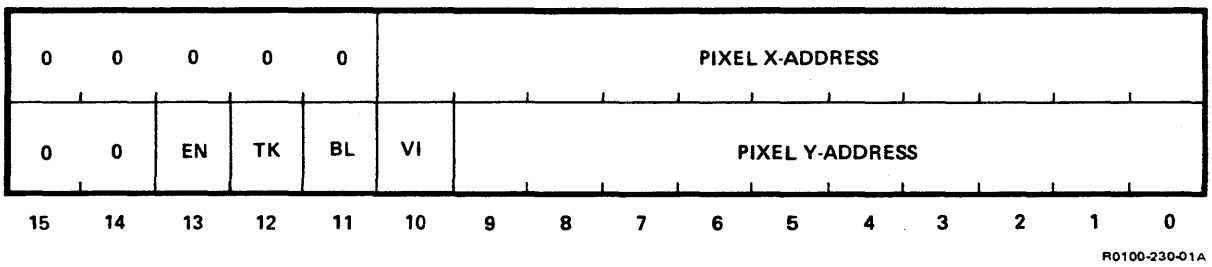
**NOTE**

The cursor's blink and visible states can be controlled manually by a cursor controller if such a device exists in a given configuration. They may be changed manually in this case without notifying the host processor.



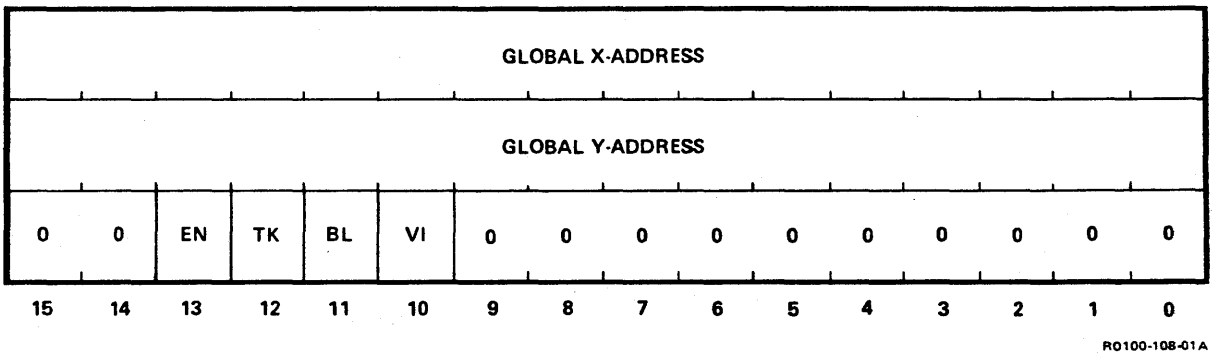

| Parameters | Description |
|---|---|
| X-ADDRESS | Defines the current element screen coordinate of the cursor specified by DEVICE. |
| Y-ADDRESS | Defines the current line screen coordinate of the cursor specified by DEVICE. |
| VI | Defines the current visible/invisible state of the cursor specified by DEVICE. If VI = 0, the cursor is not visible. If VI = 1, the cursor is visible. |
| BL | Defines the blink state of the cursor selected by DEVICE. If BL = 0, the cursor is non-blinking. If BL = 1, the cursor is blinking at a rate of one Hz. |
| TK | Defines the current state of the 2-position TRACK switch on the joystick or trackball if present. If TK = 0, the TRACK function is not enabled. If TK = 1, the TRACK function is enabled. |
| EN | Defines the state of momentary-action ENTER switch on the joystick or trackball if present. If EN = 0, the ENTER switch is not depressed. If EN = 1, the ENTER switch is depressed. |

| Errors | Description |
|---|---|
| Code = 1701 | Attempt to execute RCSS from within a display list. |



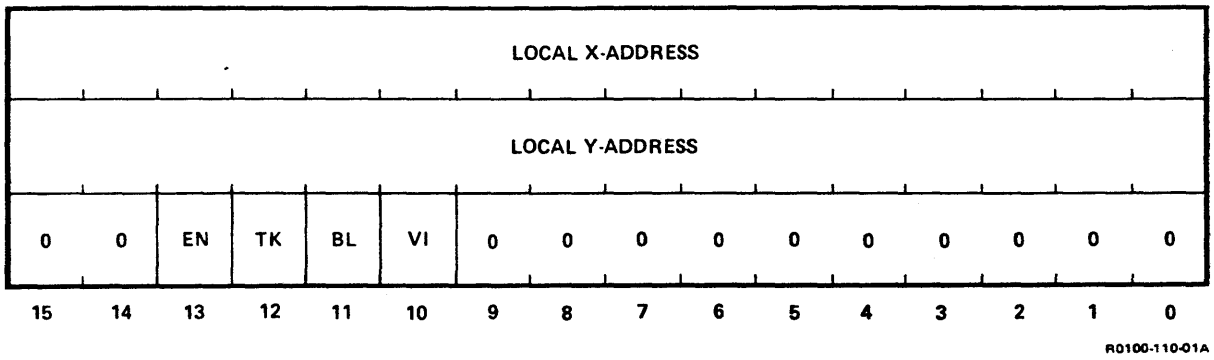

| Parameters | Description |
|---|---|
| X-ADDRESS | Defines the current pixel element coordinate of the cursor specified by DEVICE. |
| Y-ADDRESS | Defines the current line pixel coordinate of the cursor specified by DEVICE. |
| VI | Defines the current visible/invisible state of the cursor specified by DEVICE. If VI = 0, the cursor is not visible. If VI = 1, cursor is visible. |
| BL | Defines the blink state of the cursor selected by DEVICE. If BL = 0, the cursor is non-blinking. If BL = 1, the cursor is blinking at a rate of one Hz. |
| TK | Defines the current state of the 2-position TRACK switch on the joystick or trackball if present. If TK = 0, the TRACK function is not enabled. If TK = 1, the TRACK function is enabled. |
| EN | Defines the state of momentary-action ENTER switch on the joystick or trackball if present. If EN = 0, the ENTER switch is not depressed. If EN = 1, the ENTER switch is depressed. |

| Errors | Description |
|---|---|
| Code = 2E01 | Attempt to execute RCSP from within a display list. |




| Parameters | Description |
|---|---|
| X-ADDRESS | Defines the current element global coordinate of the cursor specified by DEVICE. |
| X-ADDRESS | Defines the current line global coordinate of the cursor specified by DEVICE. |
| VI | Defines the current visible/invisible state of the cursor specified by DEVICE. If VI = 0, the cursor is not visible. If VI = 1, the cursor is visible. |
| BL | Defines the blink state of the cursor selected by DEVICE. If BL = 0, the cursor is non-blinking. If BL = 1, the cursor is blinking at a rate of one Hz. |
| TK | Defines the current state of the 2-position TRACK switch on the joystick or trackball if present. If TK = 0, the TRACK function is not enabled. If TK = 1, the TRACK function is enabled. |
| EN | Defines the state of momentary-action ENTER switch on the joystick or trackball if present. If EN = 0, the ENTER switch is not depressed. If EN = 1, the ENTER switch is depressed. |

| Errors | Description |
|---|---|
| Code = 2F01 | Attempt to execute RCSG from within a display list. |




| Parameters | Description |
|---|---|
| X-ADDRESS | Defines the current element local coordinate of the cursor specified by DEVICE. |
| Y-ADDRESS | Defines the current line local coordinate of the cursor specified by DEVICE. |
| VI | Defines the current visible/invisible state of the cursor specified by DEVICE. If VI = 0, the cursor is not visible. If VI = 1, the cursor is visible. |
| BL | Defines the blink state of the cursor selected by DEVICE. If BL = 0, the cursor is non-blinking. If BL = 1, the cursor is blinking at a rate of one Hz. |
| TK | Defines the current state of the 2-position TRACK switch on the joystick or trackball if present. If TK = 0, the TRACK function is not enabled. If TK = 1, the TRACK function is enabled. |
| EN | Defines the state of momentary-action ENTER switch on the joystick or trackball if present. If EN = 0, the ENTER switch is not depressed. If EN = 1, the ENTER switch is depressed. |

| Errors | Description |
|---|---|
| Code = 8301 | Attempt to execute RCSL from within a display list. |

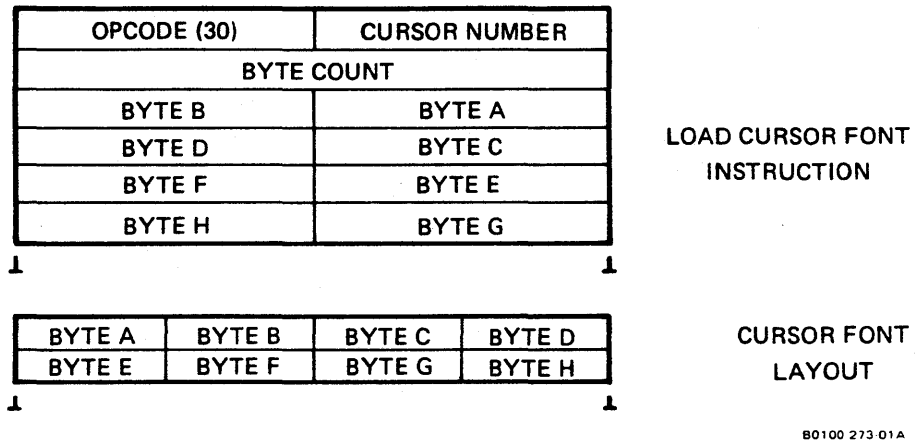
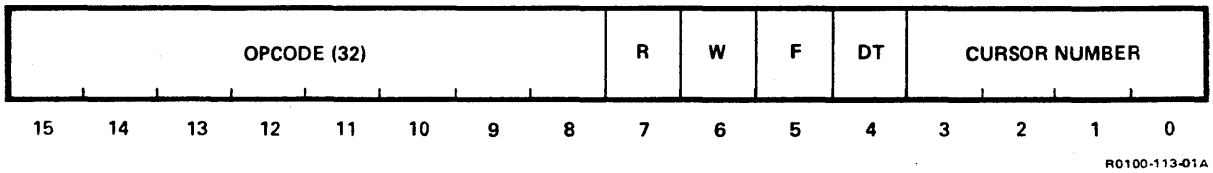



Figure 2-15. Cursor Font Format

## CENTER WINDOW AT CURSOR (CWC)



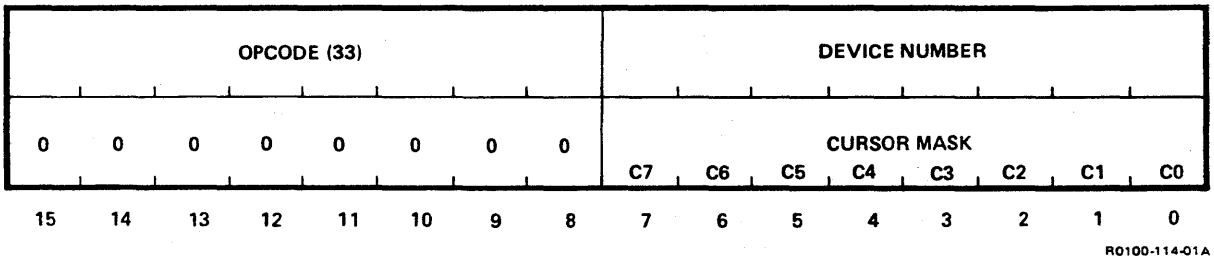

CWC is a special-format instruction which is used to modify the REFRESH WINDOW, WRITE-ENABLE WINDOW, FORMAT WINDOW, or DETECT WINDOW. The selected window is centered at the selected cursor's global coordinates relative to the current context. The size in pixels of the selected window will be maintained.

| Parameters | Description |
|---|---|
| CURSOR NUMBER | Defines the cursor whose global coordinates are to be used (range 0 through 7). |
| R | When set to 1, the REFRESH WINDOW is centered around the cursor. The REFRESH WINDOW is the X-resolution by Y-resolution rectangle of refresh memory and is positioned by changing the WRITE-ENABLE WINDOW values. The WRITE-ENABLE WINDOW will retain the same relative position in refresh memory. The new WRITE-ENABLE WINDOW minimum values are: $$XWEWMIN = XCURSOR + XWEWOFFSET - \frac{XRES}{2}$$ $$YWEWMIN = YCURSOR + YWEWOFFSET - \frac{YRES}{2}$$ |
| W | When set to 1, the WRITE-ENABLE WINDOW is centered around the cursor. This is done by changing the WRITE-ENABLE WINDOW and WRITE-ENABLE WINDOW OFFSETS. The REFRESH WINDOW in global space will remain at the same relative position. |
| F | When set to 1, the FORMAT WINDOW is centered around the cursor. |
| DT | When set to 1, the DETECT WINDOW is centered around the cursor. |

### Errors

None

## DEFINE DEVICE/CURSOR CONFIGURATION (DDCC)




DDCC is a special-format instruction which defines the interactive device (joystick, trackball, etc.) to cursor configuration. It allows any or all of four cursors to be associated with an interactive device. There can be up to eight cursors and eight interactive devices in an RM-9460 system. Due to hardware limitations, cursors 0 through 3 can be associated only with interactive devices 0 through 3, and cursors 4 through 7 with interactive devices 4 through 7. If this rule is violated, an error interrupt is generated.

| Parameters | Description |
|---|---|
| DEVICE NUMBER | Defines the interactive device to be used to update the associated cursors (range 0 through 7). |
| CURSOR MASK | Defines the cursors to be updated by the interactive device defined by DEVICE NUMBER. |

| Errors | Description |
|---|---|
| Code = 3327 | Illegal device/cursor association attempted. |

Examples:

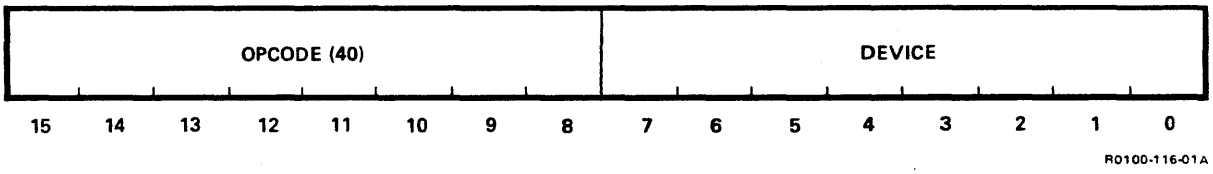
1. The sequence DDCC + 01(H)
   000E(H)

   assigns device 1 to update cursors 1, 2, and 3.

2. The sequence DDCC + 05(H)
   0001(H)

   is illegal since it attempts to assign device 5 to update cursor 0.

## READ TABLET COORDINATES (RTC)




RTC is a special-format instruction which conditions the RM-9460 for the transfer of two words of graphic tablet positional information to the host processor. After the RTC instruction has been transferred from the host processor to the RM-9460, the host processor must initiate a two 16-bit word data transfer from the RM-9460. If these two words are not read back properly by the host processor, the interprocessor communication sequence may halt. This condition can be rectified only by a hard system reset. The values returned are the graphic tablet coordinate over which the positioning device is located.

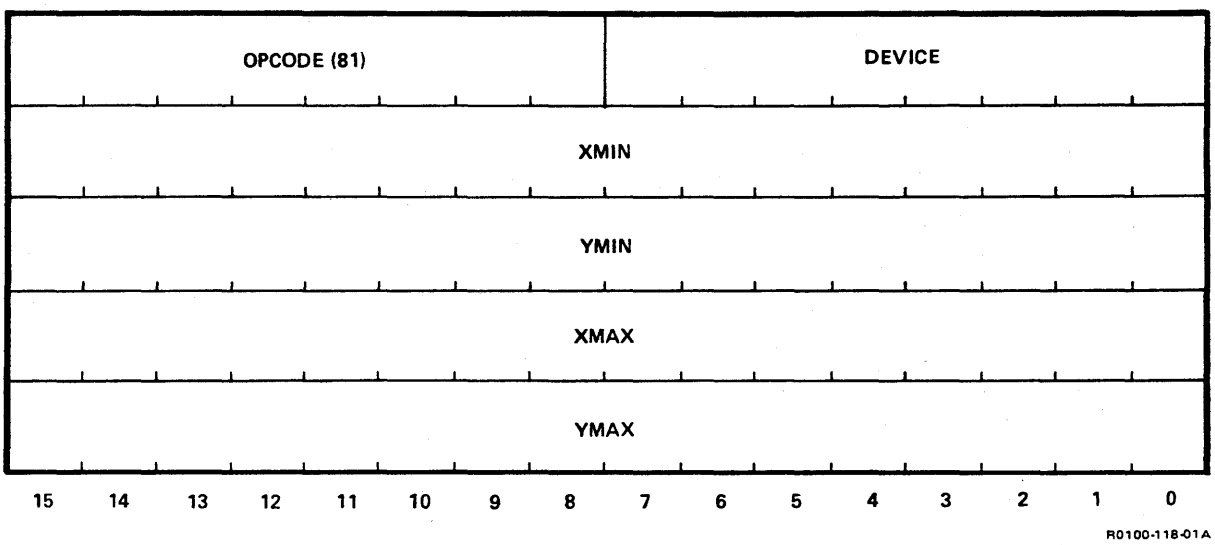
| Parameters | Description |
|---|---|
| DEVICE | Specifies the graphic tablet whose coordinates are to be read back to the host processor after the RTC instruction has been issued. The DEVICE assignments are listed below. |

| DEVICE | GRAPHIC TABLET |
|---|---|
| 0 | Tablet #0 of serial link PCB #0 |
| 1 | Tablet #1 of serial link PCB #0 |
| 2 | Tablet #2 of serial link PCB #0 |
| 3 | Tablet #3 of serial link PCB #0 |
| 4 | Tablet #0 of serial link PCB #1 |
| 5 | Tablet #1 of serial link PCB #1 |
| 6 | Tablet #2 of serial link PCB #1 |
| 7 | Tablet #3 of serial link PCB #1 |

### Data Format

The format of the two data words to be read back to the host processor is shown in the outline that follows.

## SET CURSOR WINDOW (SCW)

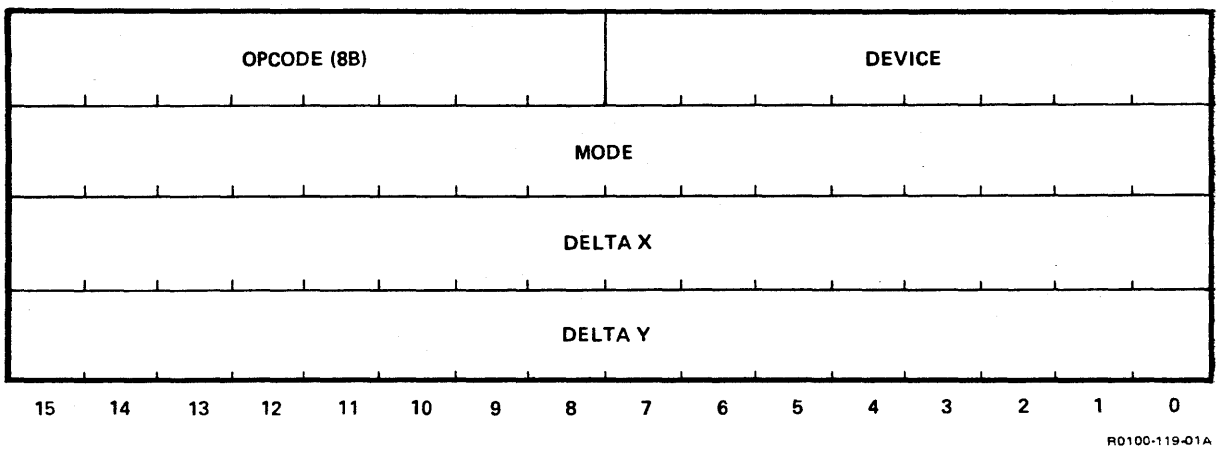



SCW is a special-format instruction which is used in conjunction with the graphic tablet operating in non-menu mode. (See appendix C for a description of the graphic tablet operating modes.) This instruction is used to specify a rectangular area on the tablet surface that will provide full screen cursor window; the associated cursor will be placed at the same relative position on the display screen as the stylus is within the cursor window. When a point is digitized outside the cursor window, no cursor movement will take place. In either case the host is notified and may read either the tablet coordinate, using RTC or RTCS, or the cursor coordinate, using RCS, RCSS, RCSP, RCSG, or RCSL.

| Parameters | Description |
|---|---|
| DEVICE | Specifies the graphic tablet whose active area is to be defined by the instruction. The device assignments are listed below. |

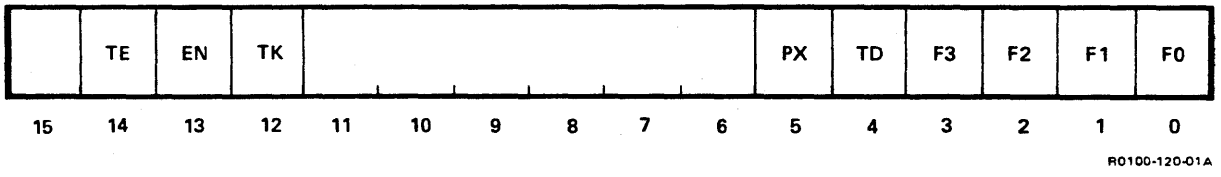
| DEVICE | GRAPHIC TABLET |
|---|---|
| 0 | Tablet #0 of serial link PCB #0 |
| 1 | Tablet #1 of serial link PCB #0 |
| 2 | Tablet #2 of serial link PCB #0 |
| 3 | Tablet #3 of serial link PCB #0 |
| 4 | Tablet #0 of serial link PCB #1 |
| 5 | Tablet #1 of serial link PCB #1 |
| 6 | Tablet #2 of serial link PCB #1 |
| 7 | Tablet #3 of serial link PCB #1 |

## SET TABLET MODE (STM)



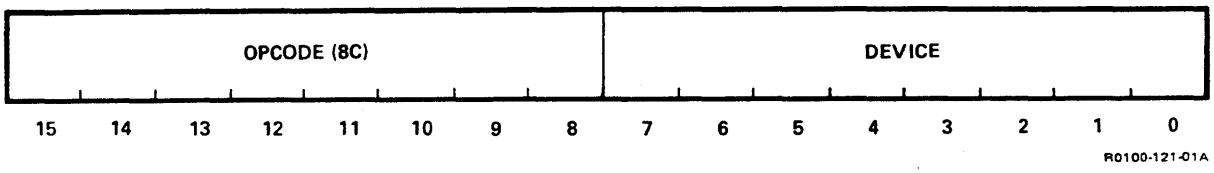

STM is a special-format instruction which sets the graphic tablet working mode and specifies a TRACK distance that can be used to reduce the number of interrupts when digitizing a line drawing.

| Parameters | Description |
|---|---|
| MODE | Defines tablet working mode. |




| | |
|---|---|
| EN | Enter Mode. When set, an interrupt will be generated if any button on the puck is depressed. |
| TE | Trailing Edge Mode. When set, an interrupt will be generated if any button on the puck is released. |
| PX | Proximity Mode. When set, an interrupt will be generated if the PUCK/STYLUS enters or leaves proximity of the tablet. |
| TK | TRACK Mode. When set, an interrupt will be generated if any button is held down and the PUCK/STYLUS is moved. TD ( bit 4) can be used to specify how much the PUCK must be moved to cause a TK interrupt. |

## READ TABLET COORDINATES AND STATUS (RTCS)




RTCS is a special-format instruction which conditions the RM-9460 for the transfer of two words of X,Y coordinates and one word of tablet status information to the host processor. After the RTCS instruction has been transferred from the host processor to the RM-9460, the host must initiate a three 16-bit word data transfer from the RM-9460. If the three words are not read back properly by the host processor, the inter-processor communication sequence may halt. This condition can be rectified only by using a hard system reset. The values returned are the graphic tablet coordinate and tablet status.

| Parameters | Description |
|---|---|
| DEVICE | Specifies the graphic tablet whose coordinates and status are to be read back to the host processor after the RTCS instruction has been issued. The DEVICE assignments are listed below. |

| DEVICE | GRAPHIC TABLET |
|---|---|
| 0 | Tablet #0 of serial link PCB #0 |
| 1 | Tablet #1 of serial link PCB #0 |
| 2 | Tablet #2 of serial link PCB #0 |
| 3 | Tablet #3 of serial link PCB #0 |
| 4 | Tablet #0 of serial link PCB #1 |
| 5 | Tablet #1 of serial link PCB #1 |
| 6 | Tablet #2 of serial link PCB #1 |
| 7 | Tablet #3 of serial link PCB #1 |

### Data Format

The format of the three data words to be read back to the host processor is shown in the outline that follows.

| Parameters | Description |
|---|---|
| PX | When set, the STYLUS/PUCK has left or entered the proximity of the table. If MU or WI is set, then the STYLUS or PUCK has entered the proximity of the tablet; while if MU and WI are reset, then the STYLUS or PUCK has left the proximity of the tablet. |
| TD | When set, an interrupt caused by moving the STYLUS/PUCK out of TD distance has been generated to the host. |
| F3,F2,F1,F0 | When any of these are set, the associated button on the PUCK has been depressed. For PUCKS with only one button or for a STYLUS, depressions will cause F0 to be set. |

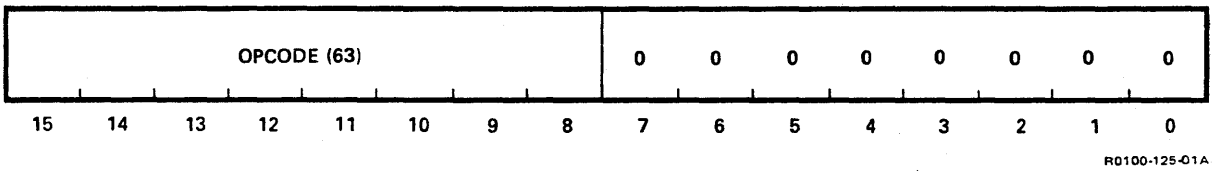
| Errors | Description |
|---|---|
| Code = 8C01 | Attempt to execute RTCS from within a display list. |

| Parameters | Description |
|---|---|
| E | Determines whether or not an early return from the display list execution will occur. If set to 1, after the number of hits have been detected, display list processing will cease. If set to 0, after the number of hits have been detected, display list processing will continue until normal completion of the display list execution. |
| DETECT WINDOW | Defines the rectangular area in global space that will be detect-enabled. |
| NUMBER OF HITS TO IGNORE | Defines the number of detect hits to ignore before recording of detect hits starts. The parameter provides a means of indenting past known hits in an instruction list. |
| DETECT CLASS NUMBER RANGE | Defines the master range of detect class numbers that must be currently selected by the SET DETECT DATA instruction for detect hits to be recorded. This parameter provides a means of partitioning an instruction list and performing detection on smaller parts of the list. The current DISPLAY CLASS parameter and ranges have no effect when in detect mode. |
| NUMBER OF HITS TO RECORD | Defines the maximum number of detect hits that are to be recorded into the detect buffer. Once the limit is reached, all subsequent hits are ignored. The maximum number of hits that can be recorded is 64. |
| MINIMUM DETECT LEVEL | Defines the subroutine call level within a display list at which detailed detect recording is suspended. Hits generated by lower-level subroutines are recorded only as one hit at the specified higher level. |
| DETECT MEMORY PLANE MASK | Defines the memory planes within the DETECT WINDOW that are detect-enabled. When a detect occurs, this value is compared with the current WRITE MASK and, if both have matching ones in any of the 16-bit positions, a valid detect has occurred and will be recorded. |

## Errors

None

## DISABLE DETECT (DD)

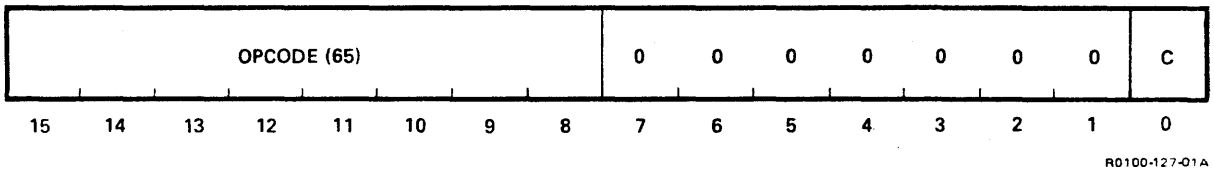



DD is a special-format instruction which switches the RM-9460 out of detect mode. Subsequent data-producing instructions behave normally and produce the normal effects on refresh memory. Detection buffer pointers and counters are unaltered and may be read back using the SENSE DETECT STATUS and READBACK DETECT BUFFER instructions.

Errors

None

## RESUME DETECT (RD)



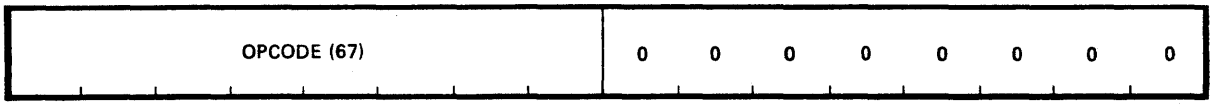

RD is a special-format instruction which reenables hit detection after it has been previously suspended through an SD instruction. The hit detection is resumed without resetting the buffer pointers, counters, etc. Since the RM-9460 is in detect mode, data generated by subsequent instructions is not written into refresh memory.

| Flags | Description |
|---|---|
| C | C is used to disable counting of instructions while in suspend mode. If C = 1, then the RELATIVE INSTRUCTION NUMBER (see READBACK DETECT BUFFER) will not be incremented for this instruction. This will allow host interaction (that is, SUSPEND, interact, RESUME) to be transparent to the detect process. |

Errors

None

## SENSE DETECT STATUS (SDS)



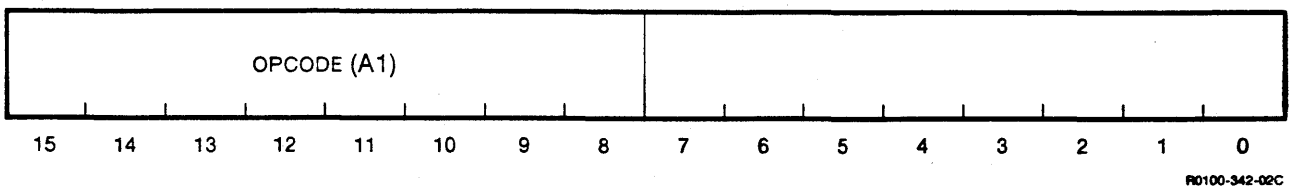

SDS is a special-format instruction which causes the RM-9460 to send to the host computer the number of hits currently in the hit detect list (buffer). This instruction is executed only when transmitted from the host; it cannot be executed from within a display list. If no hits are recorded, zeros are sent to the host computer. If more hits are detected than are allowed to be recorded (the number of hits to record in SDP instruction), the number passed to the host is the two's-complement negative value of the actual number of hits recorded. After issuing this instruction, the host computer should read back one word from the RM-9460. If it does not, a communication halt may result that can be resolved only with a hard system reset.

| Errors | Description |
|---|---|
| Code = 6701 | Attempt to execute SDS from within a display list. |

| Parameters | Description |
|---|---|
| NUMBER OF BYTES TO READ | Defines the number of bytes of detect record information the host computer reads back. It should be noted that each hit record is 12 bytes, so the host computer must read back the number of hits desired times 12 bytes (6 words). If a readback of the specified amount of data does not occur, a communication halt may result that can be resolved only with a hard system reset. |

| Errors | Description |
|---|---|
| Code = 6801 | Attempt to execute RDB from within a display list. |

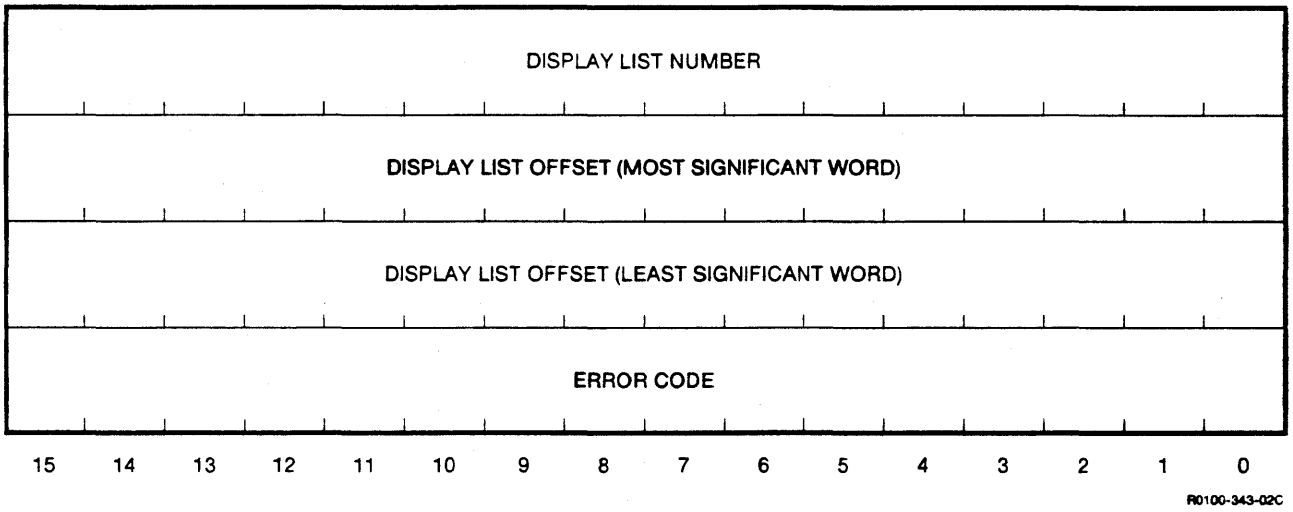
## READ ERRORS (REAERR)




REAERR is a special-format instruction which returns one eight-byte message for each error detected but ignored during display list execution. The SETER instruction should be used to specify which errors to ignore, and the SERRC instruction should be used to determine how many error messages need to be read.

### Data Format

The format of each error message read back is as follows.




| Parameters | Description |
|---|---|
| DISPLAY LIST NUMBER | Indicates the display list where the error occurred. |
| DISPLAY LIST OFFSET | Indicates the display list offset where the error occurred. |
| ERROR CODE | The error code. |

This page not used.

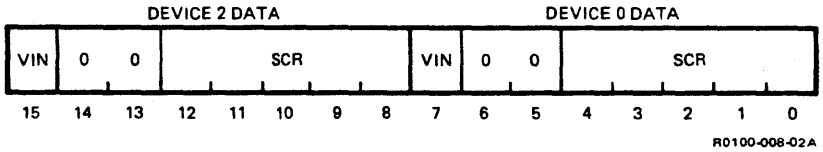
| Parameters | Description |
|---|---|
| R | Returns the status of the PCBs in the system. When R is one, the RM-9460 reads back the status of all four possible video 12 PCBs, whether or not they are installed in the system. The status of each video 12 PCB (device) is returned in a byte. Relative bits zero through four contain the number of full line scrolls that have been performed so far. This scroll count value can range from zero to twenty-four. The visible/invisible status is reflected in relative bit seven. |

Z80 system processor: A zero in relative bit seven indicates that the video 12 PCB is in the same visible/invisible state as after the last reset. A one in relative bit seven signifies that the video 12 PCB is in the opposite state that it had at the last reset. A soft reset will not alter the current visible/invisible state of the board, but it will reset this bit. Further, a hard reset will set the video 12 PCB itself to a visible state, and reset this bit.

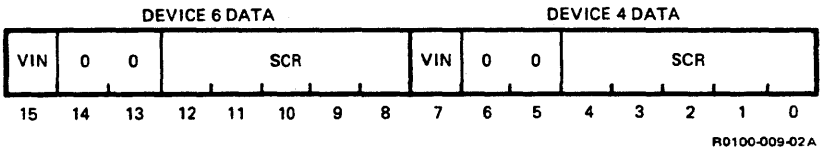
MC68000 system processor: A zero in relative bit seven indicates that the board is invisible. A one in relative bit seven indicates the board is visible.

The format for the two status words is as follows:

Status word 1




Status word 2




where VIN - is the visible/invisible state flag.
SCR - is the count of full line scrolls.

| Parameters | Description |
|---|---|
| LOAD VALUE | Defines the resolution timing constants that the video 12 PCB will use. The LOAD VALUE may range from zero to seven, with the following resolution timing constant definitions: |

| Value | Resolution |
|---|---|
| 0 | 480 x 640 interlaced |
| 1 | 512 x 640 interlaced |
| 2 | 512 x 640 repeat |
| 3 | 1024 x 1280 interlaced |

**NOTE**

Setting the resolution timing constants of the video 12 PCB to other than the standard constants may result in the display of unintelligible information on the CRT monitor.

## Errors

None

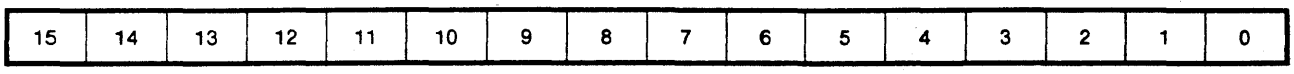
| Errors | Description |
|---|---|
| Code = 8F07 | Odd byte count. |

Example:

The sequence

```
MCPWT + DF
0006(H)                 ;DATA LENGTH WORD (six bytes follow)
0900(H)                 ;MCP INSTRUCTION (load COP)
0200(H)                 ;COP X-value
0100(H)                 ;COP Y-value
```

loads COP in the currently selected MCPs.




X0100-365-02D

Since the Printacolor printer can print only 127 colors, no more than seven planes of image data are read for printing. The first active bit in the mask determines which seven planes are to be used. If the first active bit is bit 0, then planes 0 - 6 will be used; if it is bit 1, planes 1 - 7 will be used. To print an image in planes 2 - 8, the mask should be set to 01FC(H). Less than seven planes of data can be printed by masking off unwanted bits with the mask. To print an image in planes 5, 7, 9, and 11, the mask should be set to 0AA0(H).

X MIN — The minimum X border of the image to be printed (refresh memory coordinates).

Y MIN — The minimum Y border of the image to be printed (refresh memory coordinates).

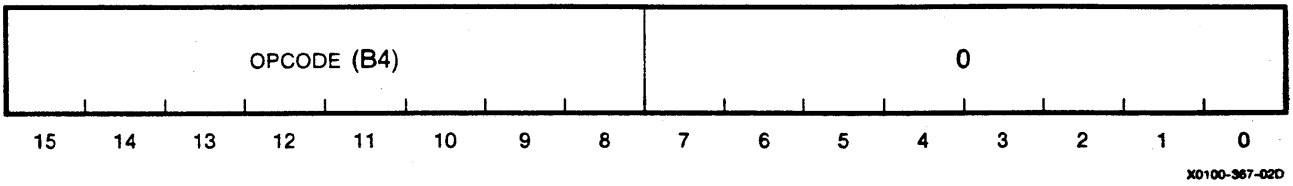
X MAX — The maximum X border of the image to be printed (refresh memory coordinates).

Y MAX — The maximum Y border of the image to be printed (refresh memory coordinates).

## Errors

None.

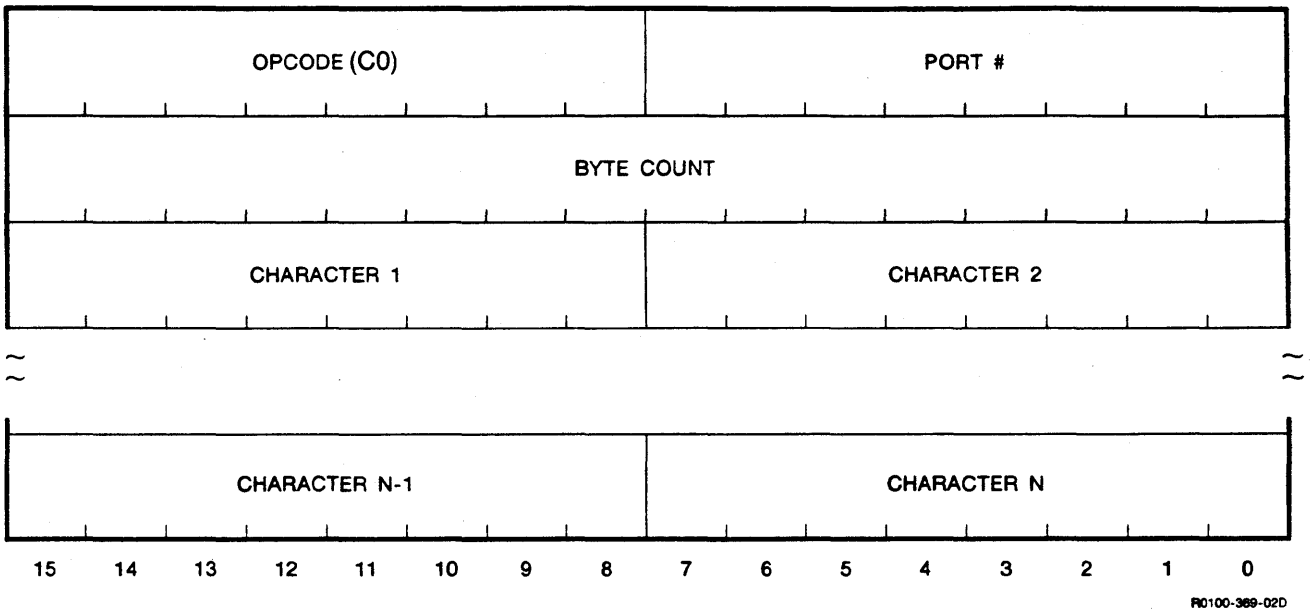
## SENSE PRINTER PROCESS STATUS (SPPS)




SPPS is a special-format instruction which will return one status word containing the number of image lines left to be printed.

| Errors | Description |
|---|---|
| Code = B401 | Attempt to execute SPPS from within a display list. |

## WRITE PROCESSOR BOARD PORT (WPBP)



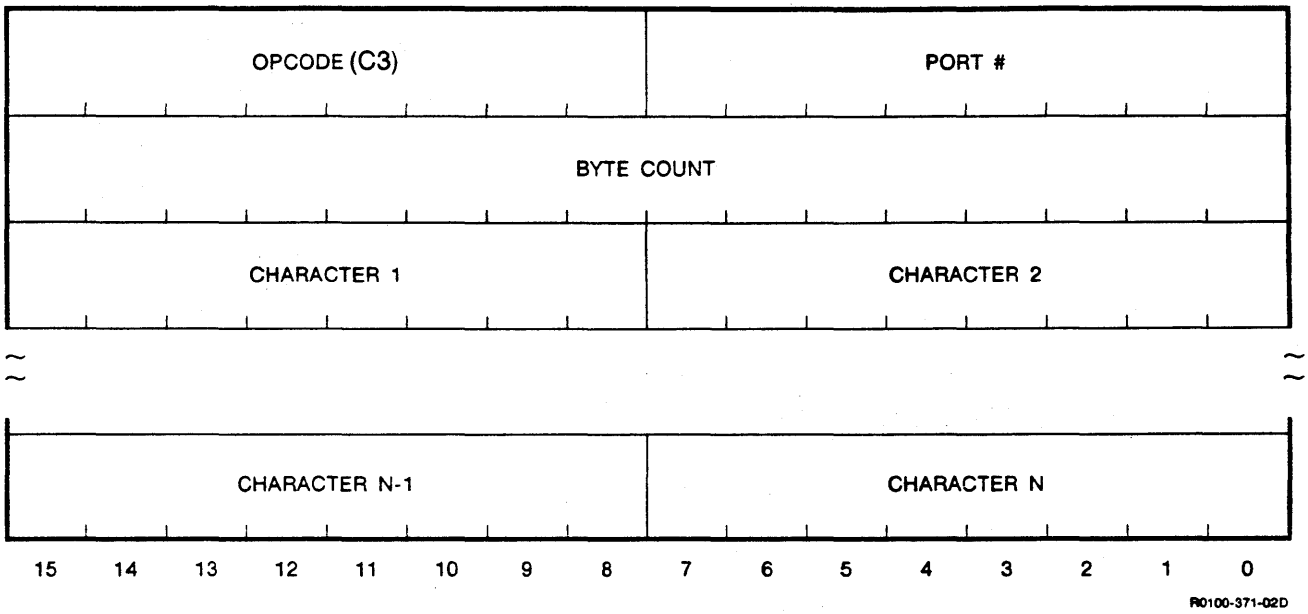

WPBP is a special-format instruction which is used to send characters to a specified serial output port on a MC68000 display processor, usually for the purpose of programming a device.

| Parameters | Description |
|---|---|
| PORT # | Specifies the serial output port where the characters are output. The ports are specified according to numbers from 0 to 3, where 0 is the top and 3 is the bottom port on the processor board. |
| BYTE COUNT | Specifies how many characters are to be output to the specified serial port (one character per byte). |
| CHARACTERS | Specifies the eight bits of data per character that are output to the specified port. The value for each character can range from 0 to FF(H). The order in which the characters are output from each data word is high byte first, low byte second. |

Errors

| | |
|---|---|
| Code = C053 | Illegal port number. |

## WRITE SLC CURSOR PORT (WSLCCP)



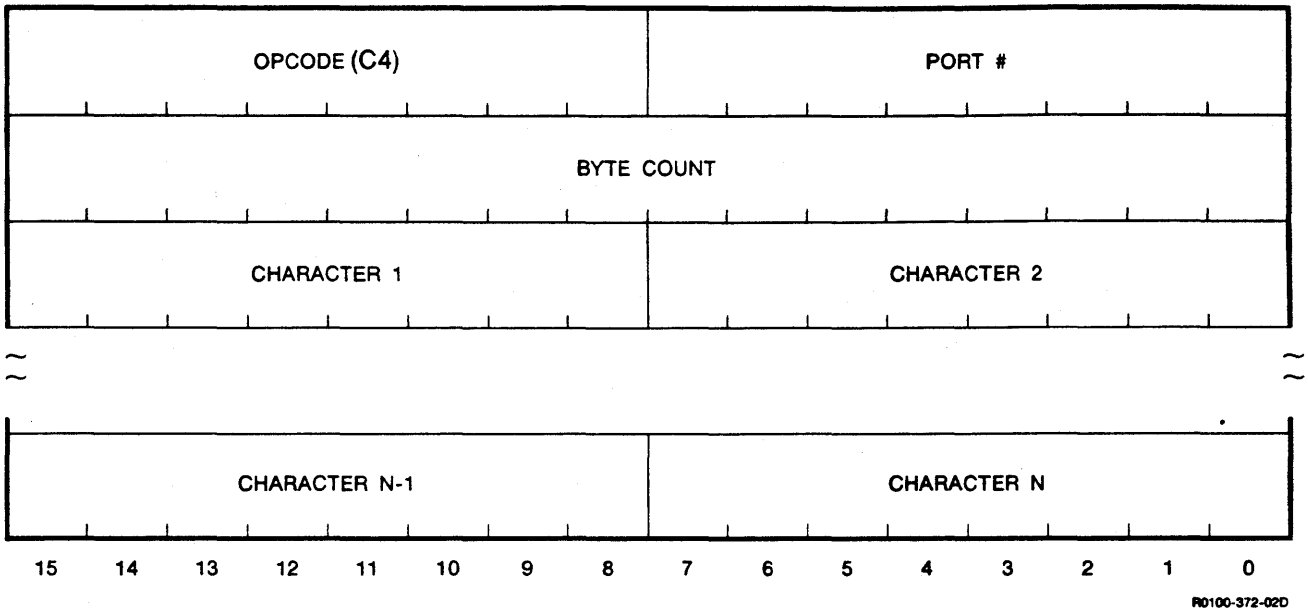

WSLCCP is a special-format instruction which is used to send characters to a specified serial output port on a on a serial link card (SLC), usually for the purpose of programming a device connected to the port.

| Parameters | Description |
|---|---|
| PORT # | Specifies the serial output cursor port where the characters are output. The ports are specified as follows: |

| Specified PORT | Output Port |
|---|---|
| 0 | cursor port #0 of SLC #0 |
| 1 | cursor port #1 of SLC #0 |
| 2 | cursor port #2 of SLC #0 |
| 3 | cursor port #3 of SLC #0 |
| 4 | cursor port #0 of SLC #1 |
| 5 | cursor port #1 of SLC #1 |
| 6 | cursor port #2 of SLC #1 |
| 7 | cursor port #3 of SLC #1 |

Please note that the cursor port number is the same as the cursor controller number, but not the same as the physical port number.

## WRITE SLC CURSOR PORT REVERSE PACKING (WSLCCPRP)



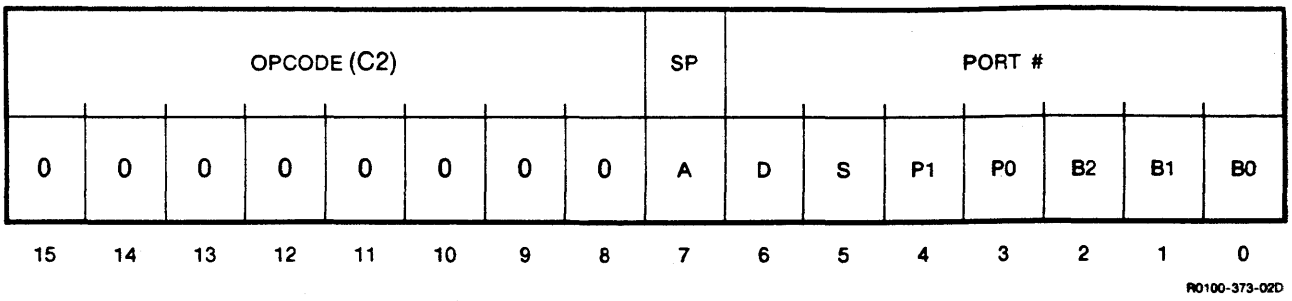

WSLCCPRP is a special-format instruction which is used to send characters to a specified serial output port on a on a serial link card (SLC), usually for the purpose of programming a device connected to the port.

| Parameters | Description |
|---|---|
| PORT # | Specifies the serial output cursor port where the characters are output. The ports are specified as follows: |

| Specified PORT | Output Port |
|---|---|
| 0 | cursor port #0 of SLC #0 |
| 1 | cursor port #1 of SLC #0 |
| 2 | cursor port #2 of SLC #0 |
| 3 | cursor port #3 of SLC #0 |
| 4 | cursor port #0 of SLC #1 |
| 5 | cursor port #1 of SLC #1 |
| 6 | cursor port #2 of SLC #1 |
| 7 | cursor port #3 of SLC #1 |

Please note that the cursor port number is the same as the cursor controller number, but not the same as the physical port number.

## WRITE SERIAL PORT CONFIGURATION (WSPC)




WSPC is a special-format instruction which is used to set up an output port on the display processor or serial link card (SLC). Auto enable, data bits per character, stop bits per character, parity, and baud rate can be programmed.

| Parameters | Description |
|---|---|
| SP | Specifies whether the port to be programmed is on the display processor (SP=1) or the SLC (SP=0). |
| PORT # | Specifies which port on the display processor or SLC card is to be programmed. The ports on the display processor are specified, from top to bottom, as ports 0-3. The ports on the SLC are specified as follows: |

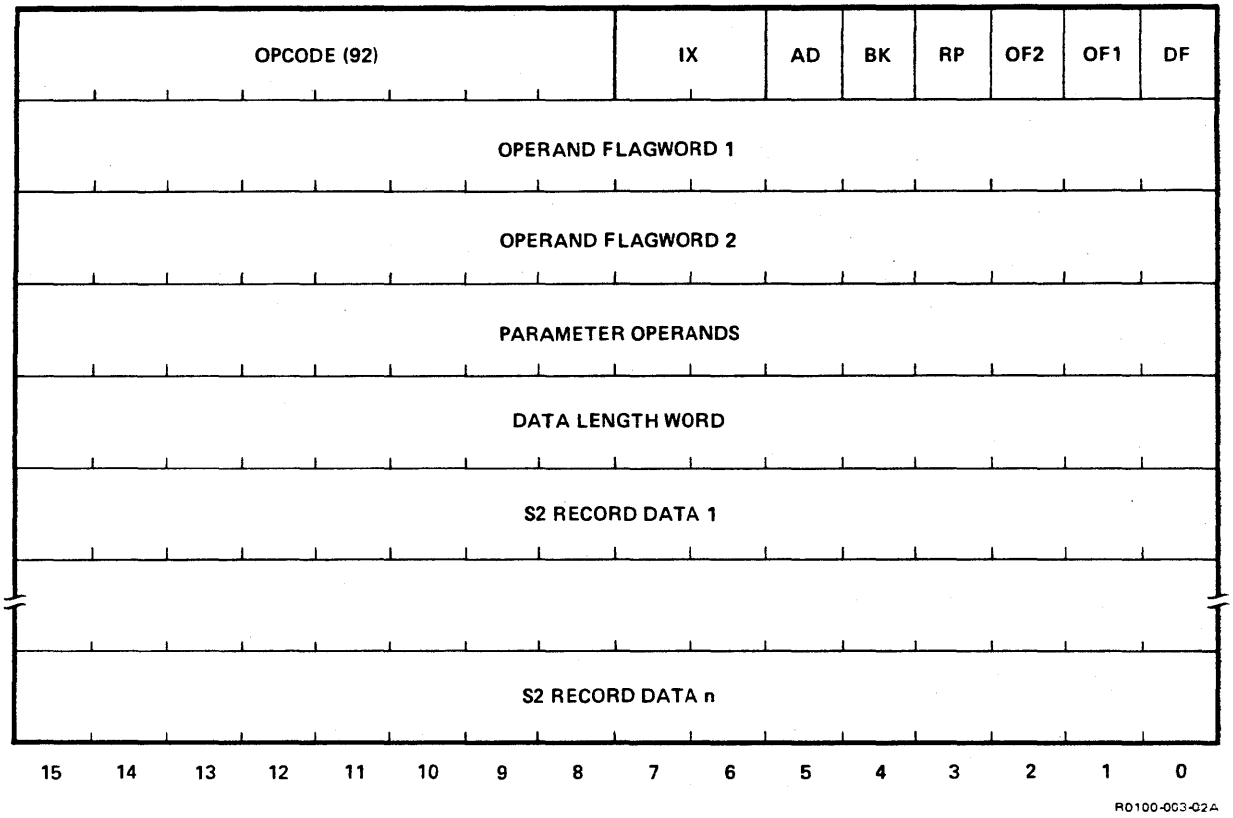
| Specified PORT | Output Port on SLC (from top to bottom) |
|---|---|
| 0 | port #0 of SLC #0 |
| 1 | port #1 of SLC #0 |
| 2 | port #2 of SLC #0 |
| 3 | port #3 of SLC #0 |
| 4 | port #4 of SLC #0 |
| 5 | port #5 of SLC #0 |
| 6 | port #6 of SLC #0 |
| 7 | port #7 of SLC #0 |
| 8 | port #0 of SLC #1 |
| 9 | port #1 of SLC #1 |
| 10 | port #2 of SLC #1 |
| 11 | port #3 of SLC #1 |
| 12 | port #4 of SLC #1 |
| 13 | port #5 of SLC #1 |
| 14 | port #6 of SLC #1 |
| 15 | port #7 of SLC #1 |

This page not used.

Example:

The instruction 9020 reads back the first 32 bytes [20(H)] from the internal configuration buffer.

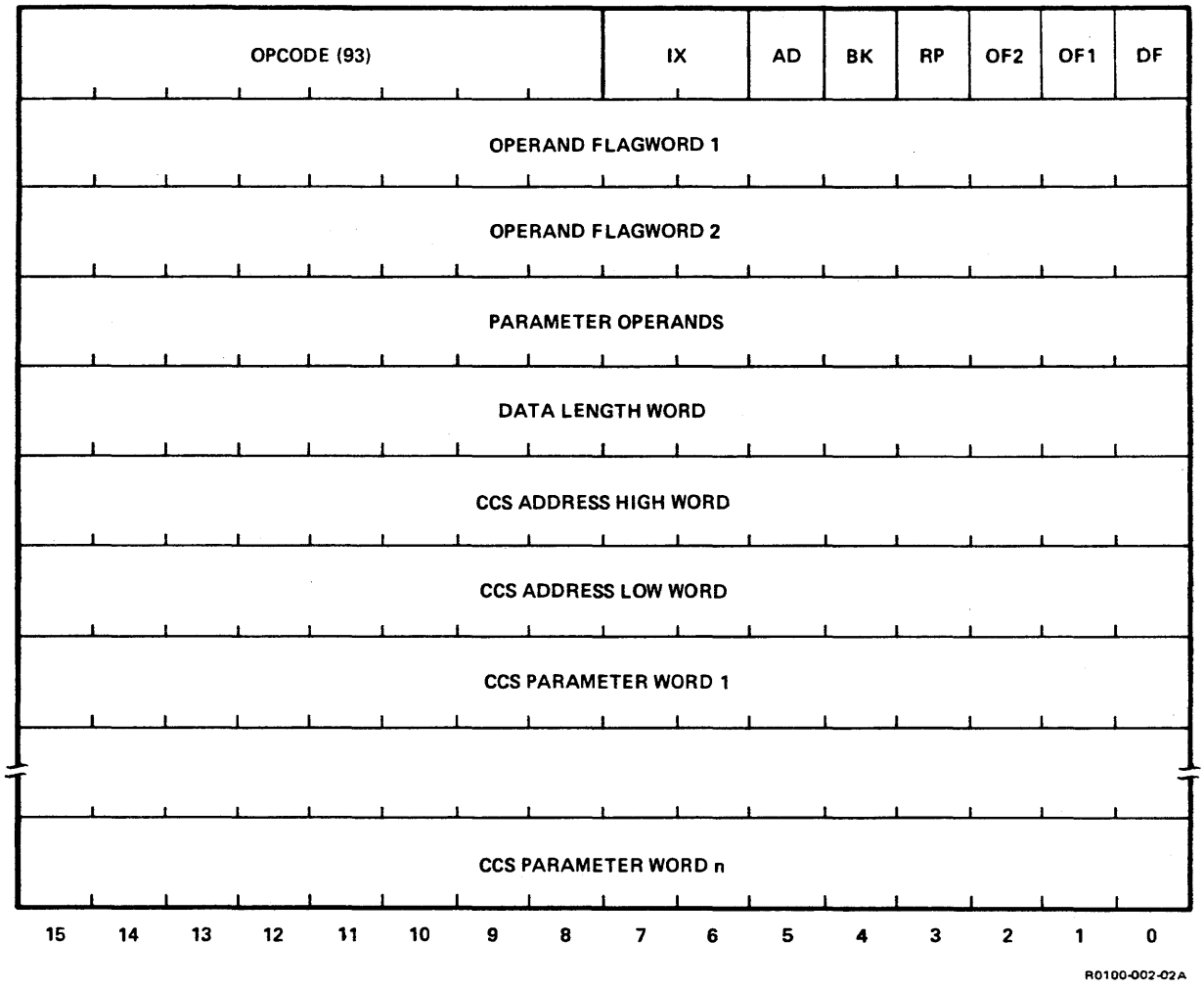
## LOAD CONTROL STORE (LCS) (MC68000 System Processor only)




LCS is a normal-format instruction which loads the control store area in RAM with data representing Motorola S2-records. The data is sent to the RM-9460 as ASCII characters. An unlimited number of S2-records can be loaded with one execution of LCS, provided a maximum of 65534 ASCII bytes of data are sent. Sufficient control storage should be allocated with an ACS instruction before LCS is executed.

| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, FORMAT WINDOW, BASELINE, and START-POINT are evaluated. |
| RP | Defines the byte-accessing order for ASCII characters within each 16-bit data word from the host. If RP = 0, the manner by which characters are accessed is high byte first, then low byte. If RP = 1, the manner by which characters are accessed is low byte first, then high byte. |

## CALL CONTROL STORE (CCS) (MC68000 System Processor only)

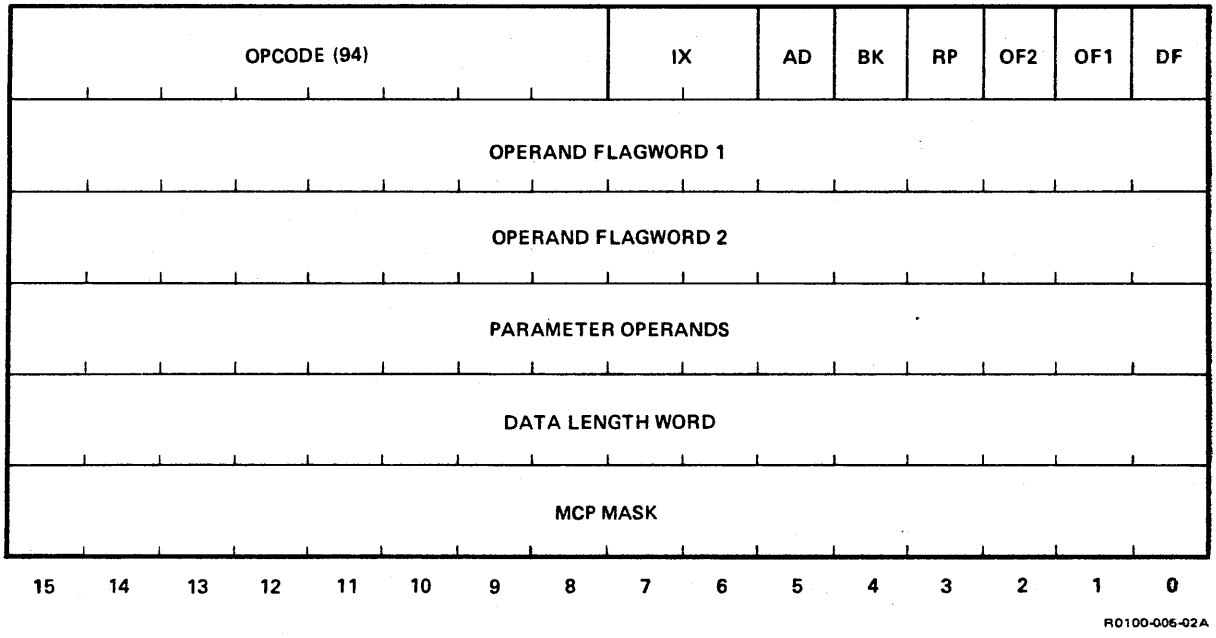



CCS is a normal-format instruction which starts the execution of a user-supplied control store routine. Starting address and parameters, if any, are specified in the data portion of the CCS instruction. The user's control store routine is responsible for reading all existing parameter data.

The control store routine must be allocated and loaded prior to execution of the CCS instruction. Control store routines are allocated with the ALLOCATE CONTROL STORE (ACS) instruction and loaded with the LOAD CONTROL STORE (LCS) instruction.

| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, FORMAT WINDOW, BASELINE, and START-POINT are evaluated. |

## RESET MCPs (RSM) (MC68000 System Processor only)




RSM is a normal-format instruction which resets the specified MCPs. DMA transfer from the host processor to the specified MCP(s) is stopped.

| Flags | Description |
|---|---|
| IX | Defines the address mode where INDEX 1, INDEX 2, FORMAT WINDOW, BASELINE, and START-POINT are evaluated. |

| Parameters | Description |
|---|---|

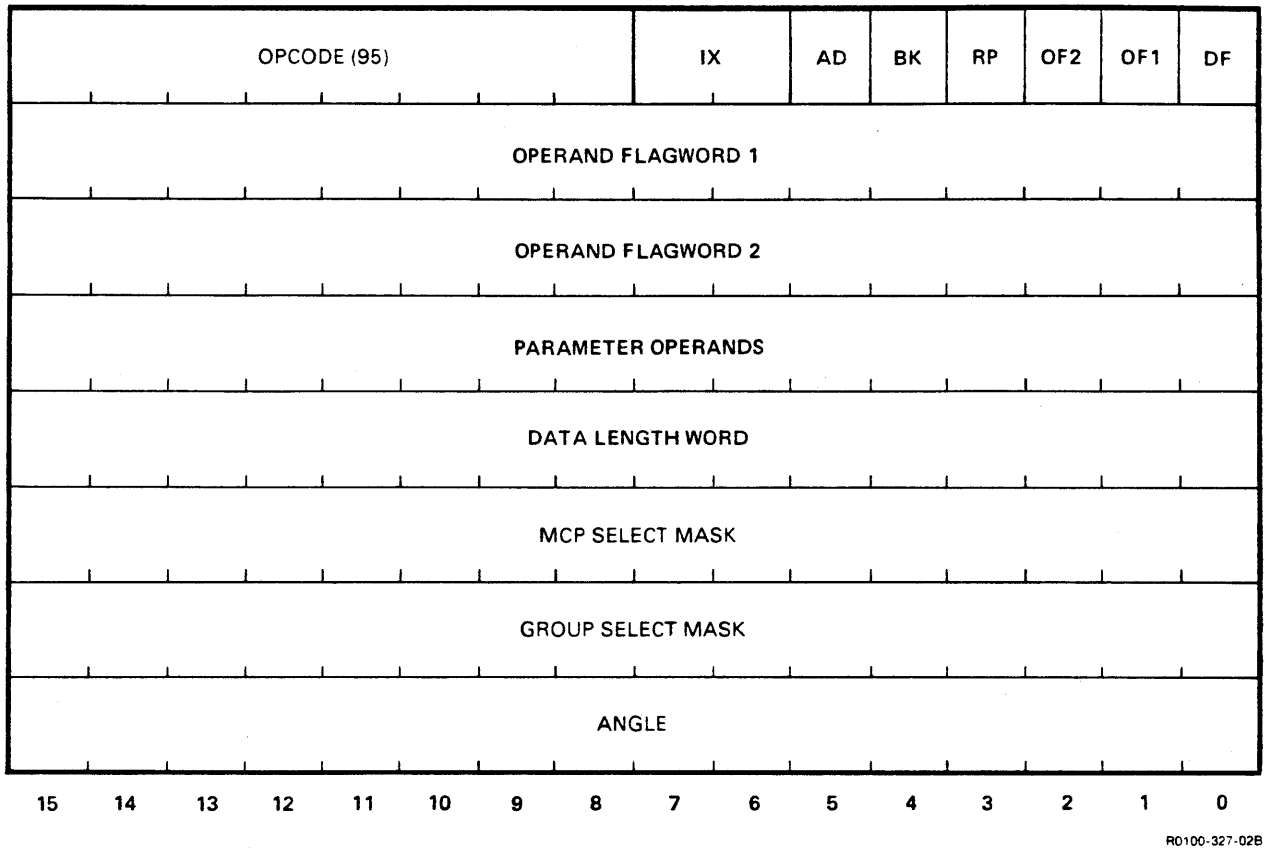
All normal-format parameters apply to the RSM instruction in that the parameters may be redefined.

| | |
|---|---|
| DATA LENGTH WORD | Defines the number of bytes in the MCP MASK word that follows. This value must be set to 2. |

### Data Format

The data word consists of an MCP mask specifying which MCPs to reset. If bit 0 is ON, MCP 0 is reset; bit 1 resets MCP 1, etc.

## COPY IMAGE AND ROTATE (COPYIR) (MC68000 System Processor only)




COPYIR is a normal-format instruction that copies and rotates image data from a source MCP/group to a destination MCP/group. Most LAF functions can be performed during copy and rotate for angles of rotation that are multiples of 90°. LAF does not apply for angles of rotation that are not multiples of 90°. The image copy and rotation is performed in two operations. First COPYIR reads a line of image data from the refresh memory belonging to the source MCP/group and stores the line in MC68000 RAM. Then COPYIR writes the data into the refresh memory belonging to the destination MCP/group, controlled by the LAF parameter.

The current MCP/group is the destination, and the source MCP/group is specified in the instruction. The FORMAT WINDOW is the source area, and the current operating point (COP) is the starting location where the data will be written. In LAF replacement mode, multiple MCPs and groups as destinations are allowed. In all other cases of LAF operations, only a single MCP/group as destination is permitted.

| Flags | Description |
|---|---|

START-POINT (contd.)

| Angle | START-POINT |
|---|---|
| 0 | Upper left |
| 90 | Lower left |
| 180 | Lower right |
| 270 | Upper right |

WRITE-ENABLE WINDOW — Defines the area of global space where data may be written.

W.E.W. OFFSET — Defines the area of refresh memory that overlays global space where data may be written.

DISPLAY CLASS — Defines the current DCL value. If this value is in one of the defined display-class ranges, then writing to refresh memory is enabled; otherwise, writing to refresh memory is disabled.

IMAGE MODE — Defines the reading and writing mode used to copy image data. If FUNCTION is specified and the angle of rotation is a multiple of 90°, the image processing function is performed with the whole word of the current destination memory for all image modes.

Image Mode 0 : refresh memory plane = WMSK .AND. FFFF
Image Mode 1 : refresh memory plane = WMSK .AND. 00FF
Image Mode 2 : refresh memory plane = WMSK .AND. FF00

DATA LENGTH WORD — Defines number of bytes for MCP SELECT MASK, GROUP SELECT MASK, and ANGLE. The value is 6.

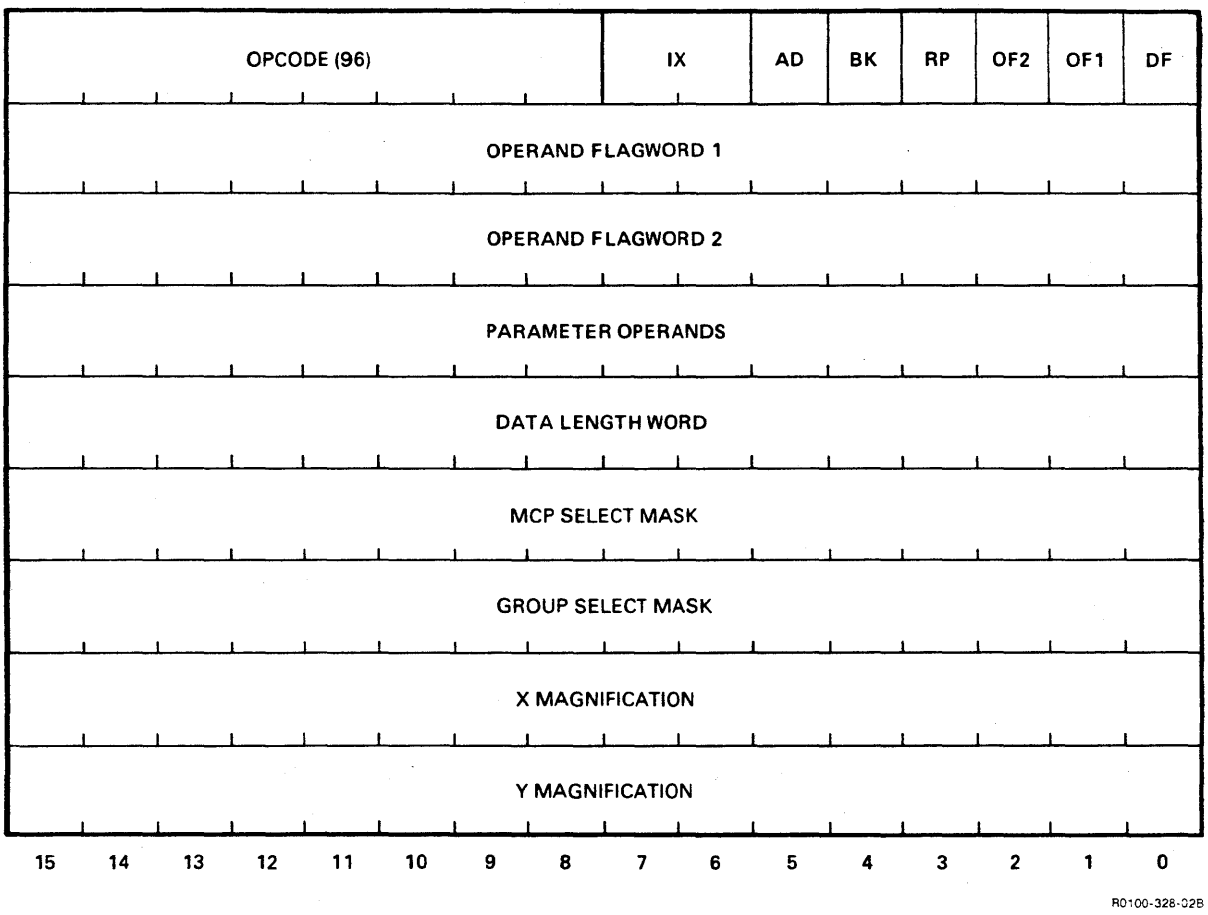
Data Format

The word following the DATA LENGTH WORD is the MCP SELECT MASK value for the source MCP. The bit set to 1 in the MCP SELECT MASK selects the corresponding MCP as the source MCP; for example, bit 0 set to 1 selects MCP 0, bit 1 set to 1 selects MCP 1, etc.

The word following the MCP SELECT MASK is the GROUP SELECT MASK for the source group associated with the selected source MCP. The bit set to 1 in the GROUP SELECT MASK selects the corresponding group as the source group.

The word following the GROUP SELECT MASK is the ANGLE of rotation. A positive ANGLE value rotates the copied image counterclockwise, while a negative ANGLE value rotates the copied image clockwise.

## COPY IMAGE AND MAGNIFY (COPYIM) (MC68000 System Processor only)




COPYIM is a normal-format instruction that copies and magnifies image data from a source MCP/group to a destination MCP/group. The image copy and magnification is performed in two operations. First COPYIM reads a line of image data from the refresh memory belonging to the source MCP/group and stores the line in MC68000 RAM. Then COPYIM writes the data into the refresh memory group belonging to the destination MCP/group.

The current MCP/group is the destination, and the source MCP/group is specified in the instruction. The FORMAT WINDOW is the source area and the current operating point (COP) is the starting location where the data will be written.

| Flags | Description |
|---|---|
| IX | Defines the address mode in which INDEX 1, INDEX 2, FORMAT WINDOW, and START-POINT are evaluated. |

| Parameters | Description |
|---|---|
| IMAGE MODE | Defines the reading and writing mode used to copy image data.<br><br>Image Mode 0 : refresh memory plane = WMSK .AND. FFFF<br>Image Mode 1 : refresh memory plane = WMSK .AND. 00FF<br>Image Mode 2 : refresh memory plane = WMSK .AND. FF00 |
| DATA LENGTH WORD | Defines the number of bytes for MCP SELECT MASK, GROUP SELECT MASK, X MAGNIFICATION, and Y MAGNIFICATION. The value is 8. |

## Data Format

The word following the DATA LENGTH WORD is the MCP SELECT MASK value for the source MCP. The bit set to 1 in the MCP SELECT MASK selects the corresponding MCP as the source MCP; for example, bit 0 set to 1 selects MCP 0, bit 1 set to 1 selects MCP 1, etc.

The word following the MCP SELECT MASK is the GROUP SELECT MASK for the source group associated with the selected source MCP. The bit set to 1 in the GROUP SELECT MASK selects the corresponding group as the source group.

The words following the GROUP SELECT MASK are the X MAGNIFICATION and Y MAGNIFICATION. The magnification value can range from 1 through 16.

## COP Movement

After completion of the COPYIM instruction, the COP is the starting point in the FORMAT WINDOW, determined by SCAN and video orientation.

| Errors | Description |
|---|---|
| Code = 9607 | Odd byte count. |
| Code = 9617 | No MCPs selected to read from. |
| Code = 9618 | Too many MCPs selected to read from. |
| Code = 9619 | No group selected to read from. |
| Code = 961A | Too many groups selected to read from. |
| Code = 964A | Bad magnification range (1-16 allowed). |

WQT is a special-format instruction that writes blocks of text. Text size is fixed and is written left to right. A block consists of (X,Y) start address, color, number of characters, and N characters (two per word). Acceptable character values are 20(H) through 5F(H).

If your firmware is release 10 or higher, the instruction format will differ slightly. The lower byte of the first word should contain zeros, and a second word containing the NUMBER OF BLOCKS should be inserted between the first word and BLOCK 1 X-ADDRESS.

**NOTE**

WQT performs no error checking, so submit the data with care.

| Parameters | Description |
|---|---|
| NUMBER OF BLOCKS | The number of text string blocks included with this instruction. |
| BLOCK i X-ADDRESS | The starting X screen coordinate for text string block i. |
| BLOCK i Y-ADDRESS | The starting Y screen coordinate for text string block i. |
| BLOCK i COLOR | The VLT address containing the desired color for text string block i. |
| BLOCK i NUMBER OF CHARACTERS | The count of the number of characters in text string block i. |

## Data Format

Submit the text string data two characters per word, one character per byte. In the first word, the first character is in the low-order byte, and the second character is in the high-order byte. In the second word, the third character of the text string is in the low-order byte, and the fourth character is in the high-order byte.

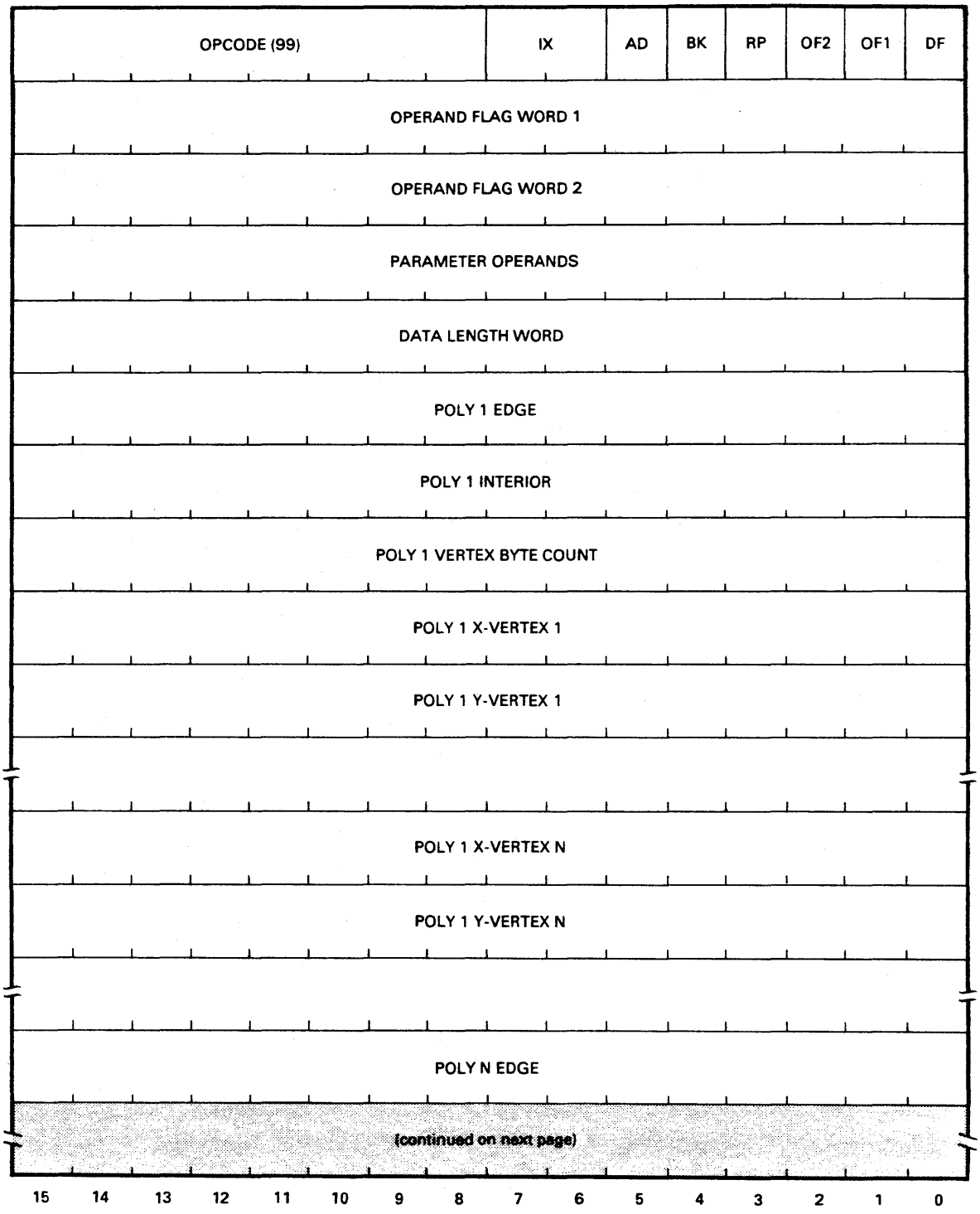
## Errors

None

| Flags | Description |
|---|---|
| BK | Defines the polygon edge and interior values written into refresh memory. If BK = 0, the polygon edge is drawn in the current foreground color, and the polygon interior is filled with the current background color. If BK = 1, the polygon edge is drawn in the current background color, and the polygon interior is filled with the current foreground color. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes enabled for writing data. |
| FOREGROUND | Defines the polygon edge color/intensity data value when BK = 0, and the polygon interior fill color/intensity data value when BK = 1. |
| BACKGROUND | Defines the polygon interior fill color/intensity data value when BK = 0, and the polygon edge color/intensity data value when BK = 1. |
| INDEX 1 | Displaces the values to be used for the INDEX 2 parameter operand and all vertices when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for all vertices when IX = 10(B). |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DATA LENGTH WORD | Defines the number of bytes of vertex data that follows. The value is the number of points in the polygon times 4. |
| X-VERTEX 1 | The 2's complement 16-bit value of the X screen coordinate for the first point in the polygon. |
| Y-VERTEX 1 | The 2's complement 16-bit value of the Y screen coordinate for the first point in the polygon. |
| X-VERTEX N | The 2's complement 16-bit value of the X screen coordinate for the last point in the polygon. |
| Y-VERTEX N | The 2's complement 16-bit value of the Y screen coordinate for the last point in the polygon. |

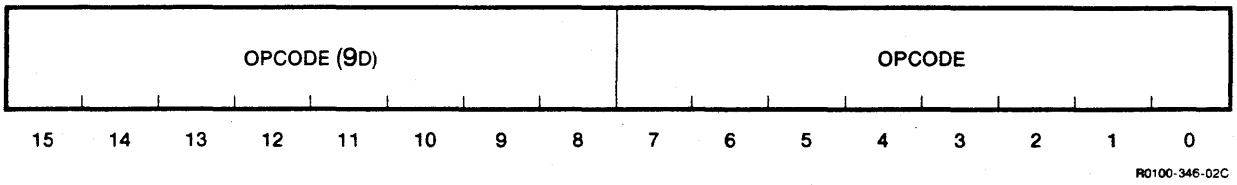
## FILL COLORED POLYGON (FCPOLY) (MC68000 System Processor only)



| Parameters | Description |
|---|---|
| INDEX 1 | Displaces the values to be used for the INDEX 2 parameter operand and all vertices when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for all vertices when IX = 10(B). |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DATA LENGTH WORD | Defines the number of bytes of polygon data that follows. The value is 6 times the number of polygons plus the sum of the VERTEX BYTE COUNT values for each polygon. |
| POLY i EDGE | The VLT address containing the desired color for the edge of polygon i. |
| POLY i INTERIOR | The VLT address containing the desired color for the interior of polygon i. |
| POLY i VERTEX BYTE COUNT | Defines the number of bytes of vertex data for polygon i. The value is the number of points in polygon i times 4. |
| POLY i X-VERTEX 1 | The 2's complement 16-bit value of the X screen coordinate for the first point in polygon i. |
| POLY i Y-VERTEX 1 | The 2's complement 16-bit value of the Y screen coordinate for the first point in polygon i. |
| POLY i X-VERTEX N | The 2's complement 16-bit value of the X screen coordinate for the last point in polygon i. |
| POLY i Y-VERTEX N | The 2's complement 16-bit value of the Y screen coordinate for the last point in polygon i. |

| Errors | Description |
|---|---|
| Code = 994B | Less than three vertices specified for polygon. |

## DELETE OPCODE (DOP)




The DOP instruction is a special-format instruction. It allows the user to delete an instruction from the firmware branch vector table. This instruction should be used with great care, as you can delete any instruction with it (even the whole instruction set). It should be noted that this change is only temporary and will be lost at the time of the next system reset. It does not matter if the reset is soft or hard.

| Parameters | Description |
|---|---|
| OPCODE | The opcode number of the instruction that you want to be removed from the branch vector table. |

### Errors

None

You can allocate up to 33 display lists. All the display lists allocated can be used at the same time. Maximum subroutine length is 16K; however, subroutines may be extended by jumping to another display list. A display list can call another display list as a subroutine, and up to 32 levels of calling are supported.

### 2.4.2 Extended Display List Processing

The EXTENDED mode can be invoked only if you have a MC68000 system with firmware 10A and above. Upon power-up, the RM-9460 sets itself in NORMAL display list processing mode. To enter EXTENDED mode, you must send the SET DISPLAY LIST MODE SWITCH (SDLMS) instruction to the RM-9460. The SDLMS instruction has a mode select parameter DS, which can be set to 0 for NORMAL mode or 1 for EXTENDED mode. If you intend to use the EXTENDED mode, you must send the SDLMS instruction with the DS parameter set to 1 before any other display list instructions are executed.

In EXTENDED mode, display list RAM is segmented into 4K-byte blocks. Depending on the available memory of your system, you may allocate up to 244 blocks for a maximum of 1 million bytes in each display list.

Each display list is mapped so that regardless of the physical address within the RM-9460, the first block is located at address 8000(H), the second block is located at address 9000(H), the third block is located at address A000(H), and each succeeding block is located 1000(H) bytes beyond the previous block. Therefore, if a display list is allocated 20 blocks, its addressable range would be from 8000(H) to 1BFFF(H).

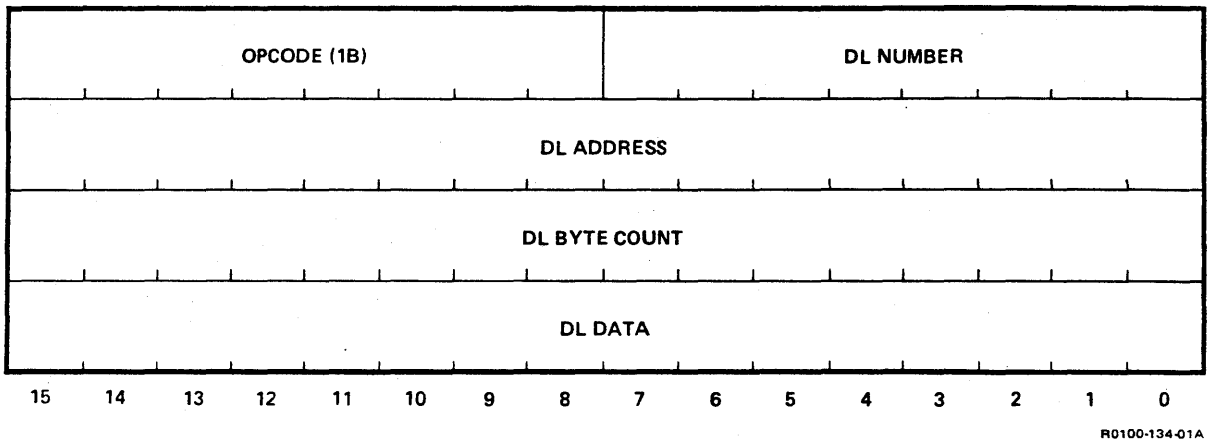
In EXTENDED mode, you can create a COMMON display list FF(H) that can be resident at all times. It is allocated and loaded in the same manner as other extended display lists, except that it is mapped into addresses 0000(H) to 7FFE(H), allowing a maximum COMMON display list size of 32K bytes (or eight blocks). Accordingly, regardless of the size of all other extended display lists, they will be able to access the COMMON display list.

A total of 256 separate display lists can be allocated. The REGULAR display lists are 0 - FE(H), excluding 80(H), which is system-reserved. The COMMON display list is FF(H). Only 33 of these display lists can be used at the same time. Maximum subroutine length is just under a million bytes, if you have the maximum available memory. However, up to 32 levels of nested subroutine calls allow you to effectively extend the length of the subroutines.

- Multiply Display List Registers (MULDLR)
- Divide Display List Registers (DIVDLR)
- Add Display List Registers (ADLR)
- Subtract Display List Registers (SUBDLR)
- AND Display List Registers (ANDDLR)
- OR Display List Registers (ORDLR)
- XOR Display List Registers (XORDLR)
- Load Display List Register (LDLR)
- Store Display List Register (STDLR)
- Parameter Load Display List Registers (PLDLR)
- Jump Conditional Upon Display List Register (JDLR)
- Switch Low Speed Mode (SWLOW)
- Switch Low Speed Mode (SWLOW)
- Set Display List Mode Switch (SDLMS)

| Errors | Description |
|---|---|
| (NORMAL or EXTENDED mode) | |
| Code = 3B01 | Attempt to execute ALDL from within a display list. |
| Code = 3B02 | Not enough RAM available for allocation. |
| Code = 3B0E | Attempt to allocate a display list already allocated. |
| (NORMAL mode only) | |
| Code = 3B0C | Invalid DL NUMBER. |
| (EXTENDED mode only) | |
| Code = 3B1F | Invalid NUMBER OF BLOCKS, 0 or a value greater than 256. |

## LOAD DISPLAY LIST (LDL)



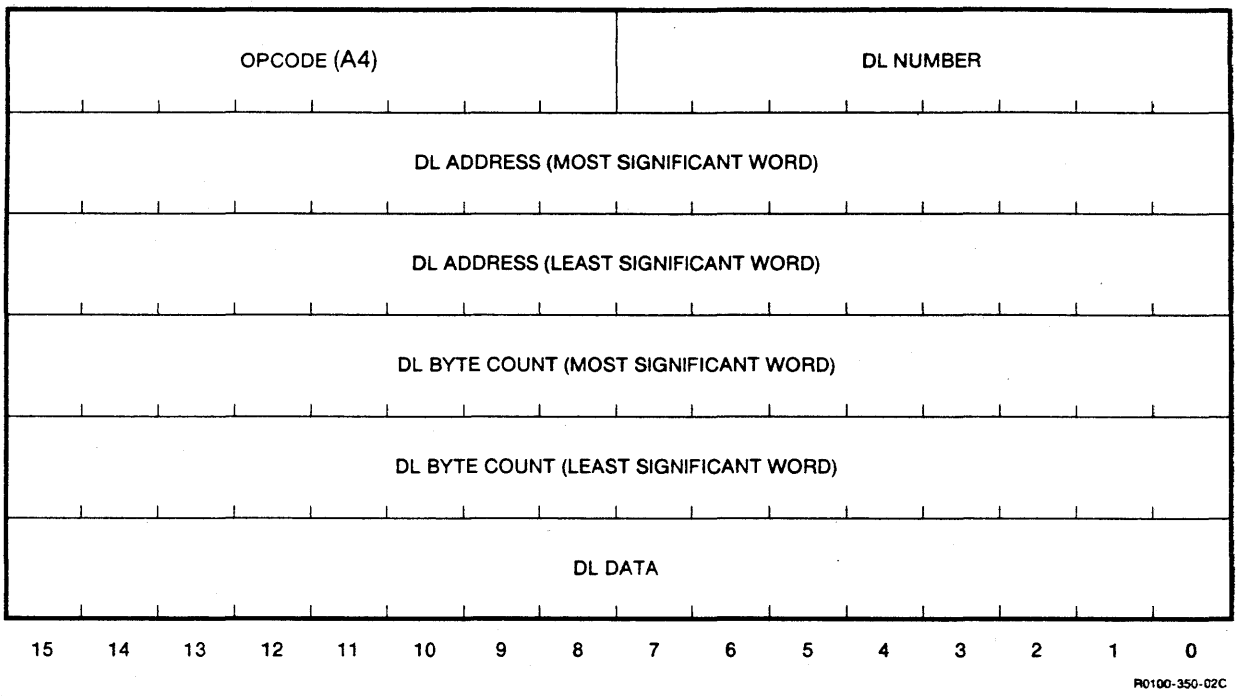

LDL is a special-format instruction which is used to store RM-9460 instructions or (Z80 instructions with the Z80 system processor) in display list RAM for subsequent execution through the CDL, XIM, or CCM instructions. This instruction cannot be executed from within a display list.

| Parameters | Description |
|---|---|
| DL NUMBER | Defines the display list number to be loaded. Can take on values of 0 through 1F(H) for normal display lists and FF(H) for the COMMON display list. |
| DL ADDRESS | Defines the starting internal address where data is to be loaded. The DL ADDRESS must be in the range of 8000(H) to BFFE(H) as this is where all display lists are stored. |
| DL BYTE COUNT | Defines the number of bytes in the instruction that are to be loaded into RAM. The value of DL BYTE COUNT should range from 0 to 16384 since larger counts overflow the display list area. |

### Data Format

The data for this instruction is formatted so that the bytes from each word are accessed high byte, then low byte, and loaded into successively-increasing byte addresses.

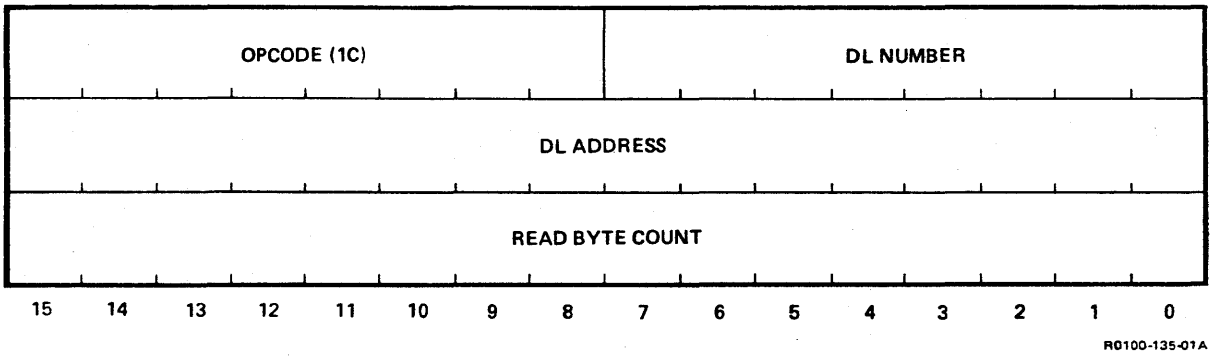
## EXTENDED LOAD DISPLAY LIST (ELDL)




ELDL is a special-format instruction which is used to store RM-9460 instructions in display list RAM for subsequent execution through the CDL, ECDL, XIM, EXIM or CCM instructions. This instruction cannot be executed from within a display list.

| Parameters | Description |
|---|---|
| DL NUMBER | Defines the display list number to be loaded. Can take on values of 0 through FE(H) for normal display lists and FF(H) for the COMMON display list. |
| DL ADDRESS | Defines the starting internal address where data is to be loaded. The DL ADDRESS must be in the range of addresses for the display list references (DL NUMBER). |
| DL BYTE COUNT | Defines the number of bytes in the instruction that are to be loaded into RAM. The value of DL BYTE COUNT should be no larger than the display list byte size, since larger counts would overflow the display list area. |

## READ DISPLAY LIST (RDL)



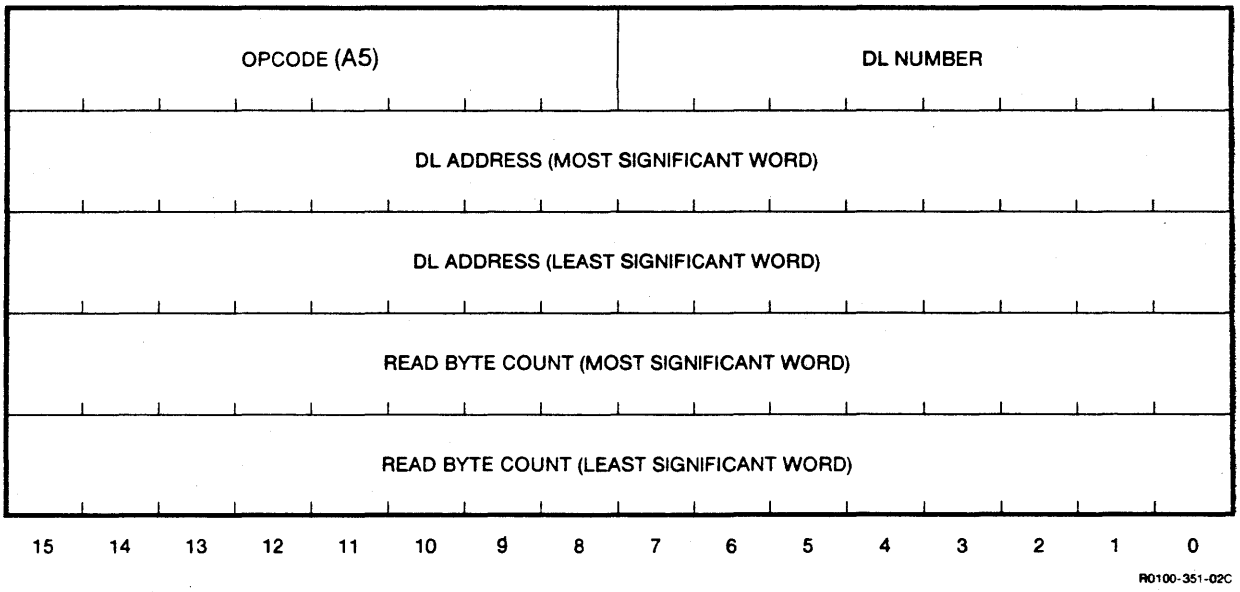

RDL is a special-format instruction which is used for reading back the contents of display list RAM storage to the host computer. This instruction cannot be executed from within a display list.

| Parameters | Description |
|---|---|
| DL NUMBER | Defines the display list number to be accessed for readback. Can take on values 0 through 1F(H) for normal display lists and FF(H) for the COMMON display list. |
| DL ADDRESS | Defines the starting internal address from which data will be read back. The DL address should be in the range 8000(H) to BFFE(H). |
| READ BYTE COUNT | Defines the number of bytes to be read back to the host computer. If it is odd, then (READ BYTE COUNT + 1) bytes are read back to the host computer. |

### Data Format

The readback data from this instruction is constructed so that data from successive byte addresses is stored high byte first, then low byte within each readback word.

## EXTENDED READ DISPLAY LIST (ERDL)



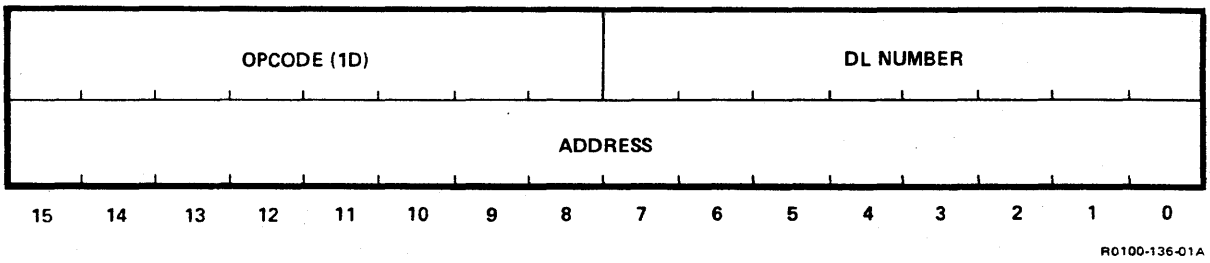

ERDL is a special-format instruction which is used for reading back the contents of display list RAM storage to the host computer. This instruction cannot be executed from within a display list.

| Parameters | Description |
|---|---|
| DL NUMBER | Defines the display list number to be accessed for readback. Can take on values 0 through FE(H) for normal display lists and FF(H) for the COMMON display list. |
| DL ADDRESS | Defines the starting internal address from which data will be read back. The DL address should be within the address range of the referenced display list (DL NUMBER). |
| READ BYTE COUNT | Defines the number of bytes to be read back to the host computer. If it is odd, then (READ BYTE COUNT + 1) bytes are read back to the host computer. |

### Data Format

The readback data from this instruction is constructed so that data from successive byte addresses is stored high byte first, then low byte within each readback word.

## CALL CONTROL MEMORY (CCM) (Z80 System Processor only)




CCM is a special-format instruction which is used to transfer control from the Ramtek-supplied firmware package to a user-loaded Z80 subroutine that resides in display list RAM. The following conditions are placed upon the user's subroutine:

1. The subroutine may use any Z80 registers except I or R.
2. The subroutine must maintain the integrity of the stack; that is, for each PUSH onto the stack, there must be a corresponding POP off the stack or the RET sequence at the end of the subroutine fails.
3. The subroutine must pass control back to the main Z80 control firmware through one of the RET instructions.
4. The subroutine must not use the Z80 halt opcode 76(H).

| Parameters | Description |
|---|---|
| DL NUMBER | Defines the display list where the subroutine to be called resides. This display list must have been allocated and loaded prior to the time of the CCM. |
| ADDRESS | Defines the internal address of the subroutine entry-point. This ADDRESS must be in the range of 8000(H) to BFFE(H). |

| Errors | Description |
|---|---|
| Code = 1D01 | Illegal instruction; attempt to use CCM instruction on RM-9460 with an MC68000 system processor. |
| Code = 1D0C | Invalid DL NUMBER; call to subroutine not attempted. |
| Code = 1D0D | DL does not exist; call to subroutine not attempted. |

| Errors | Description |
|---|---|
| Code = 1E0C | Invalid DL NUMBER; all data discarded. |
| Code = 1E0D | DL does not exist; all data discarded. |
| Code = 1E4D | Illegal COMMON display list address in extended mode. |
| Code = 1E4E | Illegal normal display list address in extended mode. |

| Errors | Description |
|---|---|
| Code = A60C | Invalid DL NUMBER; all data discarded. |
| Code = A60D | DL does not exist; all data discarded. |
| Code = A64D | Illegal COMMON display list address in extended mode. |
| Code = A64E | Illegal normal display list address in extended mode. |
| Code = A64F | Illegal display list mode for instruction to operate. |

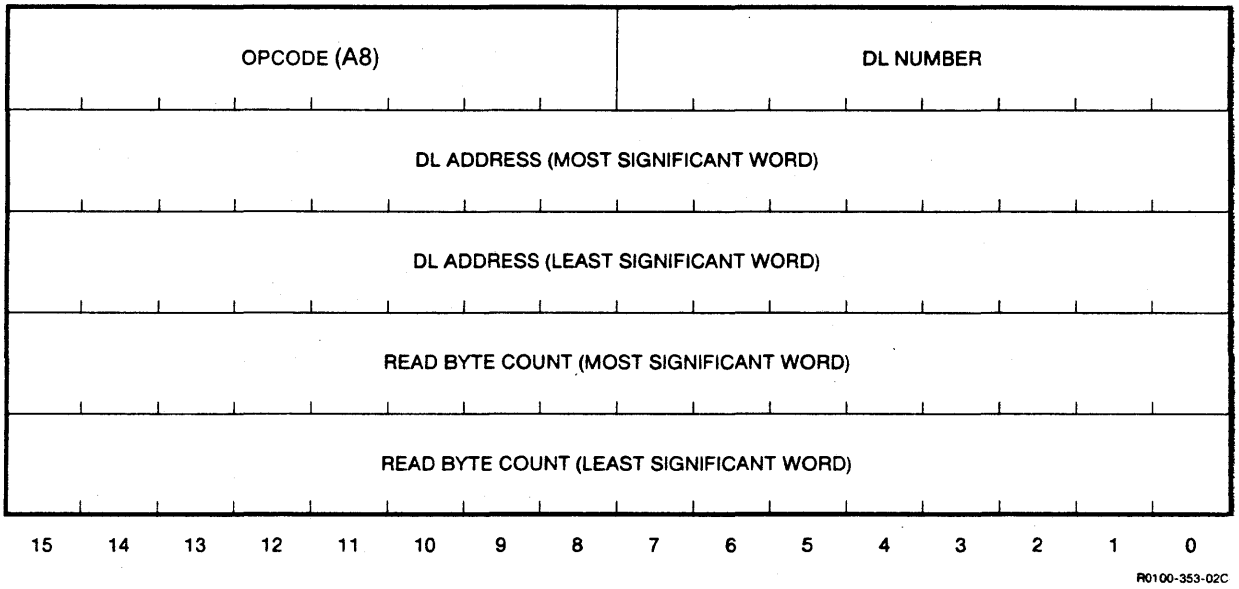
| Errors | Description |
|---|---|
| Code = 1F01 | Attempt to execute LDLRP from within a display list. |
| Code = 1F0C | Invalid DL NUMBER; all data is discarded. |
| Code = 1F0D | DL does not exist; all data is discarded. |
| Code = 1F4D | Illegal COMMON display list address in extended mode. |
| Code = 1F4E | Illegal normal display list address in extended mode. |

Data Format

The data for this instruction is formatted so that the bytes from each word are accessed low byte, then high byte and loaded into successively-increasing byte addresses.

| Errors | Description |
|---|---|
| Code = A701 | Attempt to execute LDLRP from within a display list. |
| Code = A70D | DL does not exist; all data is discarded. |
| Code = A74D | Illegal COMMON display list address in extended mode. |
| Code = A74E | Illegal normal display list address in extended mode. |
| Code = A74F | Illegal display list mode for instruction to operate. |

## EXTENDED READ DISPLAY LIST REVERSE PACKING (ERDLRP)




ERDLRP is a special-format instruction which is identical to the ERDL instruction except that the bytes in each data word are accessed in the opposite order; that is, low byte first, then high byte. This instruction cannot be executed from within a display list.

### Parameters

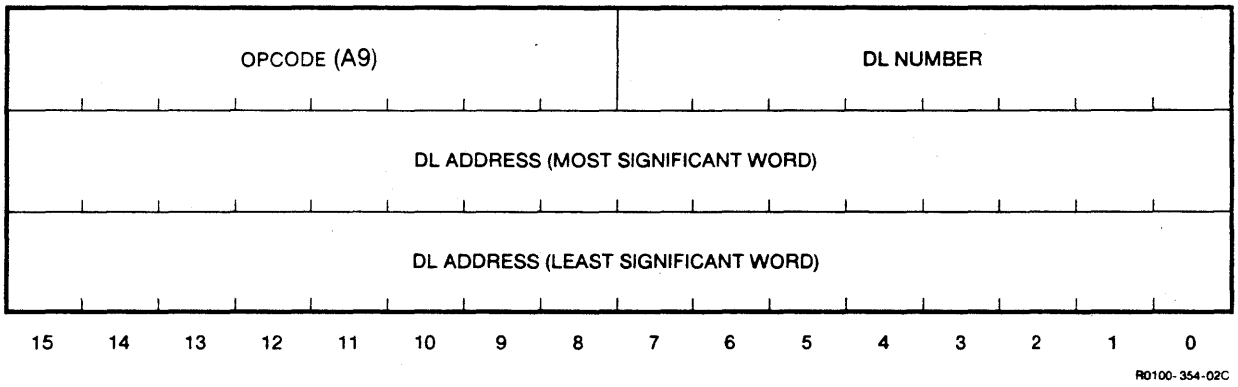
Same as ERDL Instruction

### Data Format

The readback data from this instruction is constructed so that the data from successive addresses is stored low byte, then high byte within each readback word.

| Errors | Description |
|---|---|
| Code = A801 | Attempt to execute ERDLRP from within a display list. |
| Code = A80D | DL does not exist; no data readback. |
| Code = A84D | Illegal COMMON display list address in extended mode. |
| Code = A84E | Illegal normal display list address in extended mode. |
| Code = A84F | Illegal display list mode for instruction to operate. |

## EXTENDED CALL DISPLAY LIST (ECDL)

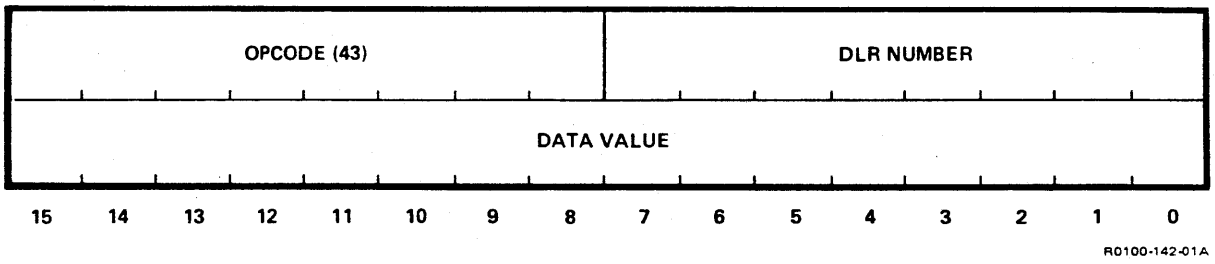



ECDL is a special-format instruction which begins RM-9460 display-list execution from the host or calls other display lists from the current display list. Up to 32 calls may be nested, after which stack overflow occurs. Once the ECDL instruction transfers execution to the specified display list and starting address, execution continues in a linear sequence until a RETDL instruction is executed. At that point, execution is transferred to the next instruction after the ECDL instruction. (ECDL and RETDL instructions are used as high-level language subroutine calls.)

| Parameter | Description |
|---|---|
| DL NUMBER | Defines the display list to be called for internal execution. DL NUMBER can range from 0 through FE(H) for explicit calls or FF(H) for calls to the current display list. |
| DL ADDRESS | Defines the address of the display-list routine to start internal execution. The DL ADDRESS must be within the address range of the selected display list (DL NUMBER). This should point to an instruction opcode word. |

| Errors | Description |
|---|---|
| Code = A90D | DL does not exist (display-list execution is terminated). |
| Code = A94D | Illegal COMMON display list address in extended mode. |
| Code = A94E | Illegal normal display list address in extended mode. |
| Code = A94F | Illegal display list mode for instruction to operate. |

## SET DISPLAY LIST REGISTER (SDLR)

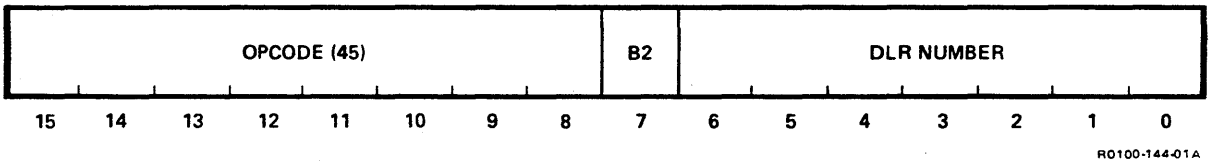



SDLR is a special-format instruction which allows the user to set the contents of any of the 16-display list registers (DLRs) to a 16-bit value.

| Parameters | Description |
|---|---|
| DLR NUMBER | Defines the DLR to be loaded. The DLR NUMBER must have a value in the range 0 to 0F(H). |
| DATA VALUE | Defines the 16-bit value to be loaded into the specified DLR. |

| Errors | Description |
|---|---|
| Code = 4310 | DLR NUMBER out of range. |

## INCREMENT DISPLAY LIST REGISTER (IDLR)

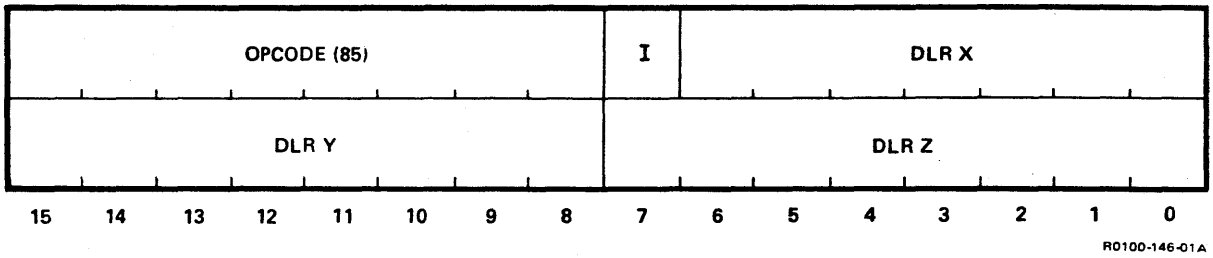



IDLR is a special-format instruction which increments the specified DLR by 1 or 2.

| Parameters | Description |
|---|---|
| DLR NUMBER | Defines the DLR to be incremented. The DLR NUMBER must have a value in the range 0 to 0F(H). |
| B2 | If B2 = 0, the DLR is incremented by 1.<br>If B2 = 1, the DLR is incremented by 2. |

| Errors | Description |
|---|---|
| Code = 4510 | DLR NUMBER out of range. |

## DIVIDE DISPLAY LIST REGISTERS (DIVDLR)




DIVDLR is a special-format instruction which divides the contents of DLR Y by the contents of DLR Z (as two's complement numbers) and stores the result in DLR X; that is, (DLR X) = (DLR Y)/(DLR Z). DLR X, Y, or Z can be either unique or the same DLR number.

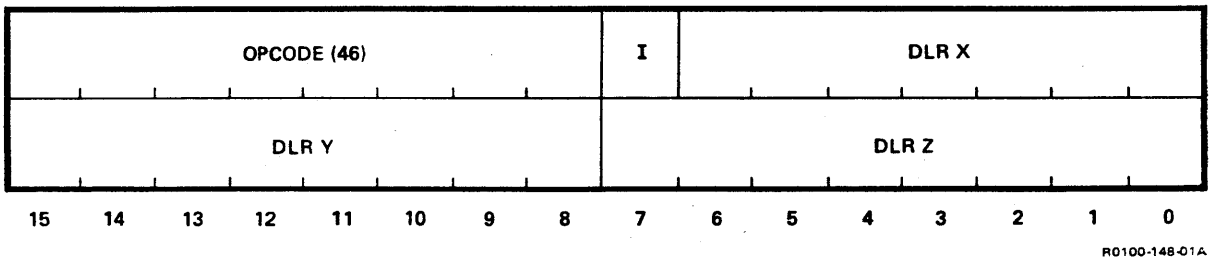
The division is performed as follows:

The signs of the operands are checked, and if either or both are negative, they are converted to positive numbers and the resultant sign is saved. The division is performed, the result is returned, and any remainder is ignored. If the result should be negative, it is complemented.

| Parameters | Description |
|---|---|
| DLR X | Defines the DLR to contain the result of the division operation, range 0 to 0F(H). |
| DLR Y | Defines the DLR value to be divided by the contents of DLR Z, range 0 to 0F(H). |
| DLR Z | Defines the DLR value to be divided into the contents of DLR Y, range 0 to 0F(H). |
| I | If I = 0, (DLR X) = (DLR Y)/(DLR Z).<br><br>If I = 1, the second word is treated as an immediate value (rather than a DLR specifier) and (DLR X) = (DLR X)/immediate value. |

| Errors | Description |
|---|---|
| Code = 8510 | DLR NUMBER out of range. |

## SUBTRACT DISPLAY LIST REGISTER (SUBDLR)

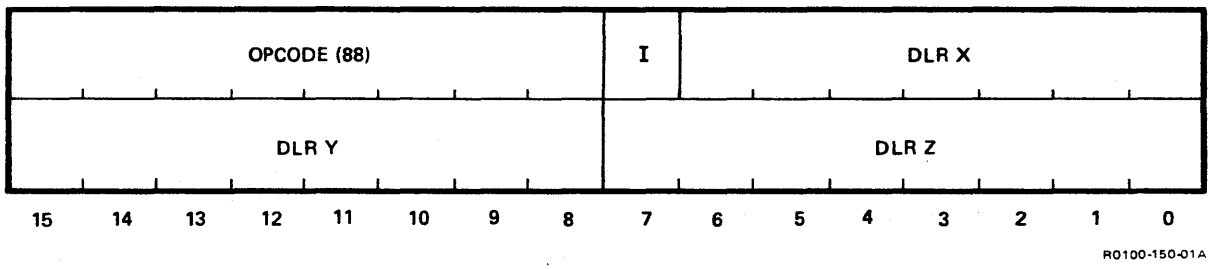



SUBDLR is a special-format instruction which subtracts the contents of DLR Z from the contents of DLR Y (as two's complement numbers) and stores the results in DLR X; that is, (DLR X) = (DLR Y) - (DLR Z). DLR X, Y, or Z can be either unique or the same DLR number.

| Parameters | Description |
|---|---|
| DLR X | Defines the DLR to contain the result of the subtraction operation, range 0 to 0F(H). |
| DLR Y | Defines the DLR from which the contents of DLR Z are to be subtracted, range 0 to 0F(H). |
| DLR Z | Defines the DLR value to be subtracted from the contents of DLR Y, range 0 to 0F(H). |
| I | If I = 0, (DLR X) = (DLR Y) - (DLR Z).<br><br>If I = 1, the second word is treated as an immediate value (rather than a DLR specifier) and (DLR X) = (DLR X) - immediate value. |

| Errors | Description |
|---|---|
| Code = 610 | DLR NUMBER out of range. |

## OR DISPLAY LIST REGISTERS (ORDLR)

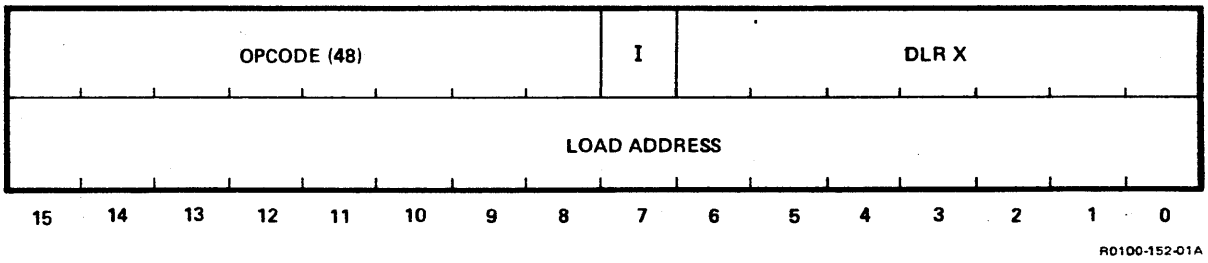



ORDLR is a special-format instruction which ORs the contents of DLR Z to the contents of DLR Y and stores the result in DLR X; that is, (DLR X) = (DL DLR X, Y, or Z can be either unique or the same DLR number.

| Parameters | Description |
|---|---|
| DLR X | Defines the DLR that will contain the result of the .OR. operation, range 0 to 0F(H). |
| DLR Y | Defines the DLR value to be ORed with the contents of DLR Z, range 0 to 0F(H). |
| DLR Z | Defines the DLR value to be ORed with the contents of DLR Y, range 0 to 0F(H). |
| I | If I = 0, (DLR X) = (DLR Y) .OR. (DLR Z).<br><br>If I = 1, the second word is treated as an immediate value (rather than a DLR specifier) and (DLR X) = (DLR X) .OR. immediate value. |

| Errors | Description |
|---|---|
| Code = 8810 | DLR NUMBER out of range. |

## LOAD DISPLAY LIST REGISTER (LDLR)

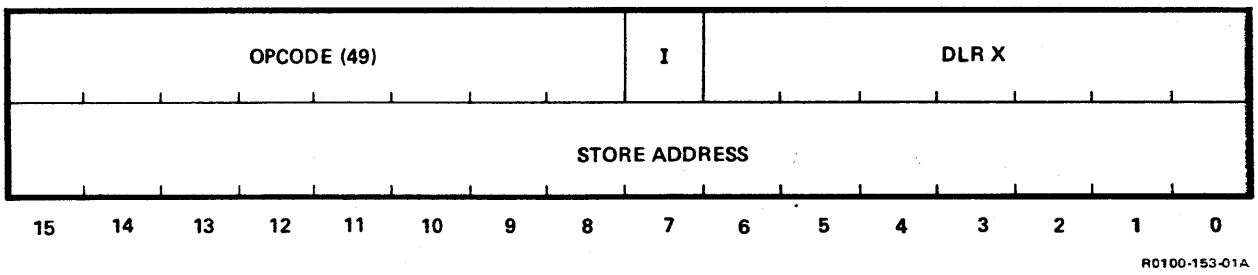



LDLR is a special-format instruction which loads DLR X with the 16-bit value stored at the address specified by the I bit and LOAD ADDRESS in the current display list.

| Parameters | Description |
|---|---|
| DLR X | Defines the DLR to be loaded with data from memory, range 0 to 0F(H). |
| I | Defines whether the load is direct or indirect through a display list register.<br><br>If I = 0, LOAD ADDRESS is used directly.<br>If I = 1, LOAD ADDRESS defines a display list register containing the actual LOAD ADDRESS. |
| LOAD ADDRESS | Contains either a memory address or a display list register number. |

| Errors | Description |
|---|---|
| Code = 4810 | DLR NUMBER out of range. |
| Code = 484D | Illegal COMMON display list address in extended mode. |
| Code = 484E | Illegal NORMAL display list address in extended mode. |

## STORE DISPLAY LIST REGISTER (STDLR)

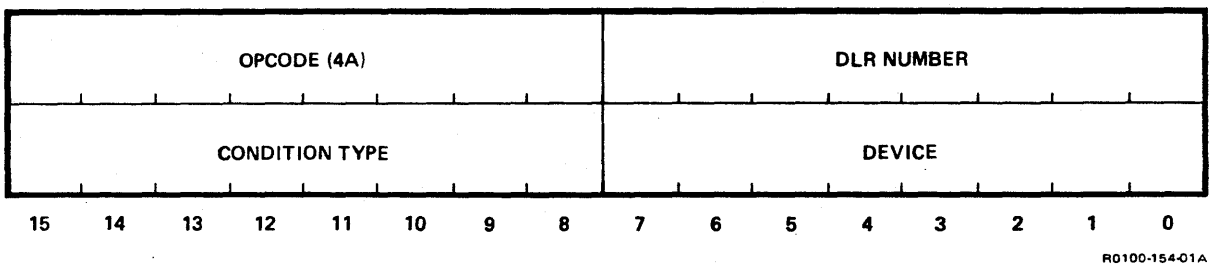



STDLR is a special-format instruction which stores the contents of DLR X at the internal address defined by the I bit and STORE ADDRESS in the current display list.

| Parameters | Description |
|---|---|
| DLR X | Defines the DLR value to be stored in internal memory, range 0 to 0F(H). |
| I | Defines whether the STORE ADDRESS is direct or indirect through a display list register.<br><br>If I = 0, STORE ADDRESS is used directly.<br>If I = 1, STORE ADDRESS defines a display list register containing the actual STORE ADDRESS. |
| STORE ADDRESS | Contains either a memory address or a display list register number. |

| Errors | Description |
|---|---|
| Code = 4910 | DLR NUMBER out of range. |
| Code = 494D | Illegal COMMON display list address in extended mode. |
| Code = 494E | Illegal normal display list address in extended mode. |

## PARAMETER LOAD DISPLAY LIST REGISTERS (PLDLR)




PLDLR is a special-format instruction which loads internal parameters into sets of DLRs. START-POINT, FORMAT WINDOW, WRITE-ENABLE WINDOW, or CURSOR VALUES can be loaded into sets of DLRs.

| Parameters | Description |
|---|---|
| DLR NUMBER | Defines the starting DLR NUMBER for parameter loading. |
| CONDITION TYPE | Defines the parameters to be loaded into the DLRs as outlined below. |
| DEVICE | Defines the cursor to be used in the parameter load. |

| DEVICE | CURSOR |
|---|---|
| 0 | Cursor #0 of serial link PCB #0 |
| 1 | Cursor #1 of serial link PCB #0 |
| 2 | Cursor #2 of serial link PCB #0 |
| 3 | Cursor #3 of serial link PCB #0 |
| 4 | Cursor #0 of serial link PCB #1 |
| 5 | Cursor #1 of serial link PCB #1 |
| 6 | Cursor #2 of serial link PCB #1 |
| 7 | Cursor #3 of serial link PCB #1 |

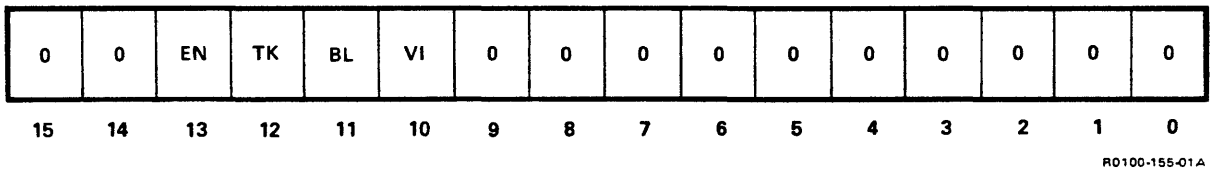
| Condition Type Code | Action |
|---|---|
| 0 | Loads the host original START-POINT operand. |

(DLR) = X-START-POINT (original)
(DLR + 1) = Y-START-POINT (original)

| Condition Type Code | Action |
|---|---|
| 8 | Loads the (global) CURSOR values specified by the device.<br><br>(DLR) = CURSOR X (cursor)<br>(DLR + 1) = CURSOR Y (cursor)<br>(DLR + 2) = CURSOR status |
| 9 | Loads the (local) CURSOR values specified by the device.<br><br>(DLR) = CURSOR X (local)<br>(DLR + 1) = CURSOR Y (local)<br>(DLR + 2) = CURSOR status |
| A(H) | Loads the tablet coordinates and status specified by the device.<br><br>(DLR) = X-coordinate<br>(DLR + 1) = Y-coordinate<br>(DLR + 2) = TABLET status |

**NOTE**

Cursor status for conditions 6-9 is in the format where the bits are defined as in the Read Cursor Status instructions.




| Parameters | Description |
|---|---|
| VI | Defines the current visible/invisible state of the cursor specified by DEVICE. If VI = 0, the cursor is not visible. If VI = 1, the cursor is visible. |
| BL | Defines the blink state of the cursor selected by DEVICE. If BL = 0, the cursor is non-blinking. If BL = 1, the cursor is blinking at a rate of one Hz. |

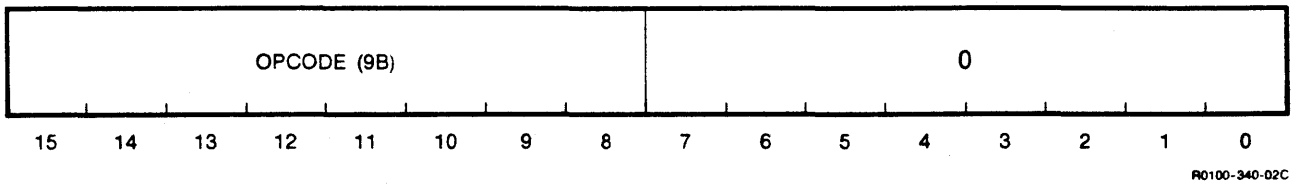
| Parameters | Description |
|---|---|
| MU | When set, the PUCK/STYLUS is in the menu area (outside window). |
| WI | When set, the PUCK/STYLUS is in the window area (specified through SCW or default window in menu mode). |
| PX | When set, the STYLUS/PUCK has left or entered the proximity of the table. If MU or WI is set, then the STYLUS or PUCK has entered the proximity of the tablet; while if MU and WI are reset, then the STYLUS or PUCK has left the proximity of the tablet. |
| TD | When set, an interrupt caused by moving the STYLUS/PUCK out of TD distance has been generated to the host. |
| F3, F2, F1, F0 | When any of these are set, the associated button on the PUCK has been depressed. For PUCKS with only one button or for a STYLUS, depressions will cause F0 to be set. |

| Errors | Description |
|---|---|
| Code = 4A10 | DLR NUMBER out of range. |
| Code = 4A11 | Illegal CONDITION TYPE was specified. |

| Errors | Description |
|---|---|
| Code = 4B0C | Invalid DL NUMBER (display list execution terminated). |
| Code = 4B0D | DL NUMBER does not exist. |
| Code = 4B10 | DLR NUMBER out of range. |
| Code = 4B20 | Illegal CONDITION value. |
| Code = 4B4D | Illegal COMMON display list address in extended mode. |
| Code = 4B4E | Illegal normal display list address in extended mode. |

| Errors | Description |
|---|---|
| Code = AC0D | DL NUMBER does not exist. |
| Code = AC10 | DLR NUMBER out of range. |
| Code = AC20 | Illegal CONDITION value. |
| Code = AC4D | Illegal COMMON display list address in extended mode. |
| Code = AC4E | Illegal normal display list address in extended mode. |
| Code = AC4F | Illegal display list mode for instruction to operate. |

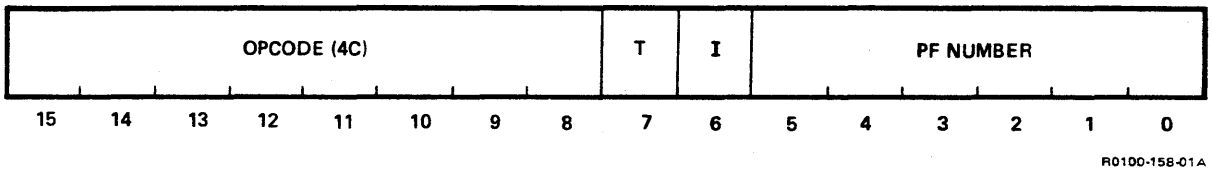
## SWITCH LOW SPEED MODE (SWLOW) (MC68000 System Processor only)




SWLOW is a special-format instruction which returns display list memory transfer to the original (low speed) mode. If an instruction does not work properly after you have switched to high speed mode with the SWHIGH instruction, execute this instruction to switch back to the standard GPIF I/O processing mode.

This page not used.

## ALLOCATE PROGRAMMABLE FONT (ALPF)



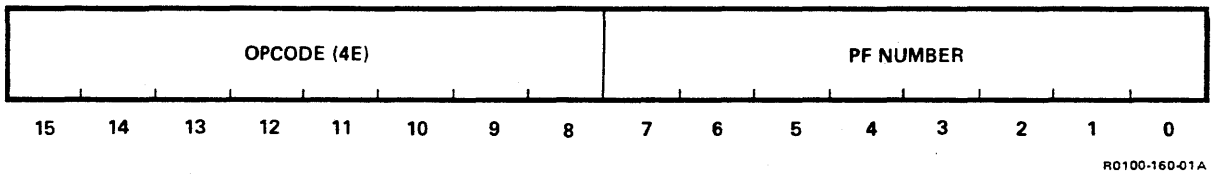

ALPF is a special-format instruction which obtains RAM from the unallocated pool based upon the type of font being set up (either 8 by 12 or 16 by 20 elements and lines, respectively, per character). The font may or may not be initialized to reflect the standard 7 by 9 font that is stored in PROM (this is instruction-defined). Character codes from 20(H) through 9F(H) may be programmed.

Font 0 is automatically allocated at power-on reset, and the standard font is initialized for operational compatibility with the RM-9000 series of display systems.

| Parameters | Description |
|---|---|
| PF NUMBER | Defines the number of the font to be allocated in the range of 0 to 0F(H). |
| I | Flags whether the font just allocated is to be initialized with a copy of the standard PROM font (I = 0 implies do not initialize). |
| T | Defines the type of font to be allocated. If T = 0, then an 8 by 12 font area is allocated, requiring one 4K block. If T = 1, then a 16 by 20 font area is allocated, requiring two 4K blocks. |

| Errors | Description |
|---|---|
| Code = 4C02 | Attempt to allocate a font where not enough free RAM exists. |
| Code = 4C0E | Attempt to allocate a font that has already been allocated. |
| Code = 4C12 | PF NUMBER is out of range. |

## ATTACH PROGRAMMABLE FONT (ATTPF)




ATTPF is a special-format instruction which defines the programmable font to be used for text generation with the current context. Programmable fonts must be allocated and attached before they can be used. The font specified must have been allocated, or the instruction causes an error interrupt (but does not change the font assignment for the current context).

| Parameters | Description |
|---|---|
| PF NUMBER | Defines the font to be assigned to the current context, range 0 to 0F(H) or FF(H) for the standard PROM font. |

| Errors | Description |
|---|---|
| Code = 4E12 | The font defined by PF NUMBER is out of range. |
| Code = 4E13 | The font defined by PF NUMBER has not been allocated. |

Data Definition

Twelve bytes of font-definition data follow immediately after the opcode/character code word. Font line 1 defines the top line of the 8 by 12 font matrix and font line 12 defines the bottom line of the font matrix. The high-order bit of each font byte represents the left margin of the font matrix and the low-order bit represents the right margin of the font matrix. This instruction is used only for 8 by 12 font types. For 16 by 20 fonts, use the LMPF instruction.

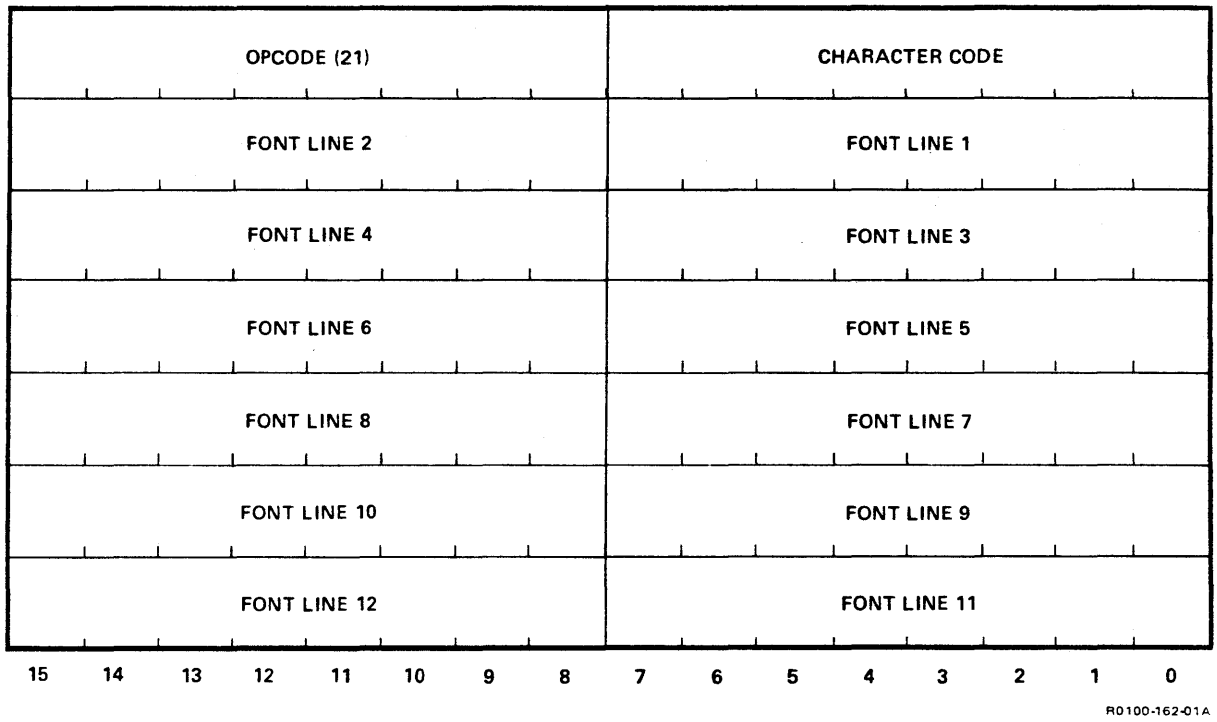
| Errors | Description |
|---|---|
| Code = 1503 | Font RAM not allocated and attached to the current context; data discarded. |
| Code = 1513 | Font to be loaded not 8 by 12; data discarded. |
| Code = 1514 | Illegal ASCII CHARACTER CODE; data discarded. |

Example:

The symbol described in figure 2-16 would be generated using the following sequence with the LPF instruction:

| HEX | Description |
|---|---|
| 159F | ;LPF + CHAR.CODE |
| 0010 | ;FONT LINE 1 & 2 |
| 3854 | ;FONT LINE 3 & 4 |
| 9210 | ;FONT LINE 5 & 6 |
| 1010 | ;FONT LINE 7 & 8 |
| 2800 | ;FONT LINE 9 & 10 |
| 0000 | ;FONT LINE 11 & 12 |
| 0C01 | ;WRITE TEXT WITH DATA FLAG |
| 0001 | ;NUMBER OF BYTES = 1 |
| 9F00 | ;CHARACTER CODE |

## LOAD PROGRAMMABLE FONT REVERSE PACKING (LPFRP)



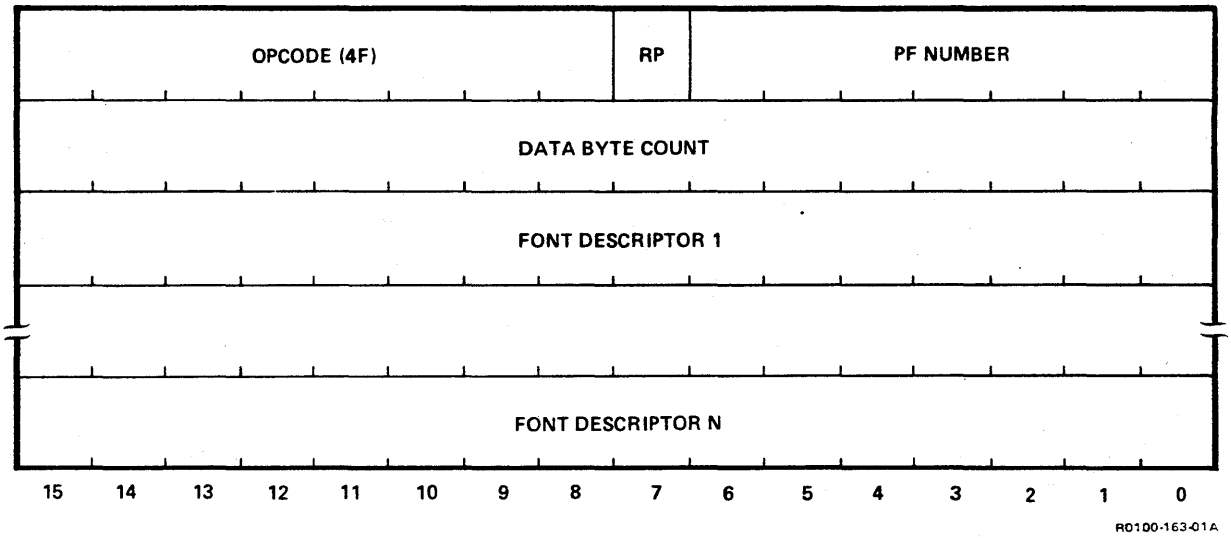

LPFRP is a special-format instruction which loads the 8 by 12 font data that is sent to display refresh memory when the byte value defined by CHARACTER CODE is transmitted as text data in a WT instruction. The operation is the same as the LPF instruction except that the font-line description bytes are reversed within each word of data.

Example:

The symbol described in figure 2-16 would be generated using the following sequence with the LPFRP instruction:

| HEX | Description |
|---|---|
| 219F | ;LPFRP + CHAR. CODE |
| 1000 | ;FONT LINES 2 & 1 |
| 5438 | ;FONT LINES 4 & 3 |
| 1092 | ;FONT LINES 6 & 5 |
| 1010 | ;FONT LINES 8 & 7 |
| 0028 | ;FONT LINES 10 & 9 |
| 0000 | ;FONT LINES 12 & 11 |
| 0C01 | ;WRITE TEXT WITH DATA FLAG |
| 0001 | ;NUMBER OF BYTES = 1 |
| 9F00 | ;CHARACTER CODE |

## LOAD MULTIPLE PROGRAMMABLE FONTS (LMPF)




LMPF is a special-format instruction which performs loading of random ASCII character codes with font data. The ASCII character codes must be in the range of 20(H) to 9F(H), or the data is ignored. The RP flag (bit 7 of the opcode word) defines the byte packing mode of the font data.

| Parameters | Description |
|---|---|
| PF NUMBER | Defines the programmable font to receive the font data. This number must be in the range of 0 to 0F(H), and the font must have been previously allocated. |
| RP | Defines the order in which the bytes of font data are accepted and used internally. If RP = 0, then the high data byte (bits 15 - 8) is used first (normal packing). If RP = 1, the low data byte (bits 7 - 0) is used first (reverse packing). |
| DATA BYTE COUNT | Defines the number of bytes of font data to follow. This number must be a multiple of 14 if the font has been allocated as an 8 by 12, or a multiple of 42 if 16 by 20; otherwise, an error interrupt is given, and the data is ignored. |
| FONT DESCRIPTOR | Contains font description data. |

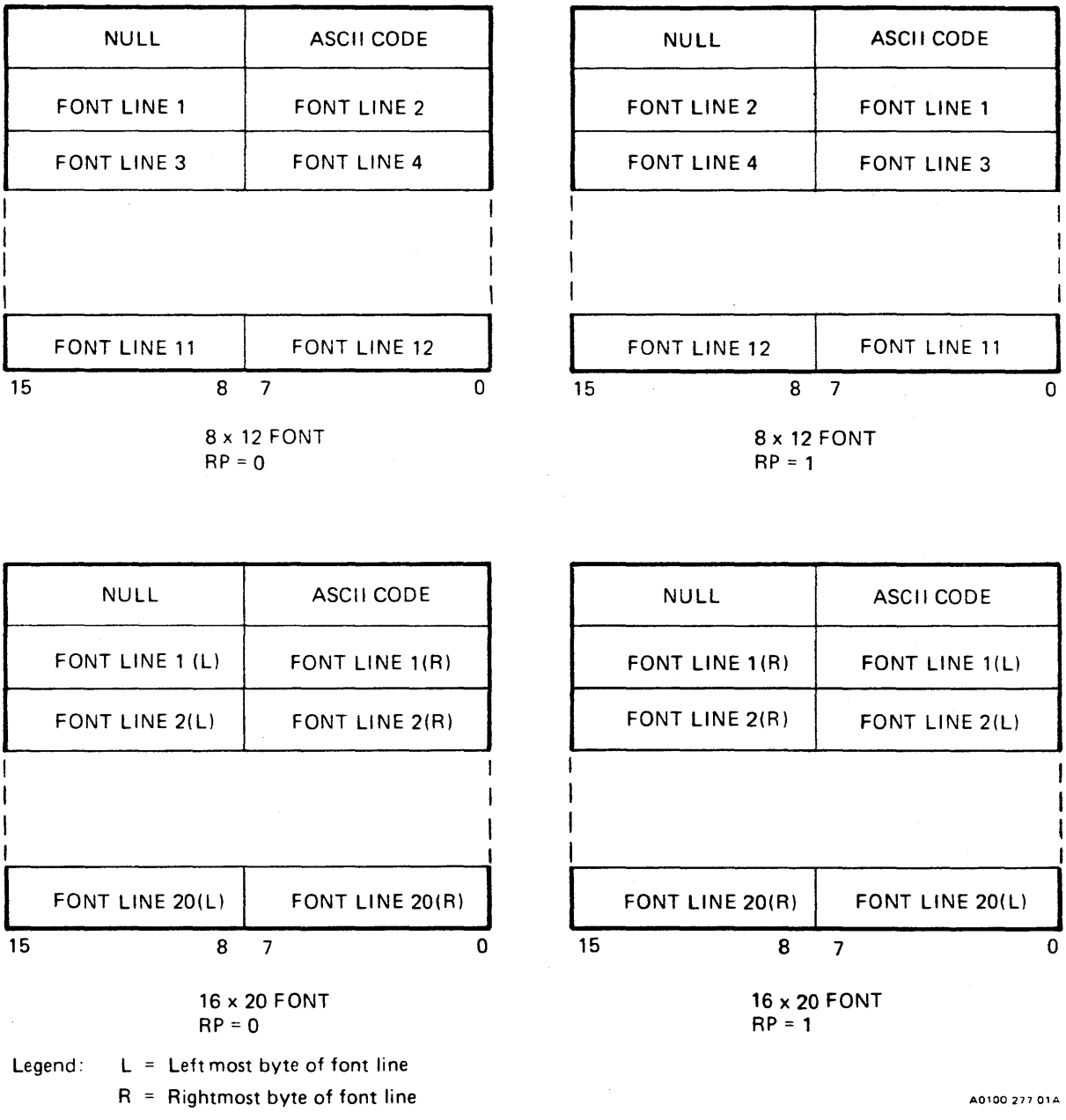
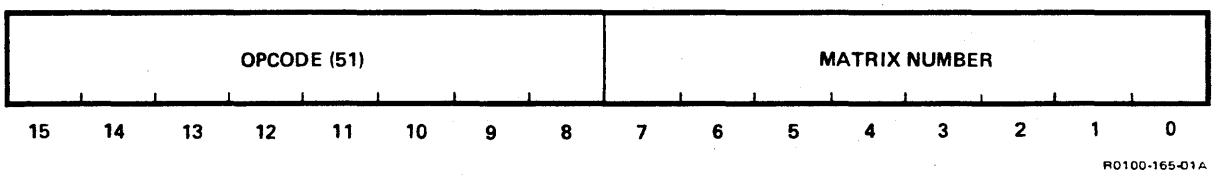



**Figure 2-17. LMPF Data Formats**

The coordinate transformation instructions are listed below:

- Push Matrix (PUSHM)
- Pop Matrix (POPM)
- Set Matrix (SETM)
- Load Current Matrix (LM)
- Store Current Matrix (SM)
- Multiply Matrices (MM)
- Multiply Matrices Immediate (MMI)
- Initialize Matrix (IM)
- Scale Matrix (SCALE)
- Translate Matrix (TRANS)
- Rotate Matrix (ROTAT)
- Read Matrix (READM)

## POP MATRIX (POPM)




POPM is a special-format instruction which retrieves the last matrix pushed onto the matrix stack (through the PUSHM instruction) and stores it in the specified matrix. If there are no entries on the matrix stack, an error interrupt is generated, and there is no change in the state of the specified matrix.

| Parameters | Description |
|---|---|
| MATRIX NUMBER | Defines the matrix to receive data from the matrix stack, range 0 through 07(H). |

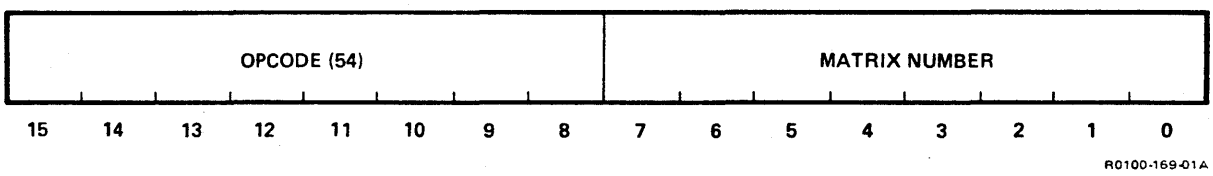
| Errors | Description |
|---|---|
| Code = 511B | MATRIX NUMBER out of range. |
| Code = 511C | No data on the matrix stack. |

These numbers are used as matrix coefficients and are loaded in the following format:

$$\begin{bmatrix} A & D & 0 \\ B & E & 0 \\ C & F & 1 \end{bmatrix}$$

| Error | Description |
|---|---|
| Code = 521B | MATRIX NUMBER out of range. |

## STORE CURRENT MATRIX (SM)

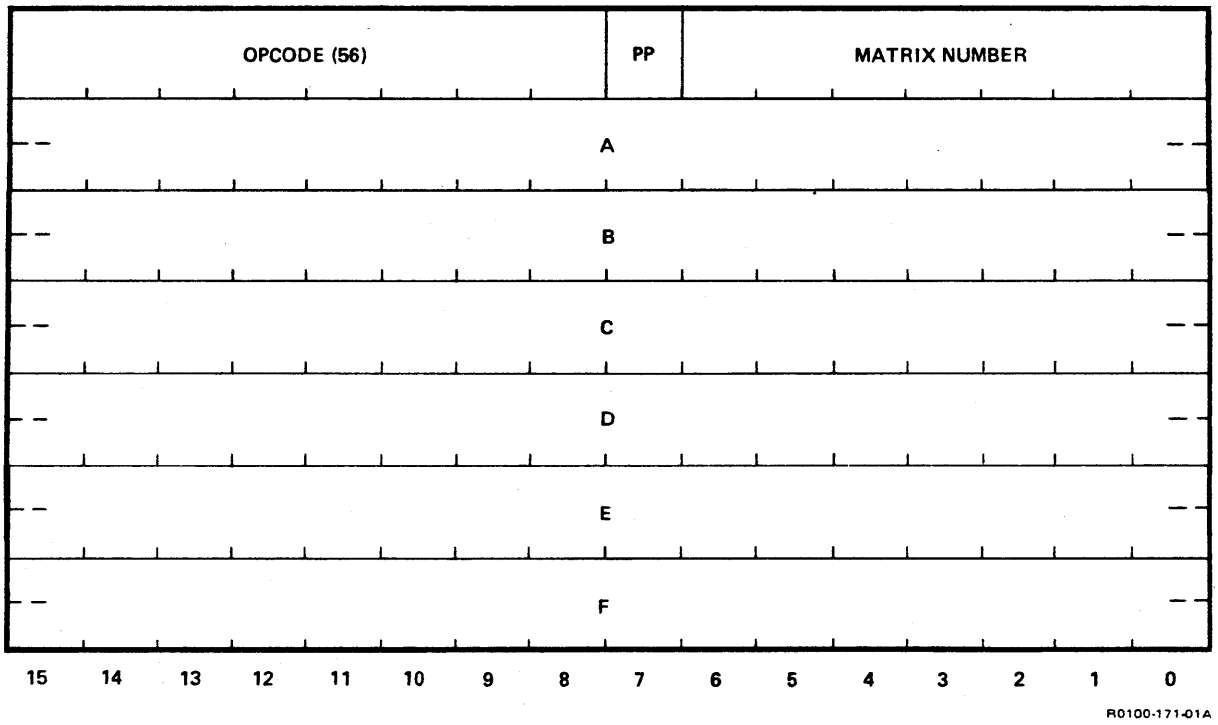



SM is a special-format instruction which stores the current matrix (matrix number 0) into the specified matrix. The contents of the current matrix are unchanged.

| Parameter | Description |
|---|---|
| MATRIX NUMBER | Defines the matrix to be loaded with the current matrix, range 0 through 07(H). |

| Error | Description |
|---|---|
| Code = 541B | MATRIX NUMBER out of range. |

## MULTIPLY MATRICES IMMEDIATE (MMI)



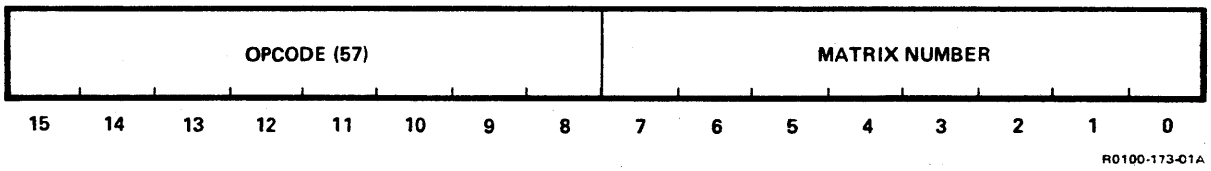

MMI is a special-format instruction which is used to multiply a specified matrix by the user-supplied matrix included in the instruction. The resulting matrix is always stored back into the specified matrix.

| Parameters | Description |
|---|---|
| MATRIX NUMBER | Defines the operand and result matrix, range 0 through 07(H). |
| PP | Defines the order of multiplication.<br>If PP = 0, then M = (DATA) * M<br>If PP = 1, then M = M * (DATA) |

### Data Format

The data consists of six 32-bit two's-complement fixed-point integer numbers, each composed of a 16-bit integer and a 16-bit function (two words per coefficient).

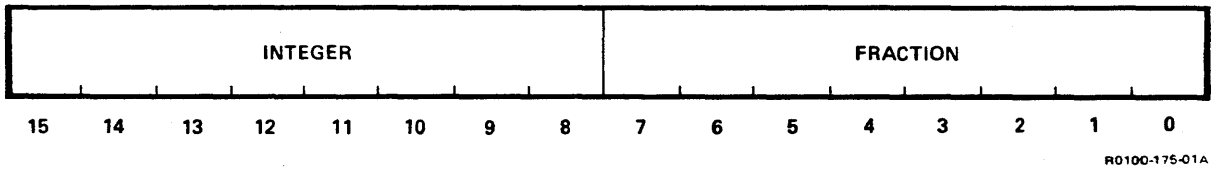
## INITIALIZE MATRIX (IM)




IM is a special-format instruction which sets the specified matrix to the identity matrix. All matrices are initialized to the identity matrix during power-on or reset.

| Parameters | Description |
|---|---|
| MATRIX NUMBER | Defines the matrix to be set to the identity matrix, range 0 through 07(H): $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |

| Errors | Description |
|---|---|
| Code = 571B | MATRIX NUMBER out of range. |




A value of 0100(H) is equal to a scale factor of 1; therefore a value greater than 0100(H) is used to scale upward.

Examples:

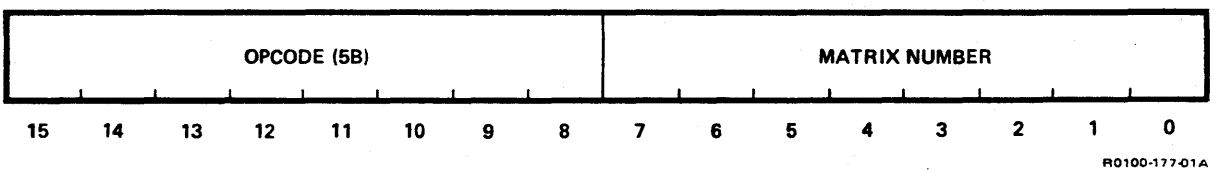
A scale factor of 0280(H) scales by 2.5. To scale downward, a value less than 0100(H) is used.

A scale factor of 0080(H) scales by 0.5. A negative scale factor carries a reflection around the scaled axis.

A value of FF00(H) is equal to a scale factor of -1 and causes an object to be drawn on the opposite side of the axis. X- and Y-scale may vary from 7FFF(H) (+127.99) to 0001 (+0.004) and from FFFF(-.004) to 8000 (-128.0).

| Errors | Description |
|---|---|
| Code = 581B | MATRIX NUMBER out of range. |

## READ MATRIX (READM)

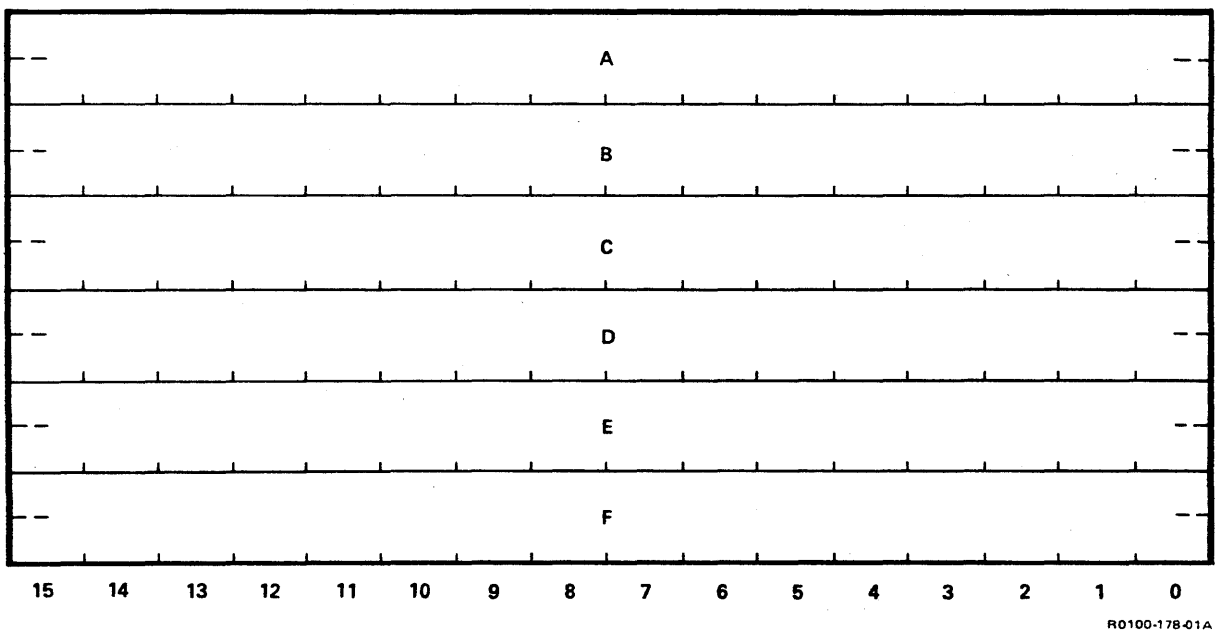



READM is a special-format instruction which reads back the current contents of the specified matrix. The host computer is expected to read back 12 words (24 bytes) of matrix coefficients; otherwise, interprocessor communication may halt.

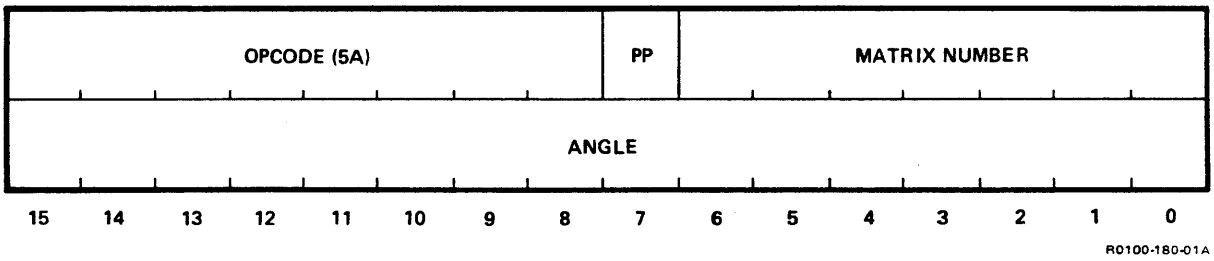
| Parameters | Description |
|---|---|
| MATRIX NUMBER | Defines the matrix to be read back, range 0 through 07(H). |

### Data Format

The host computer will read back 12 words according to the following format:



## ROTATE MATRIX (ROTAT)




ROTAT is a special-format instruction which multiplies the specified matrix by a specified angle expressed in degrees. The resultant matrix is always stored back into the specified matrix. The actual matrix operation that takes place is as follows:

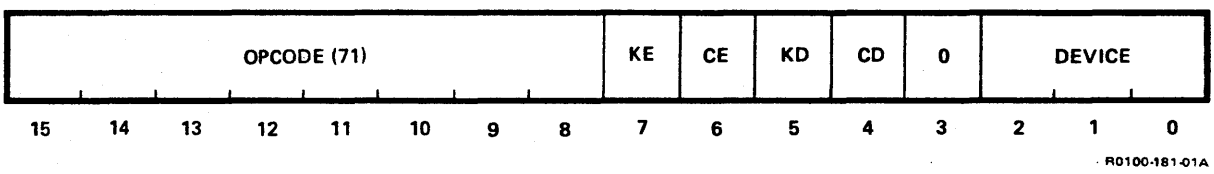
$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{bmatrix} = \begin{bmatrix} (a\cos\theta + b\sin\theta) & (c\cos\theta + e\sin\theta) & 0 \\ (b\cos\theta - a\sin\theta) & (e\cos\theta - d\sin\theta) & 0 \\ c & f & 1 \end{bmatrix}$$

| Parameters | Description |
|---|---|
| MATRIX NUMBER | Defines the matrix to be rotated. |
| PP | Defines the order of multiplication.<br>If PP = 0, then the order is as above.<br>If PP = 1, then the rotate matrix and the specified matrix are reversed. |

| Data Format | Description |
|---|---|
| ANGLE | A two's-complement, 16-bit integer that specifies the number of degrees the matrix is to be rotated. Positive values cause counterclockwise rotation with lower-left origin and clockwise rotation with upper-left origin. Negative values cause clockwise rotation with lower-left origin and counterclockwise rotation with upper-left origin. |

| Errors | Description |
|---|---|
| Code = 5A1B | MATRIX NUMBER out of range. |

## SET LOCAL FUNCTION STATE (SLFS)

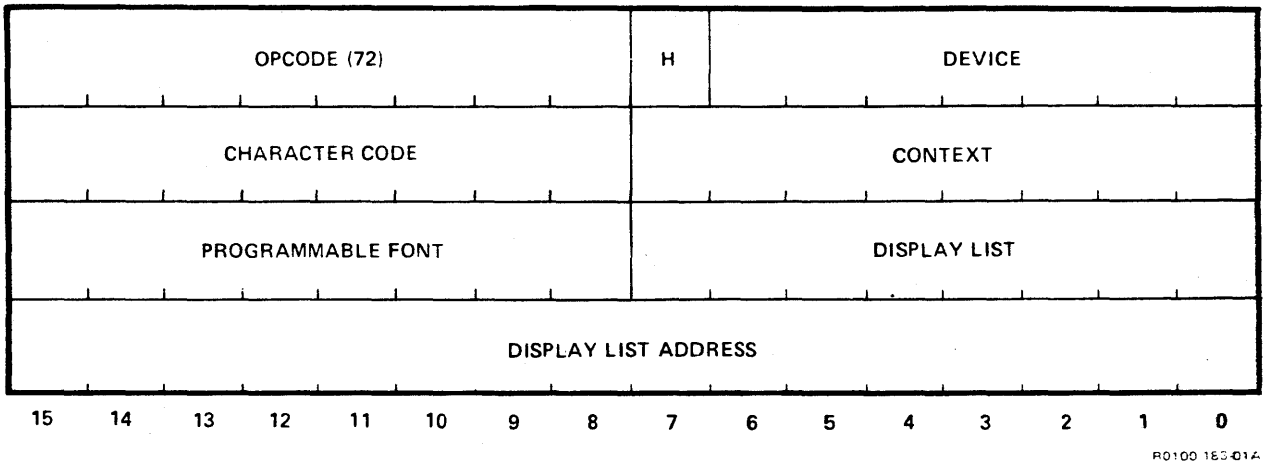



SLFS is a special-format instruction which specifies the enable/disable state of local keyboard or cursor functions. It is used as a master enable/disable "switch" for any specified device.

| Parameters | Description |
|---|---|
| DEVICE | Defines the keyboard or cursor device number for which local function scan and execution are enabled or disabled. |
| KE | This bit will enable keyboard local functions for the specified device number. If KE = 1, local functions are enabled. |
| CE | This bit will enable cursor local functions for the specified device number. If CE = 1, local functions are enabled. |
| KD | This bit will disable keyboard local functions for the specified device number. If KD = 1, local functions are disabled. |
| CD | This bit will disable cursor local functions for the specified device number. If CD = 1, local functions are disabled. |

Errors are generated during local function invocation if any of the associated environment items have been deallocated. For instance, suppose an echo state has been set up using keyboard 2, programmable font 4, and context 1. The font and context were previously allocated; otherwise errors 7603(H) or 7605(H) would have been generated. Somehow font 4 was deallocated and then a character was typed on keyboard 2 in the echo range. When the environment is set up for the character echo, the font does not exist; error 7132(H) is generated, the echo is not performed, and the previous environment is restored. Similarly, if a keyboard local function is set up as above and the font is deallocated, error 7192(H) will be generated, the local function will not be performed, and the previous environment will be restored.

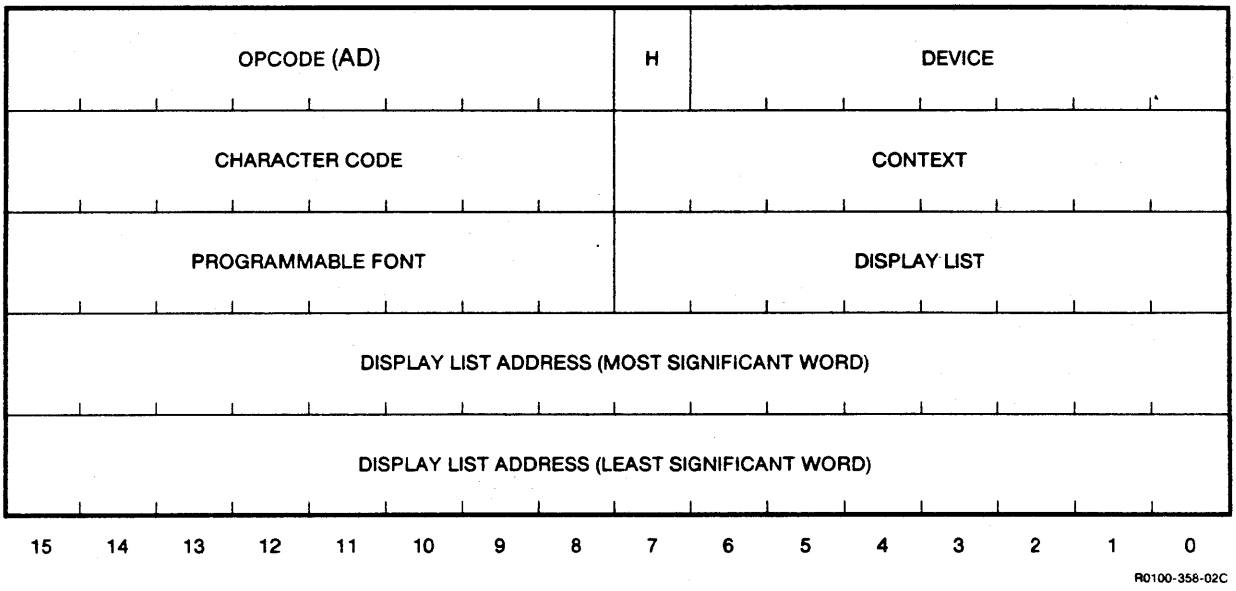
## SET LOCAL KEYBOARD FUNCTION (SLKF)




SLKF is a special-format instruction which inserts a keyboard character code into the local function table. Whenever this character is depressed on the specified keyboard (assuming keyboard local functions are enabled for this keyboard), the specified context, programmable font, and display list are made current and the display list is called at the specified address. When the display list execution terminates, the previous context, programmable font, and display list are made current and host computer instruction execution continues.

| Parameters | Description |
|---|---|
| DEVICE | Defines the keyboard device number this local function is associated with. |
| H | Specifies whether the host is to be notified when this local function is executed. If H = 0, the host computer is not notified and the character is not available for the host readback. If H = 1, the computer is notified like any other character (for example, RCV status bit in interface set); the character is inserted in the keyboard buffer and becomes available for readback. |
| CONTEXT | Defines the context used during execution of this local function. Since a context contains all the normal-format parameters, this defines the picture-creation environment used. |
| CHARACTER CODE | Defines the character code this local function is associated with. |

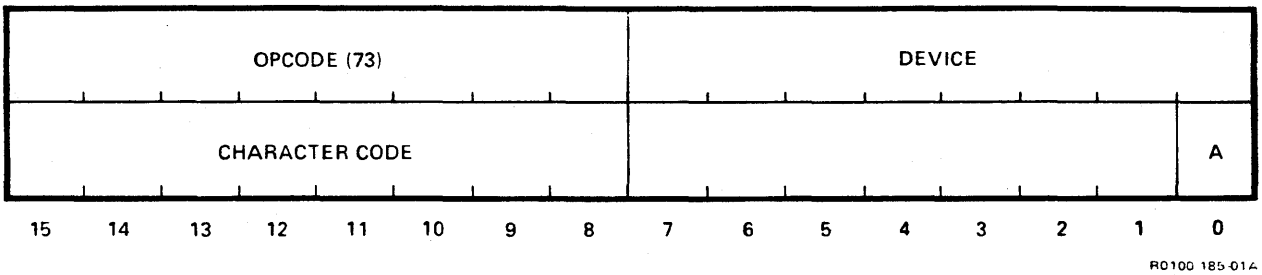
## SET LOCAL KEYBOARD FUNCTION EXTENDED (SLKFE)




SLKFE is a special-format instruction which inserts a keyboard character code into the local function table. Whenever this character is depressed on the specified keyboard (assuming keyboard local functions are enabled for this keyboard), the specified context, programmable font, and display list are made current and the display list is called at the specified address. When the display list execution terminates, the previous context, programmable font, and display list are made current and host computer instruction execution continues.

| Parameters | Description |
|---|---|
| DEVICE | Defines the keyboard device number this local function is associated with. |
| H | Specifies whether the host is to be notified when this local function is executed. If H = 0, the host computer is not notified and the character is not available for the host readback. If H = 1, the computer is notified like any other character (for example, RCV status bit in interface set); the character is inserted in the keyboard buffer and becomes available for readback. |
| CONTEXT | Defines the context used during execution of this local function. Since a context contains all the normal-format parameters, this defines the picture-creation environment used. |

## CLEAR LOCAL KEYBOARD FUNCTION (CLKF)




CLKF is a special-format instruction which deletes one or all keyboard-character associations for the specified keyboard from the local function table.

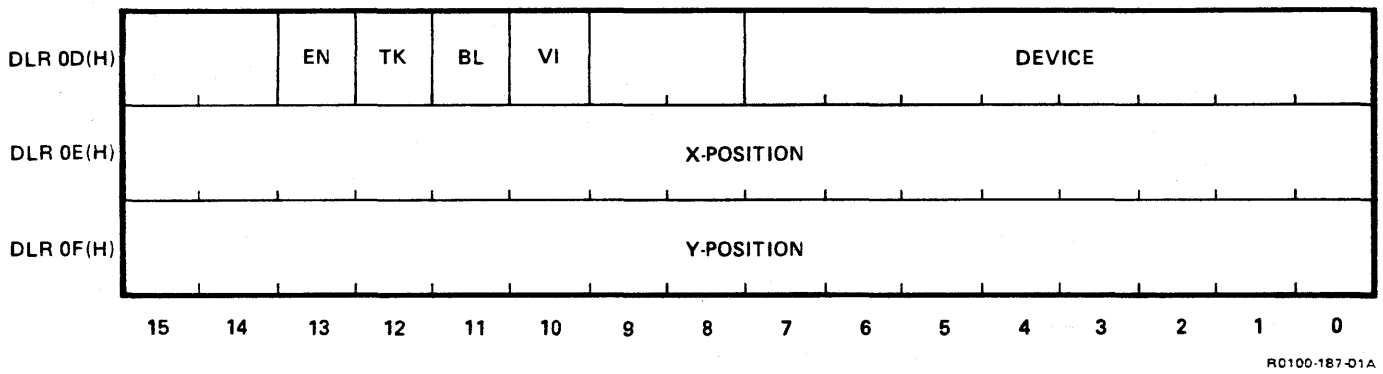
| Parameters | Description |
|---|---|
| DEVICE | Defines the keyboard device number from which this local function association is deleted. |
| A | Defines whether one or all associations with this keyboard are deleted from the local function table. If A = 0, only the association with the specified character code is deleted. If A = 1, all associations with this keyboard are deleted. |
| CHARACTER CODE | If A = 0, this defines the character code association deleted from the local function table. |

Errors

None

| Parameters | Description |
|---|---|
| DISPLAY LIST | Defines the display list that contains the RM-9460 or Z80 code to implement this local function. DISPLAY LIST can take on values 0 through 1F(H) for standard display lists and FF(H) for the COMMON display list. |
| DISPLAY LIST ADDRESS | Defines the display-list address where the code to implement this local function is based. |

Whenever a cursor controller function is enabled and invoked, the following information is placed in display list registers 0D(H), 0E(H), 0F(H):

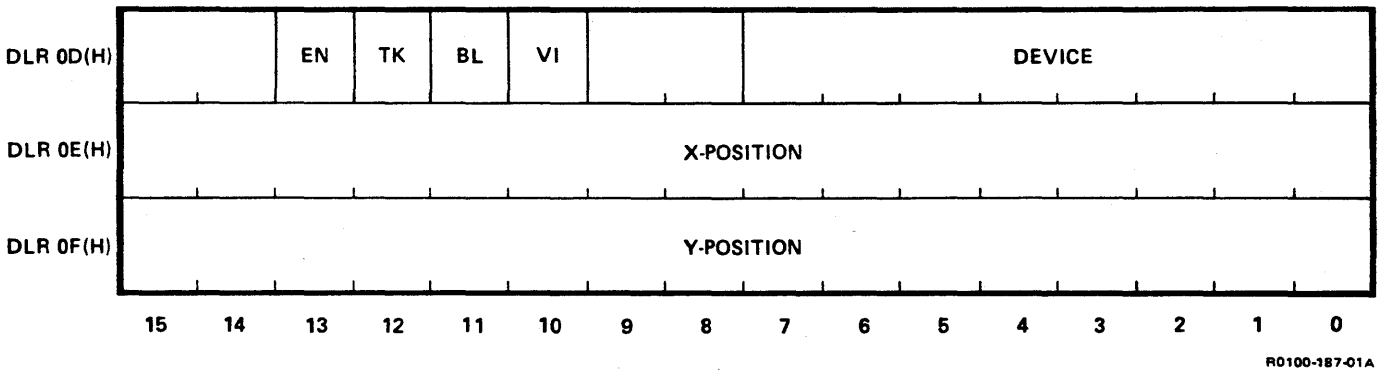



EN, TK, BL, VI are the cursor bits, and X- and Y-positions are the cursor's screen coordinates, taking into account video origin. The new association defined in this instruction overrides any previous association that has been made.

| Errors | Description |
|---|---|
| Code = 7403 | PROGRAMMABLE FONT does not exist. |
| Code = 7405 | CONTEXT does not exist. |
| Code = 740C | DISPLAY LIST number out of range. |
| Code = 740D | DISPLAY LIST does not exist. |
| Code = 7412 | PROGRAMMABLE FONT number out of range. |
| Code = 7416 | CONTEXT number out of range. |
| Code = 7433 | No room in local function table. |
| Code = 744D | Illegal COMMON display list address in extended mode. |
| Code = 744E | Illegal normal display list address in extended mode. |

| Parameters | Description |
|---|---|
| DISPLAY LIST | Defines the display list that contains the RM-9460 code to implement this local function. DISPLAY LIST can take on values 0 through FE(H) for standard display lists and FF(H) for the COMMON display list. |
| DISPLAY LIST ADDRESS | Defines the display-list address where the code to implement this local function is based. |

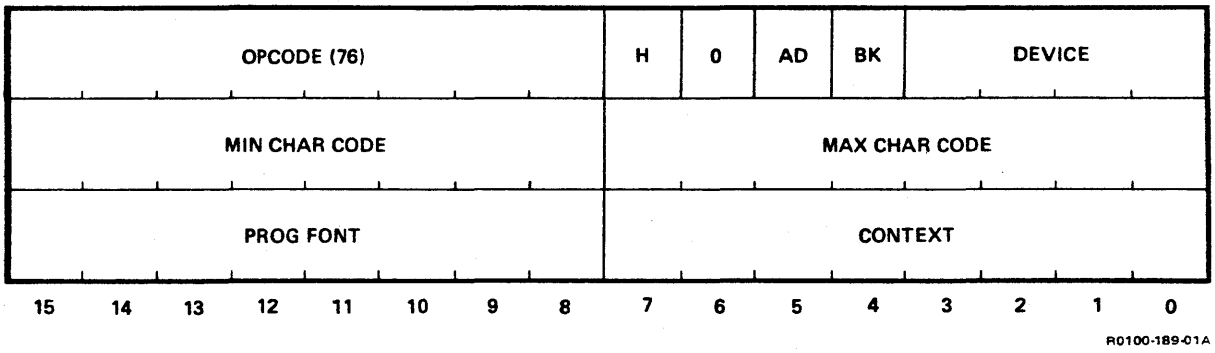
Whenever a cursor controller function is enabled and invoked, the following information is placed in display list registers 0D(H), 0E(H), 0F(H):




EN, TK, BL, VI are the cursor bits, and X- and Y-positions are the cursor's screen coordinates, taking into account video origin. The new association defined in this instruction overrides any previous association that has been made.

| Errors | Description |
|---|---|
| Code = AE03 | PROGRAMMABLE FONT does not exist. |
| Code = AE05 | CONTEXT does not exist. |
| Code = AE0D | DISPLAY LIST does not exist. |
| Code = AE12 | PROGRAMMABLE FONT number out of range. |
| Code = AE16 | CONTEXT number out of range. |
| Code = AE33 | No room in local function table. |
| Code = AE4D | Illegal COMMON display list address in extended mode. |
| Code = AE4E | Illegal normal display list address in extended mode. |
| Code = AE4F | Illegal display list mode for instruction to operate. |

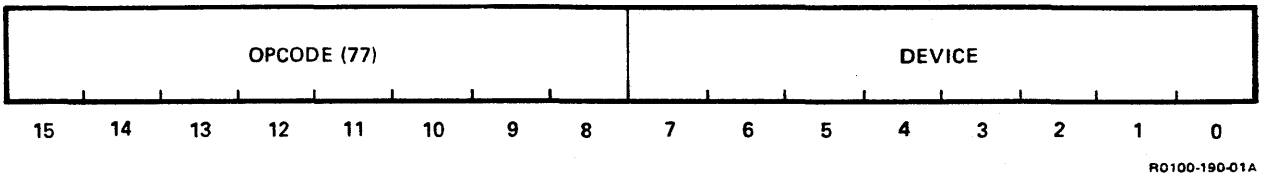
## SET KEYBOARD ECHO STATE (SKES)




SKES is a special-format instruction which causes the RM-9460 to enter a local echo mode where subsequent characters typed on the specified keyboard device are written into refresh memory and to the associated video display. Whenever a character code within the specified range is received from the keyboard, the specified context and programmable font are made current; a WRITE TEXT instruction is executed; the previous context and programmable font are made current; and the host computer instruction execution continues.

| Parameters | Description |
|---|---|
| DEVICE | Defines the keyboard device number from which characters are echoed. |
| H | Specifies whether to notify the host computer when a character is echoed. If H = 0, the host is not notified and the character is not available for host readback. If H = 1, the host is notified the same way as any other character (RCV status bit in interface set). The character is inserted in the keyboard buffer and becomes available for readback using the READ KEYBOARD (RKB) instruction. The characters will not be echoed when there is a data buffer overflow. |
| AD, BK | Define the data generation mode used in the WRITE TEXT instruction. |
| MIN CHAR CODE<br>MAX CHAR CODE | Define a range of character codes that are to be echoed. Any typed character that falls within this range is echoed and no other action is taken (except as determined by H). This means that if there is a local function association whose character falls within this range (when the character is typed), it is echoed and the local function is not executed. (See figure C-10 for the list of character codes.) |

## CLEAR KEYBOARD ECHO STATE (CKES)

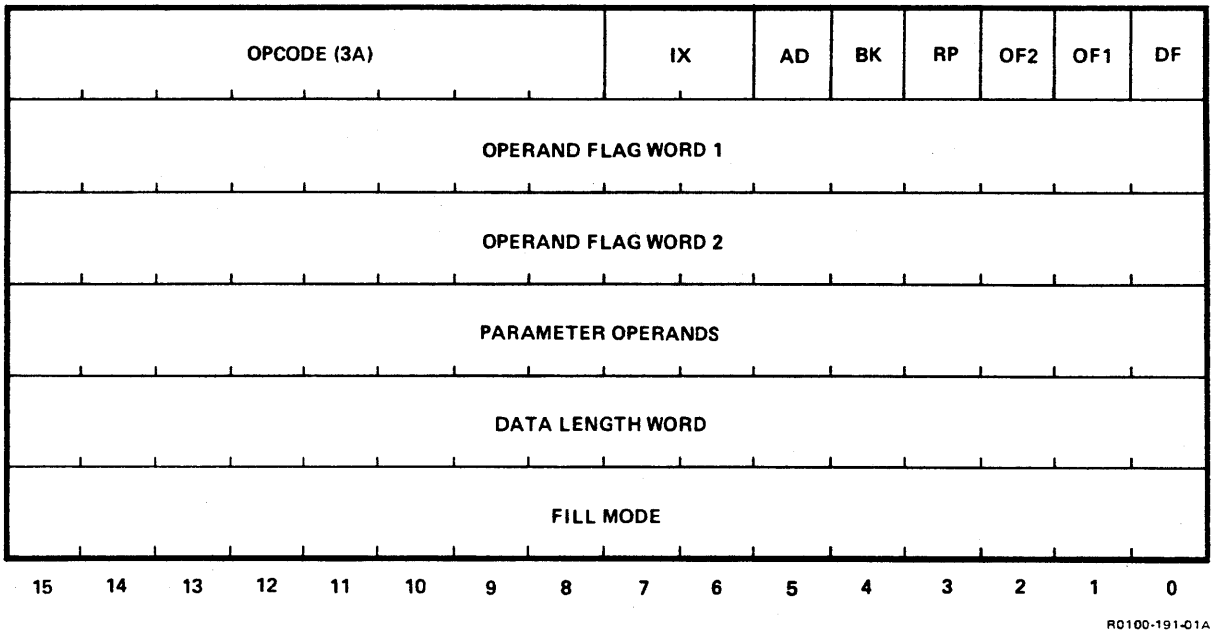



CKES is a special-format instruction which causes the RM-9460 to exit the local echo mode for the specified keyboard. All subsequent characters typed on this keyboard are examined for local function associations and handled appropriately, but none are echoed until another SKES instruction is executed.

### Errors

None

## FILL (FILL)




FILL is a normal-format instruction which writes data into any irregular polygonal region using one of two instruction-selectable algorithms. The first word of data defines the fill algorithm (MODE) to be used, and all subsequent data is discarded. All filling begins at the current operating point (COP) and proceeds to horizontally or vertically adjacent pixels.

When FILL-WHILE mode is selected, all adjacent pixels with the same refresh memory value as the pixel addressed by the COP are filled. The fill value is selected by the BK flag such that if BK = 0, the fill value is the FOREGROUND, while if BK = 1, the fill value is the BACKGROUND.

When FILL-UNTIL mode is selected, the COP and all adjacent pixels are filled until the specified boundary value is encountered. If the COP pixel value matches the boundary value, then no filling will occur. The fill value and boundary value are selected by the BK flag such that if BK = 0, the FOREGROUND is the fill value and the BACKGROUND is the boundary value; if BK = 1, the BACKGROUND is the fill value and the FOREGROUND is the boundary value.

In either algorithm data must be read back from refresh memory to perform conditional testing for boundary conditions. The READ MASK parameter controls this readback.

| Parameters | Description |
|---|---|
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DATA LENGTH WORD | When $\geq 2$, indicates that the FILL MODE word is the first word of the data stream. All subsequent data is discarded. |

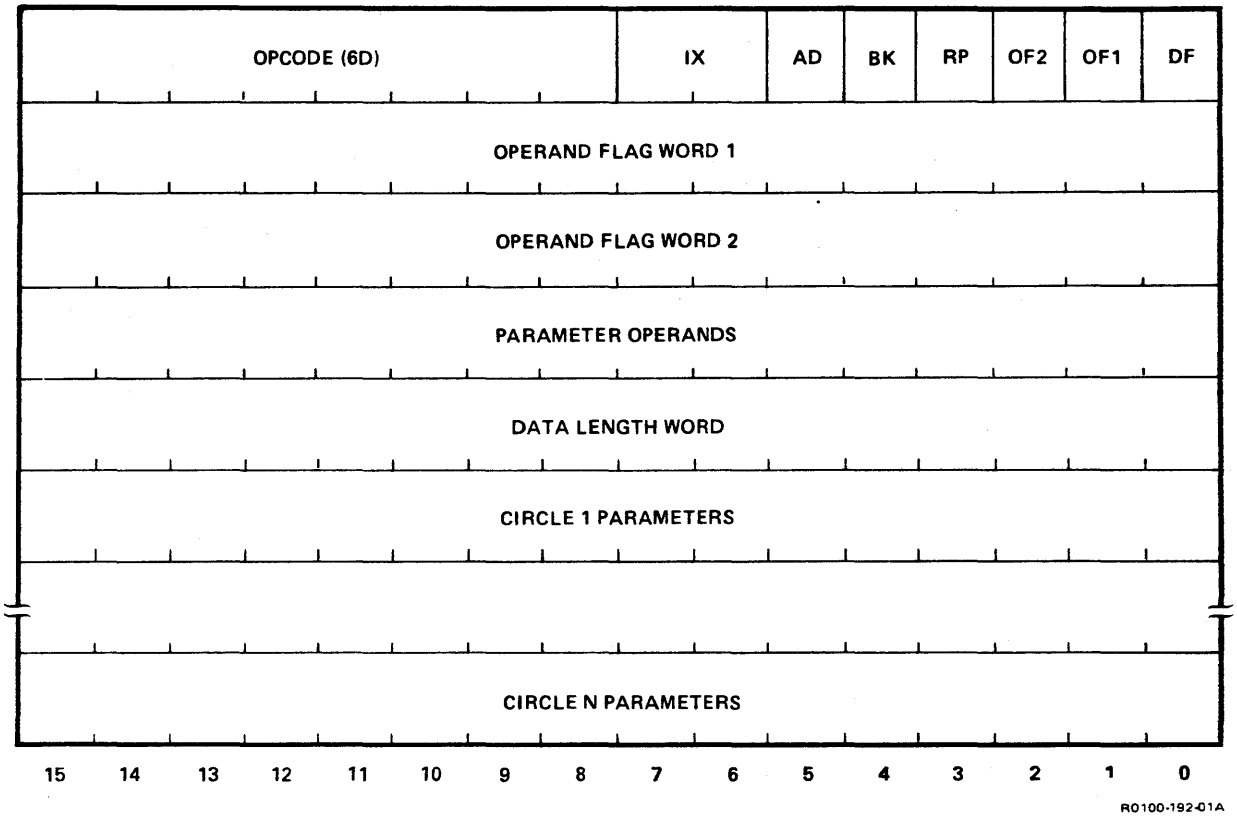
Data Format

The first word of data is used as the FILL MODE algorithm selector. All data following this word is discarded. A FILL MODE value of 0 will cause FILL-WHILE. A FILL MODE value of 1 will cause FILL-UNTIL. Any other values will generate an error and no fill actions will be performed.

| | |
|---|---|
| FILL MODE | When FILL MODE = 0 FILL-WHILE<br>When FILL MODE = 1 FILL-UNTIL |

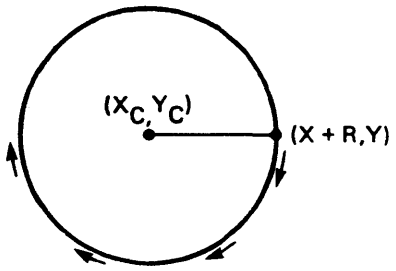
| Errors | Description |
|---|---|
| Code = 3A07 | Illegal byte count. |
| Code = 3A17 | No MCPs selected. |
| Code = 3A18 | More than one MCP selected. |
| Code = 3A19 | No memory group selected. |
| Code = 3A1A | More than one memory group selected. |
| Code = 3A39 | Polygon too complex. Out of room to save pending fill operations. |
| Code = 3A3A | Illegal fill parameter (not logical 0 or logical 1). |
| Code = 3A41 | Fill boundary not found when using FILL-UNTIL. |
| Code = 3A42 | When using FILL-WHILE: WRITE MASK .AND. READ MASK = 0 (WRITE MASK and READ MASK do not overlap), or WRITE MASK .AND. READ MASK .AND. fill value = match value (COP pixel value). |
| Code = 3A43 | WRITE MASK = 0. |

## WRITE CIRCLE (CIRC)

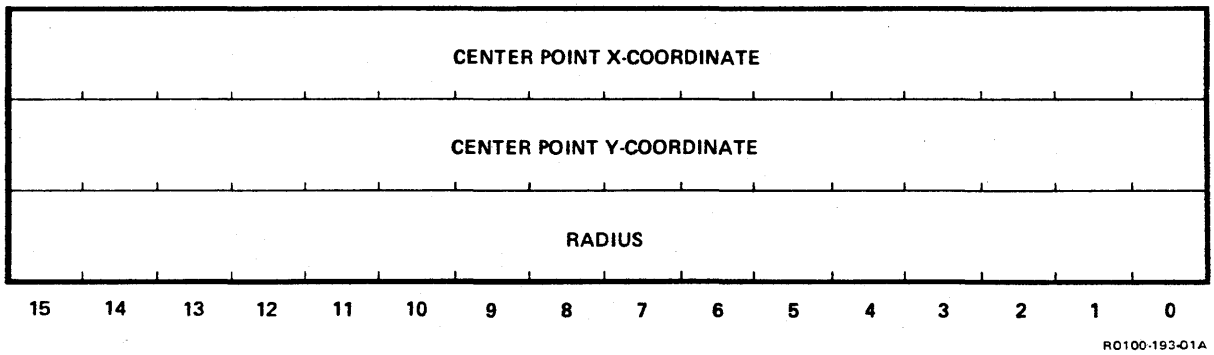



CIRC is a normal-format instruction which draws circles with the center coordinates and radius values specified in the instruction. The circle is adjusted for screen resolution so that it appears as a circle; however, it may not be a true circle in refresh memory.

The current addressing mode and coordinate transformations are applied to the circles generated by this instruction. Any number of circles may be drawn with one instruction. When the origin is set to upper left, all circles are drawn clockwise. When the instruction is complete, the current operating point (COP) is located at the center of the last circle written.



## Data Format




| | |
|---|---|
| CENTER POINT | Defines the X-Y coordinates of the center of the circle to be generated. The center point is defined by using the current addressing mode and current coordinate transformations. |
| RADIUS | Defines the radius of the circle to be drawn, as a distance in the X direction, using current coordinate transformations. |

| Errors | Description |
|---|---|
| Code = 6D07 | Illegal value for DATA LENGTH WORD; not a multiple of six. |
| Code = 6D35 | Negative radius value specified. |

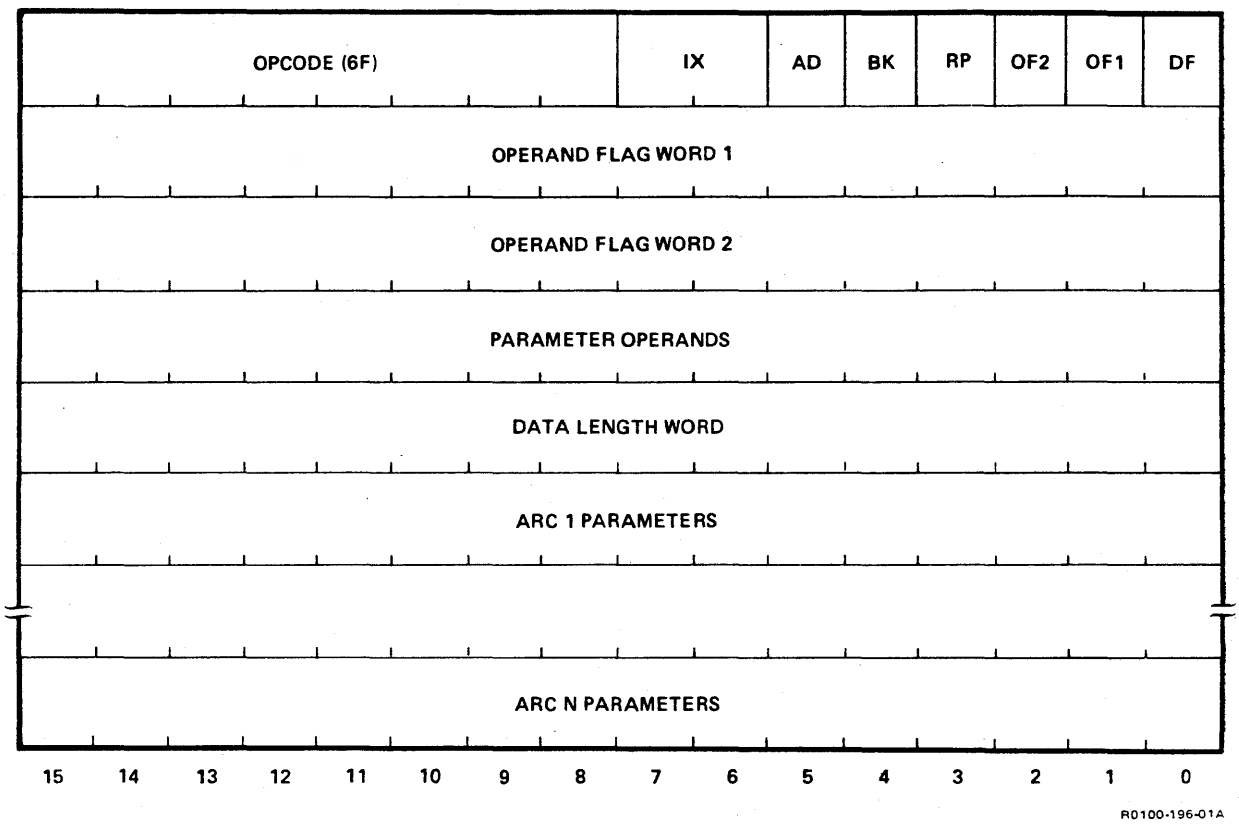
## COP Movement

The COP after executing the circle instruction is the coordinate of the center of the last circle drawn.

| Flags | Description |
|---|---|
| IX | Defines the addressing mode used to evaluate the INDEX 1 and INDEX 2 parameter operands and the three data points used to define the arc. |
| BK | Defines the selection of FOREGROUND or BACKGROUND values for the color or intensity of the arc being drawn. If BK = 0, the arc is generated with the FORE-GROUND value; otherwise, the BACKGROUND value is used. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled. |
| FOREGROUND | Defines the color or intensity value of the arc when BK = 0. |
| BACKGROUND | Defines the color or intensity value of the arc when BK = 1. |
| INDEX 1 | Displaces the values to be used for the INDEX 2 parameter operand and the three data points on the arc when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for the three data points on the arc when IX = 10(B). |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, writing into refresh memory is enabled; otherwise, writing into refresh memory is disabled. |
| DATA LENGTH WORD | Defines the total number of bytes contained in the instruction following the DATA LENGTH WORD. Any number of arcs may be drawn with one instruction. Twelve bytes of data are needed to define an arc of type 1, so the DATA LENGTH WORD must be a multiple of twelve. |

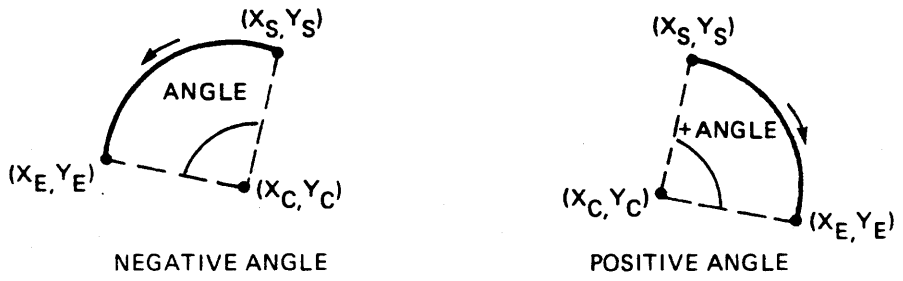
## WRITE ARC TYPE 2 (ARC2)



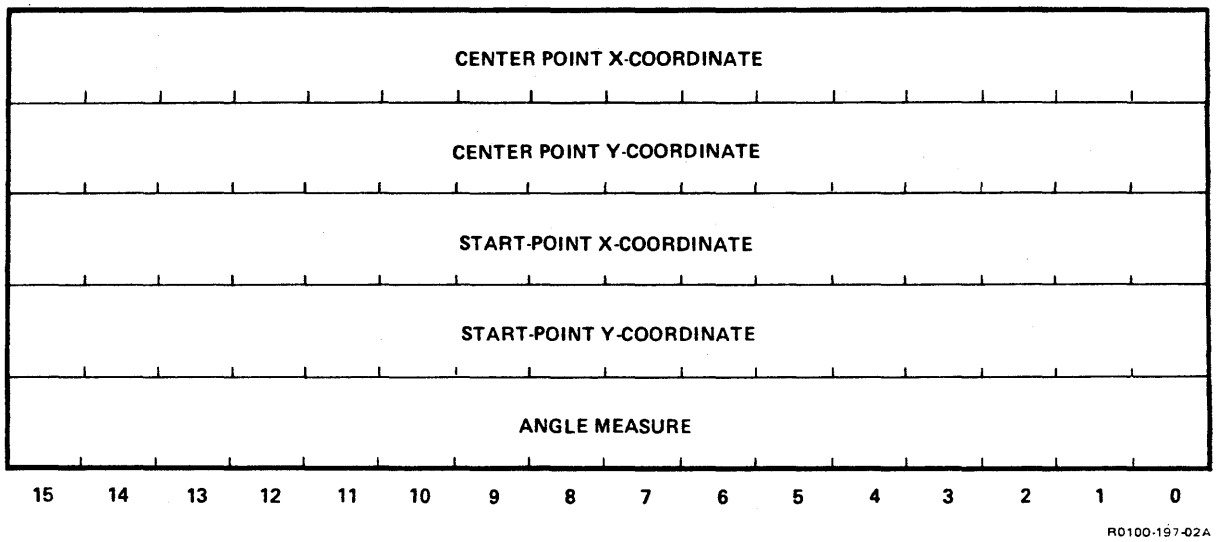

ARC2 is a normal-format instruction which draws circular arcs with a given degree measure. The center, startpoint, and degree measure of the arc are specified in the instruction. Any number of arcs may be drawn with one instruction.

The current addressing mode and coordinate transformations are applied to each coordinate before an arc is generated. At the end of the instruction, the COP is located at the endpoint of the last arc drawn.

As shown in the angles below, given center point $(X_C,Y_C)$ and startpoint $(X_S,Y_S)$ and an arc measure, the arc with the given degree measure is drawn, with the COP ending at $(X_E,Y_E)$.



## Data Format




| | |
|---|---|
| CENTER POINT X-Y | Coordinate of the center of the circle on which the COORDINATE arc lies. |
| START-POINT X-Y COORDINATE | Coordinate of the startpoint of the arc. |
| ANGLE MEASURE | The degree measure of the arc as a 16-bit two's-complement integer. |

| Errors | Description |
|---|---|
| Code = 6F07 | Illegal value for DATA LENGTH WORD; not a multiple of ten. |

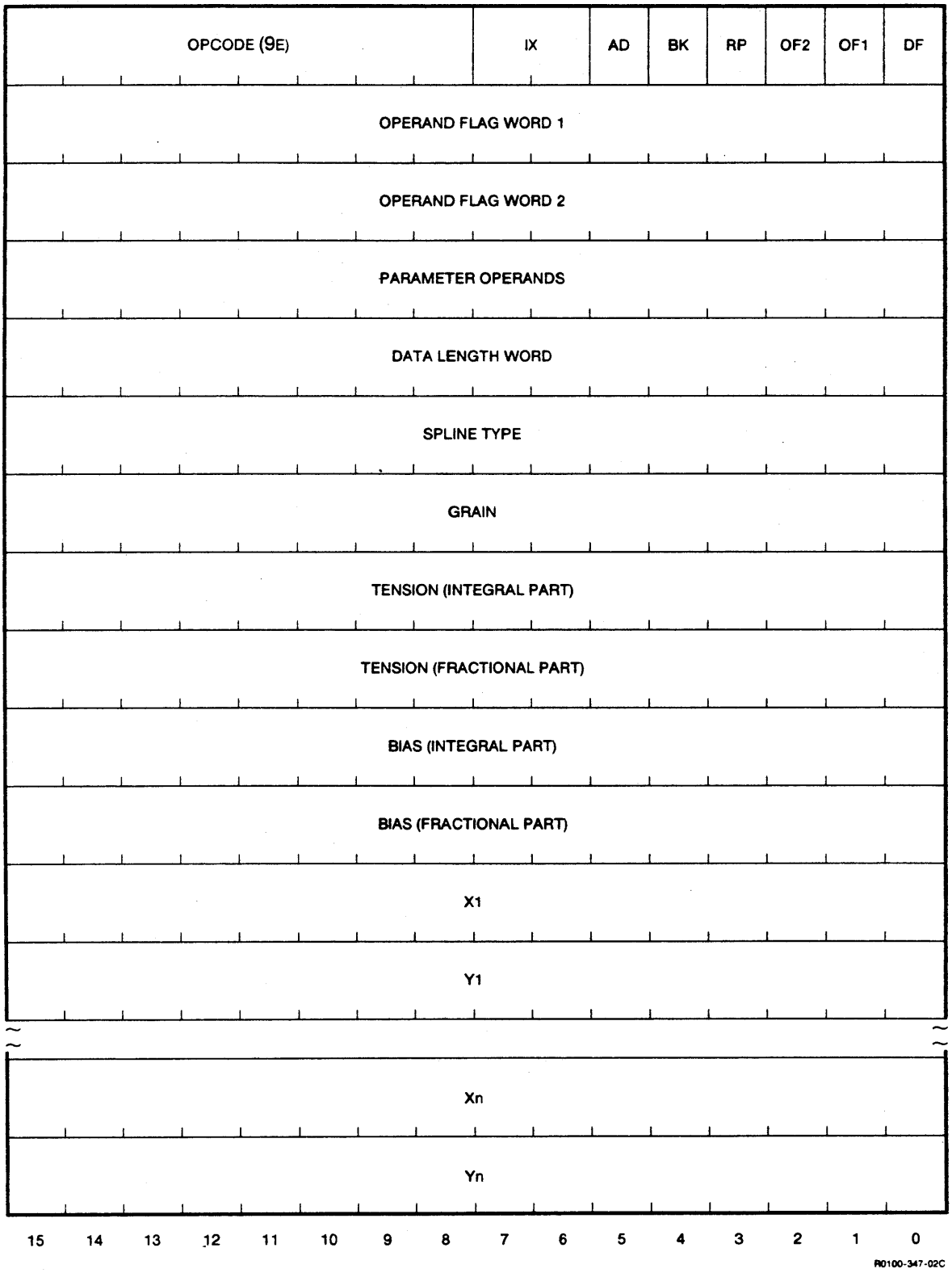
## COP Movement

The COP after execution of the ARC TYPE 2 instruction is at the endpoint of the last arc drawn.

| Flags | Description |
|---|---|
| IX | Defines the addressing mode used to evaluate the INDEX 1 and INDEX 2 parameter operands and the given start, endpoints, and radius value used to draw the arc. |
| BK | Defines the selection of FOREGROUND or BACKGROUND values for the color or intensity of the arc being drawn. If BK = 0, the arc is generated with the FOREGROUND value; otherwise, the BACKGROUND value is used. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled. |
| FOREGROUND | Defines the color or intensity value of the arc when BK = 0. |
| BACKGROUND | Defines the color or intensity value of the arc when BK = 1. |
| INDEX 1 | Displaces the values to be used for the INDEX 2 parameter operand and the start, endpoints, and radius value of the arc when IX = 01(B). |
| INDEX 2 | Displaces the values to be used for the start, endpoints, and radius value of the arc when IX = 10(B). |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, writing into refresh memory is enabled; otherwise, writing into refresh memory is disabled. |
| DATA LENGTH WORD | Defines the total number of bytes contained in the instruction following the DATA LENGTH WORD. Any number of arcs may be drawn with one instruction. Ten bytes of data are needed to define an arc of type 3, so the DATA LENGTH WORD must be a multiple of ten. |

## WRITE SPLINE (WS)



## WRITE COLOR INTERPOLATED SPLINE (WCIS)

| OPCODE (9F) | IX | AD | BK | RP | OF2 | OF1 | DF |
|---|---|---|---|---|---|---|---|
| OPERAND FLAG WORD 1 | | | | | | | |
| OPERAND FLAG WORD 2 | | | | | | | |
| PARAMETER OPERANDS | | | | | | | |
| DATA LENGTH WORD | | | | | | | |
| SPLINE TYPE | | | | | | | |
| GRAIN | | | | | | | |
| TENSION (INTEGRAL PART) | | | | | | | |
| TENSION (FRACTIONAL PART) | | | | | | | |
| BIAS (INTEGRAL PART) | | | | | | | |
| BIAS (FRACTIONAL PART) | | | | | | | |
| X1 | | | | | | | |
| Y1 | | | | | | | |
| PIXEL VALUE 1 | | | | | | | |
| (continued on next page) | | | | | | | |

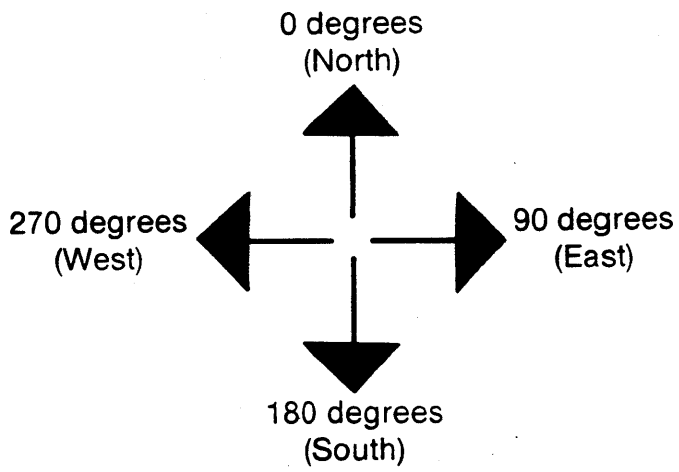
R0100-348#1-02C

| Parameters | Description |
|---|---|
| BIAS | Affects the shape of Beta spline curves by shifting the segments between the knots so that the curve is skewed toward one of the knots on either side. The bias parameter is a two-word parameter. The first word is the integral part; the second word is the fractional part. A value of 1.0 (0 X 00010000) is a reasonable default. |
| X, Y | Represent the knot coordinates. |
| PIXEL VALUE | Represents a binary pixel value that indexes a particular color in the video lookup table. |

| Errors | Description |
|---|---|
| Code = 9F07 | Illegal DATA BYTE WORD. |
| Code = 9F4C | Illegal sub function code; SPLINE TYPE must be 0 or 1. |
| Code = 9F50 | GRAIN exceeds 4096. |

AP is a special-format instruction which allows the user to draw a series of arrows, presenting wind data. For each arrow, the user specifies the position of the base of the shaft, pixel value, direction, length, and altitude.

| Parameters | Description |
|---|---|
| NUMBER OF ARROWS | Defines the number of different arrows to be drawn. |
| X, Y COORDINATE | Defines the coordinates of the base of the shaft in local coordinates. |
| PIXEL VALUE | Defines the pixel value with which the arrow is to be drawn. |
| DIRECTION | Specifies the direction toward which the arrow is pointing in one degree increments, with 0 degrees specifying north. |



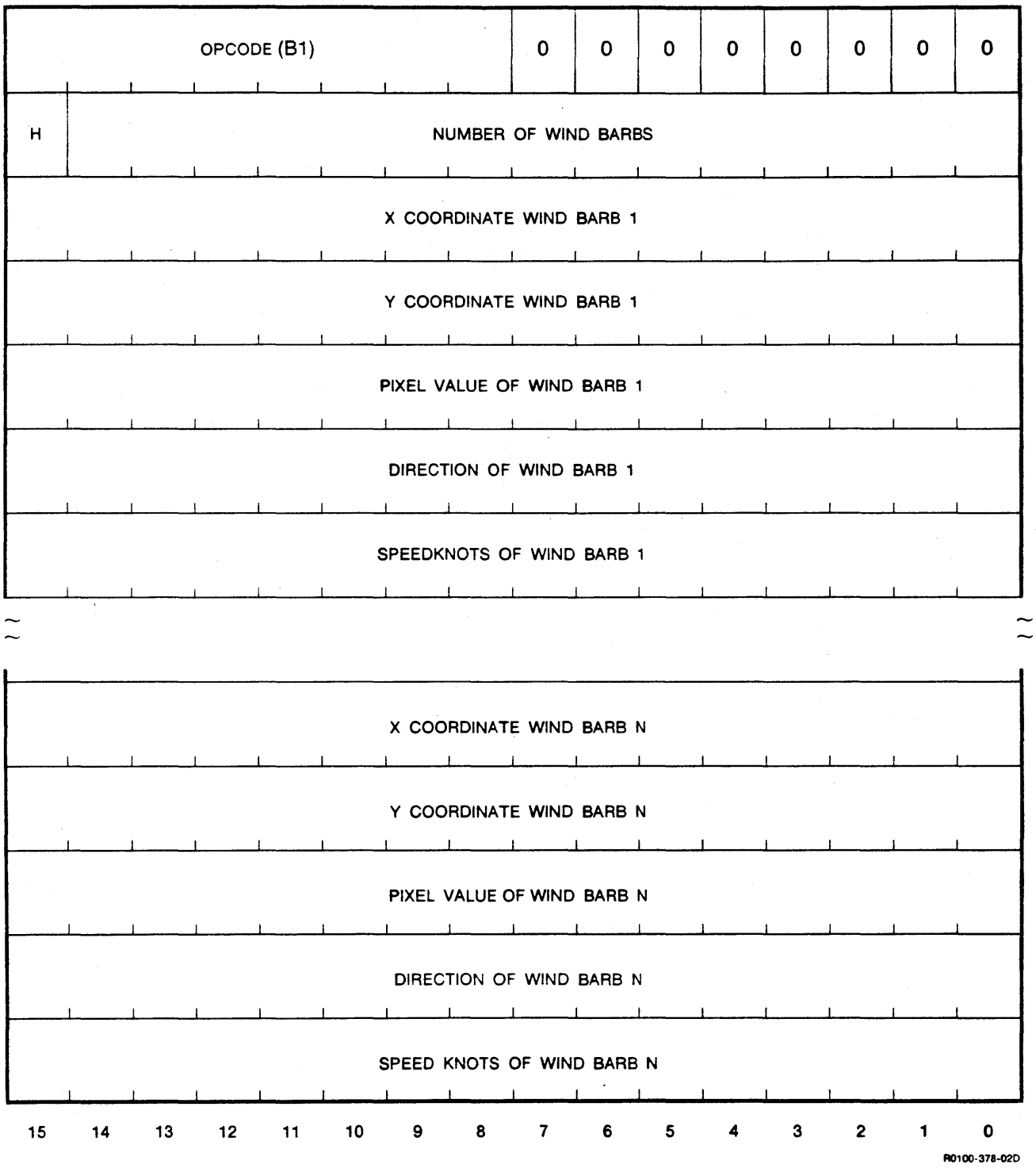

X0100-394-02D

| | |
|---|---|
| LENGTH | Defines the total length in pixels from the base of the shaft to the head of the arrow. |
| ALTITUDE | Defines the length in pixels from the base to the tip of the triangle at the head of the arrow. The width at the base of the arrow will be twice the altitude. |

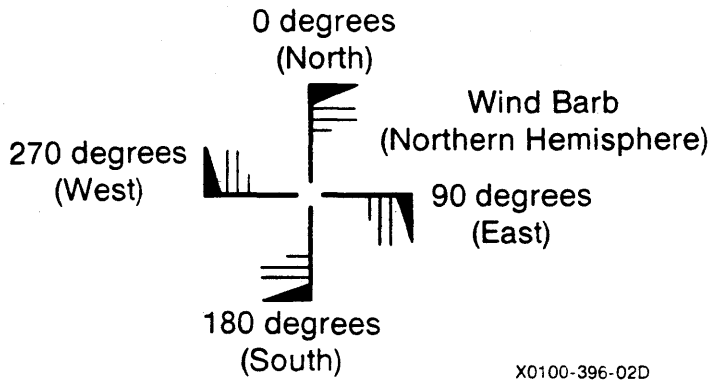
Errors

None.

## WIND BARB (WB)




WB is a special-format instruction which allows the user to draw a series of wind barbs, each specifying wind speed and direction. For each wind barb, the user specifies the position of the base of the shaft, pixel value, direction, and wind speed.




| Parameters | Description |
|---|---|
| SPEEDKNOTS | Defines the magnitude of the wind speed in whole knots. |

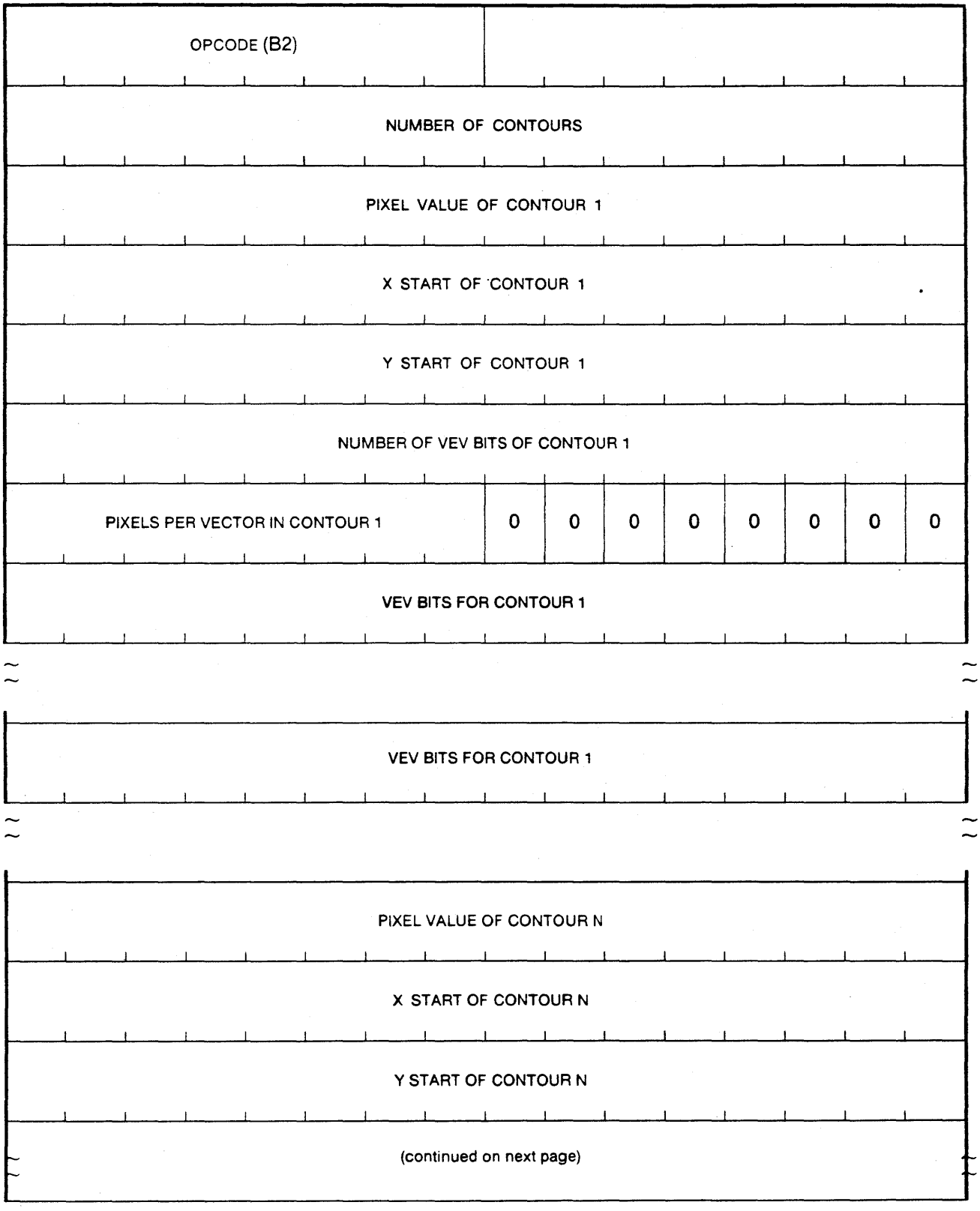
Errors

None.

Example:

| Hex Code | Description |
|---|---|
| B100 | ;WB instruction |
| 8001 | ;draw one wind barb in the Southern Hemisphere |
| 0064 | ;X coordinate |
| 0064 | ;Y coordinate |
| 0005 | ;pixel value |
| 005A | ;wind direction (90 degrees) |
| 004C | ;wind speed (77 knots) |

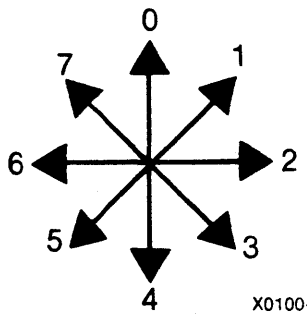
## VARIABLE EXCEPTION VECTOR (VEV)




R0100-379#1-02D

| Parameters | Description |
|---|---|
| VEV BITS | Determines the direction in which the contour is to proceed, and whether or not a vector is to be drawn. The VEV BITS are processed in nibbles from left to right in each 16-bit word. Each nibble gives a value from 0 - 16, with 0 - 8 being the draw values and 9 - 15 being the move values. The directions are shown in the following direction matrix. The directions and whether to draw or move are given in table 2-12. |

Direction Matrix




X0100-399-02D

Table 2-12. VEV Bit Value Designations

| VEV Bits | Decimal Value | Action | Direction |
|---|---|---|---|
| 0000 | 0 | Draw | Up |
| 0001 | 1 | Draw | Up and right |
| 0010 | 2 | Draw | Right |
| 0011 | 3 | Draw | Down and right |
| 0100 | 4 | Draw | Down |
| 0101 | 5 | Draw | Down and left |
| 0110 | 6 | Draw | Left |
| 0111 | 7 | Draw | Up and left |
| 1000 | 8 | Move | Up |
| 1001 | 9 | Move | Up and right |
| 1010 | 10 | Move | Right |
| 1011 | 11 | Move | Down and right |
| 1100 | 12 | Move | Down |
| 1101 | 13 | Move | Down and left |
| 1110 | 14 | Move | Left |
| 1111 | 15 | Move | Up and left |

## 2.9 TRENDING

The RM-9460 trending firmware allows the user to construct multiline trend displays. A maximum of 16 trends may be allocated and displayed at any one time with up to 12,000 data points per trend display. When a trend is allocated, the number of points per line and the number of lines are specified. Each line will contain the same number of points. The total space required must be less than 12,000 words (or 24,000 bytes).

The trend RAM is remapped during the instruction execution, overlaying the display list and programmable font RAM area. For this reason, trend instruction cannot be executed from a display list. The RAM is remapped when the instruction is complete.

Trend data bases may be initialized, updated, displayed, and erased. The data base is displayed on a fixed axis defined by the user in the display instruction. Trends may be generated in one of four progression directions, left-to-right, right-to-left, top-to-bottom, and bottom-to-top. Each line in a trend may be assigned a unique pattern and/or a unique color to provide separation and highlighting.

Trends may be displayed with complete control over starting point, quantity of points to display, and how far to move between points. This allows small deltas and all points displayed to get the big picture, and the larger deltas and fewer points displayed to examine a particular area of interest.

The 16 trends that may be allocated are referenced by numbers 0 to 15. For a trend containing N lines, the lines are numbered from 0 to N-1. For a trend containing M points, the points are referenced by numbers in the range 0 to M-1.

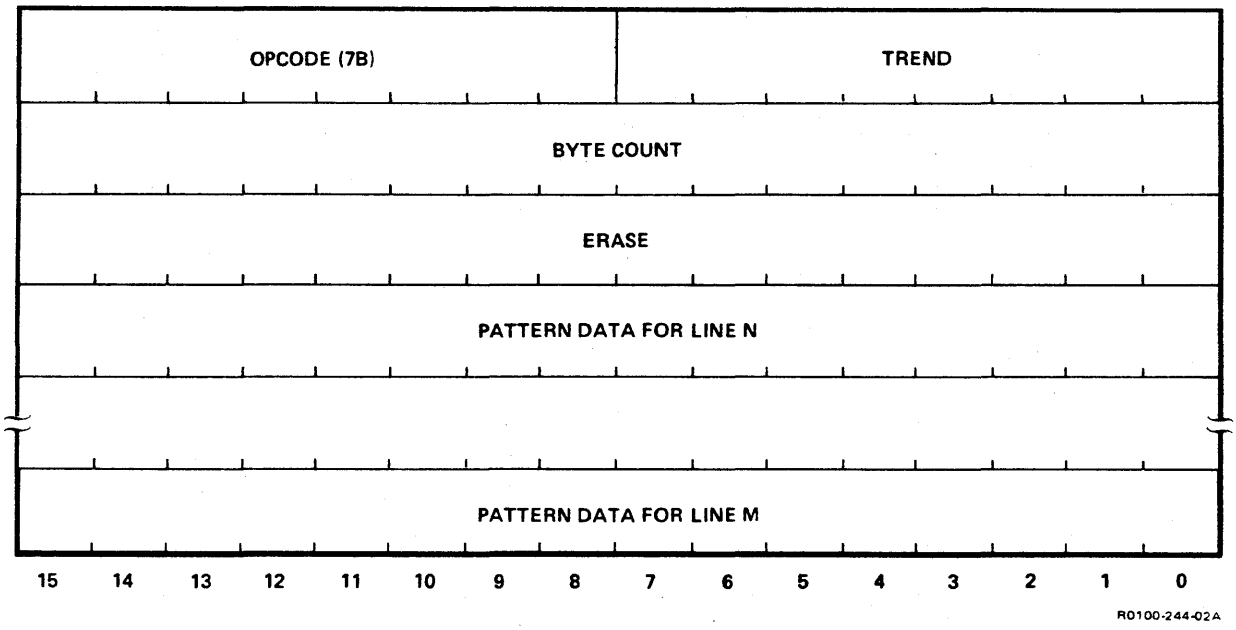
Trending is part of the standard firmware with the MC68000 system processor and is an option on the processor expansion PCB with the Z80 system processor.

Instructions contained in the trending firmware are listed below:

- Allocate Trend (ALTD)
- Init Trend (ITD)
- Load Trend Patterns (LTDP)
- Display Trend (DISTD)
- Erase Trend (ERSTD)
- Update Trend (UDTD)
- Deallocate Trend (DETD)

| Errors | Description |
|---|---|
| Code = 7802 | No RAM blocks available. |
| Code = 780E | TREND already exists. |
| Code = 7823 | Attempt to set up a trend with more than 12,000 trend points or more than 255 lines. |
| Code = 7824 | Illegal TREND number. |

## LOAD TREND PATTERNS (LTDP)




LTDP is a special-format instruction which defines the patterns and values used to draw the lines of the specified trend. This allows highlighting or separation of lines when many are drawn close together. One or all of the lines may be assigned patterns, and there is no restriction on the order in which they are specified. If desired, only the ERASE value can be defined. LTDP is part of the standard firmware for an MC68000-based RM-9460, and is an option for a Z80-based RM-9460.

| Parameters | Description |
|---|---|
| BYTE COUNT | Describes how much data is included in this instruction. If only ERASE is to be defined, then BYTE COUNT = 2. If, however, one or more pattern data blocks are included, the byte count must equal BC = 2 + (number of pattern data blocks) * 10. |
| ERASE | Defines the value used to erase all lines in this trend before an update redisplay is performed. |

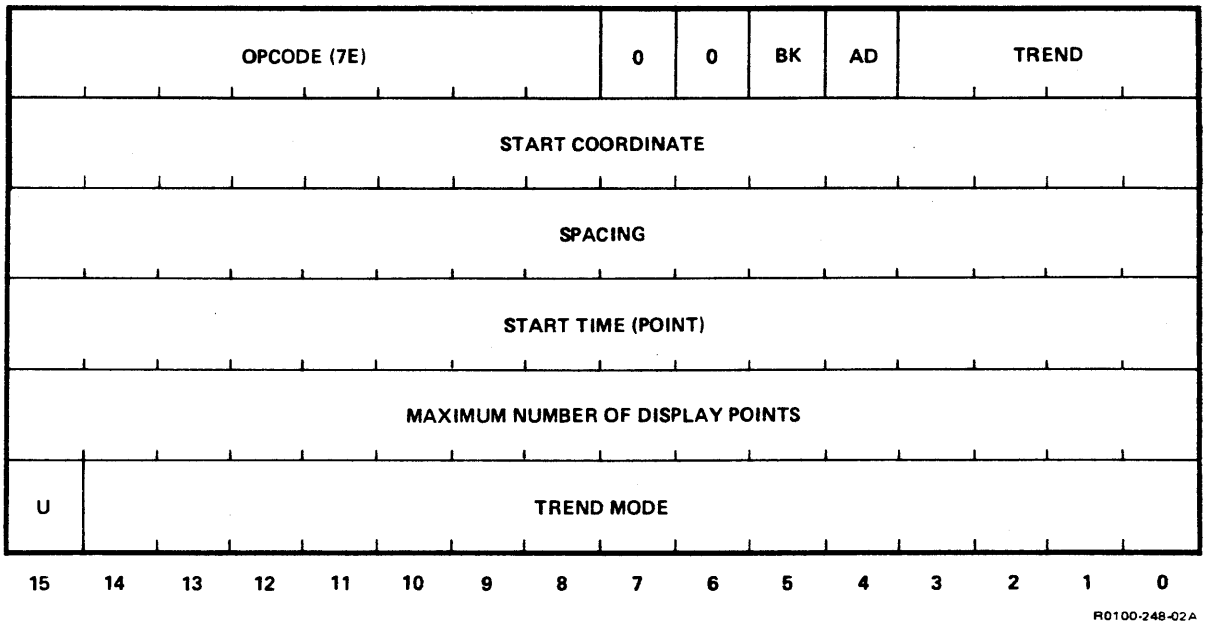
| Errors | Description |
|---|---|
| Code = 7B07 | Illegal BYTE COUNT. All data will be discarded and an illegal instruction will be generated. |
| Code = 7B24 | Illegal TREND number. |

line is discarded, and one point for each new point is added to the end of the line. UDTD is part of the standard firmware for an MC68000-based RM-9460, and is an option for a Z80-based RM-9460.

If the trend has been redisplayed, the COP is left at the last displayed point and the FOREGROUND, BACKGROUND, and LINE PATTERN parameters remain unchanged.

| Errors | Description |
|---|---|
| Code = 7C07 | Illegal BYTE COUNT. The specified BYTE COUNT will not allow an equal number of new data points to be added to each trend line. The data will be received and discarded, and an illegal instruction interrupt will be generated. |
| Code = 7C24 | Illegal TREND number. |

## DISPLAY TREND (DISTD)




DISTD is a special-format instruction which will cause the lines of the specified trend to be displayed. If a trend display has been generated previously, the old display will be erased before the data is redrawn with new parameters. DISTD is part of the standard firmware for an MC68000-based RM-9460, and is an option for a Z80-based RM-9460.

### Display Flags

Two bits are used to select graphics plotting options:

Bits

| 5 | 4 |
|---|---|
| BK | AD |

BK — Background Flag. Selects between normal and reverse background.

AD — Additive Write Flag. Suppresses writing data bits of logical 0 within the trend pattern.

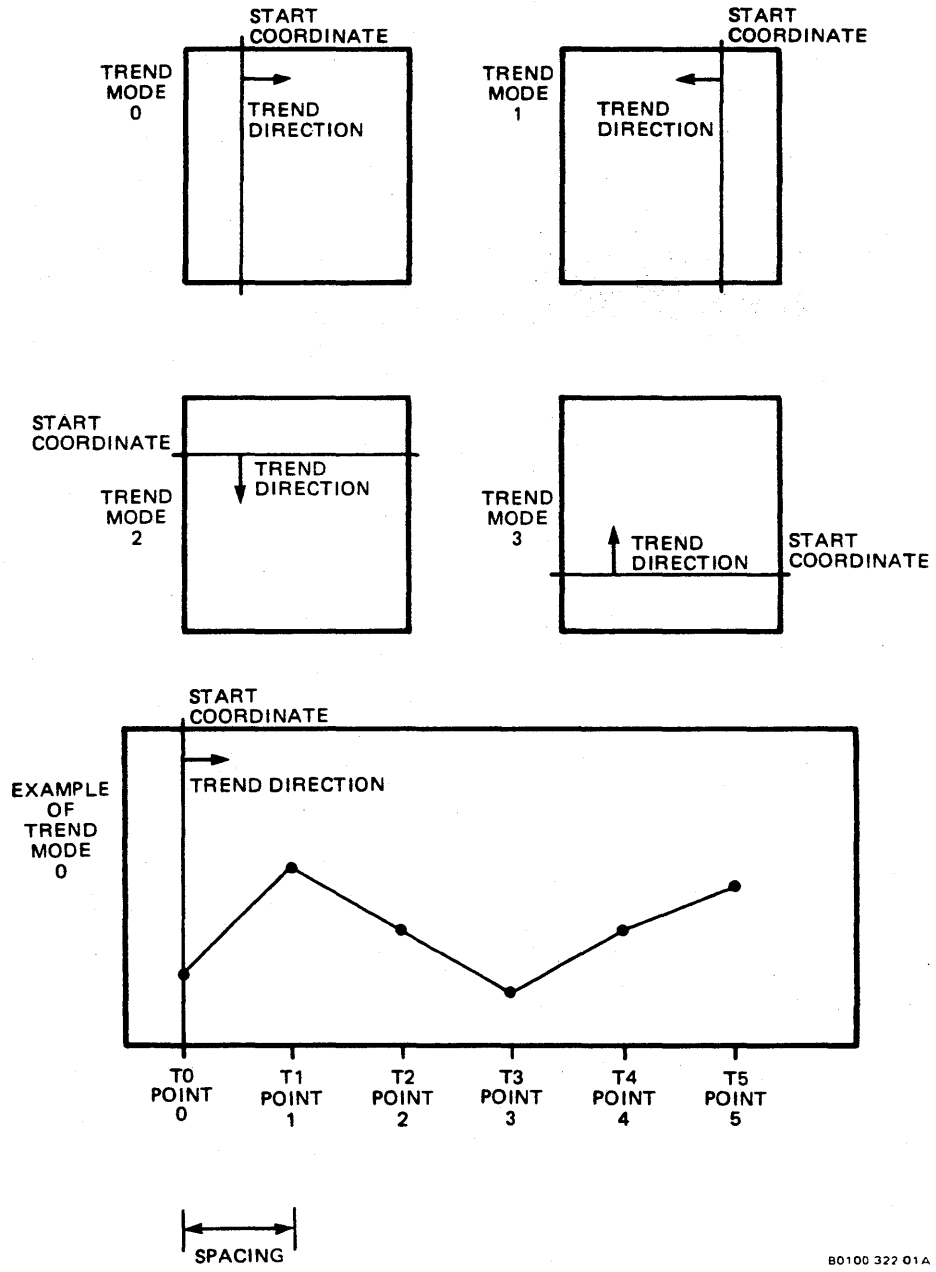



Figure 2-18. Display Trend Parameters

## 2.10 PIXEL FORMATTER

The RM-9460 pixel formatter firmware allows the user to store pictures more compactly and to use the full data path to the RM-9460. For example, when transferring 4-bit image data, four pixels can be contained in each data word sent to the RM-9460. The firmware also allows the user to scale image data by specifying a vector that the data will appear along. This technique can be used to display radial data or to rotate image data so the screen presentation is not restricted to an orthogonal box.

The pixel formatter instructions are listed below:

- Write Packed Image (WPI)
- Write Image Vectors (WIV)

| Flags | Description |
|---|---|
| IX | Defines the address mode in which INDEX 1, INDEX 2, FORMAT WINDOW, and START-POINT are evaluated. |

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes enabled for writing data. It should match the memory slice being updated with the packed image data. |
| INDEX 1 | Displaces the values used for INDEX 2, FORMAT WINDOW, and START-POINT when IX = 01(B). |
| INDEX 2 | Displaces the values used for FORMAT WINDOW and START-POINT when IX = 10(B). |
| FORMAT WINDOW | Defines the rectangular area where image data is written. If START-POINT is not set explicitly in the WPI instruction, FORMAT WINDOW and SCAN define the starting pixel coordinate. |
| SCAN | Defines the primary and secondary update directions as well as the starting pixel coordinate when START-POINT is not defined explicitly in the WPI instruction. |
| START-POINT | Defines the coordinate of the first pixel to be written with image data. If not defined, the starting pixel coordinate is the COP from the last instruction executed. If the START-POINT is not located on the FORMAT WINDOW starting edge of primary scan, then all data will be discarded and an error interrupt generated. |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, then writing to refresh memory is enabled; otherwise, writing into refresh memory is disabled. |

| Errors | Description |
|---|---|
| Code = 7F07 | Illegal byte count; either odd byte or not enough data to fill the last line. |
| Code = 7F30 | Illegal packing parameter; either pixel length less than 2 or greater than 5 or pixel count * pixel length > 16. |
| Code = 7F31 | COP not positioned at the FORMAT WINDOW starting edge of primary scan. |

Example:

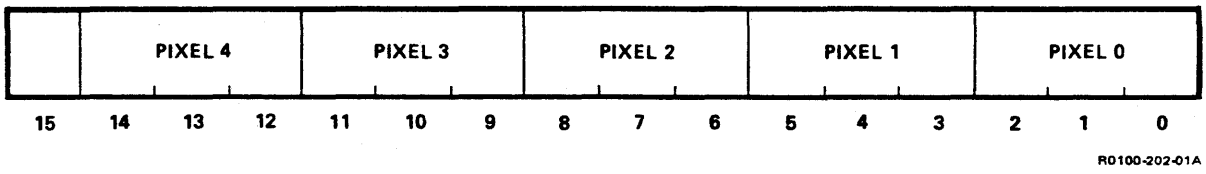
Window width = 30 pixels
Window height = 10 pixels
Pixel length = 3 bits
Pixel count = 5 pixels/data word
Subchannel for data = Refresh memory planes 8-6

30 pixels per line = 6 words per line
5 pixels per data word

6 words per line * 10 lines = 60 words = 120 bytes

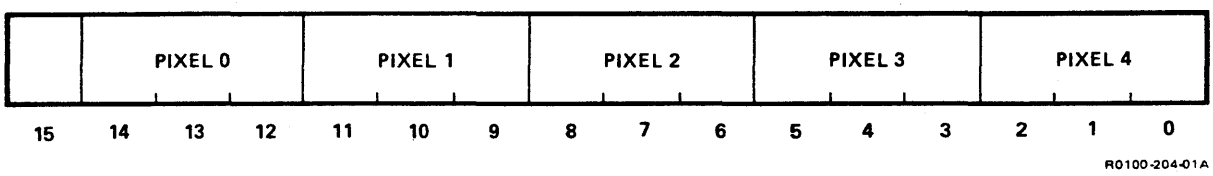
2 bytes per packing descriptor + 120 bytes for data = 122 bytes = data length word = 7A(H)

Data packed in each word starting from low to high according to the following format:



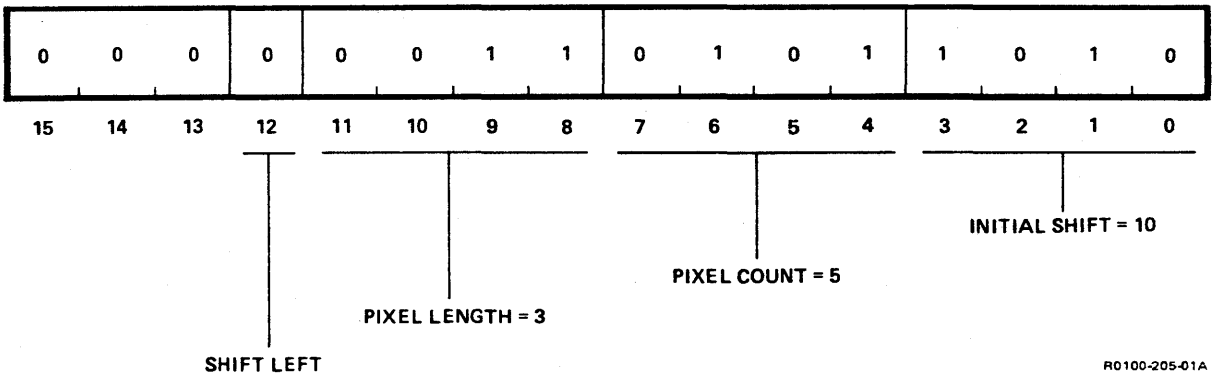

Shift direction = Shift right
Initial shift = 10 (to shift pixel 0 right in order to align with bits 8-6)

If the data was packed in each word starting from high to low, the packing descriptor would be:




Shift direction = Shift left
Initial shift = 10 (to shift pixel 0 left in order to align with bits 8-6)

PACKING DESCRIPTOR




Packing descriptor = 035A

| Parameters | Description |
|---|---|
| WRITE MASK | Defines the refresh memory planes that are write-enabled to receive image vector data. |
| INDEX 1 | Displaces the values to be used for INDEX 2 and each vector's endpoint coordinates if IX = 01(B). |
| INDEX 2 | Displaces the values to be used for each vector's endpoint coordinates if IX = 10(B). |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, writing to refresh memory is enabled; otherwise, writing to refresh memory is disabled. |
| DATA LENGTH WORD | Defines the total number of bytes contained in the WIV instruction following the DATA LENGTH WORD. Since data in the WIV instruction is always interpreted as words, if the DATA LENGTH WORD is odd, an illegal instruction interrupt is generated and all data discarded. A maximum of 65,534 bytes may be transferred with one WIV instruction, but the following equation must be obeyed: |

$$\text{DATA LENGTH WORD} = 4 + \sum_{i=1}^{\text{Vectorcount}} \left[ 5 + \frac{\text{Number of Image Data Values + Mode Adjust 1}}{\text{Mode Adjust 2}} \right] * 2$$

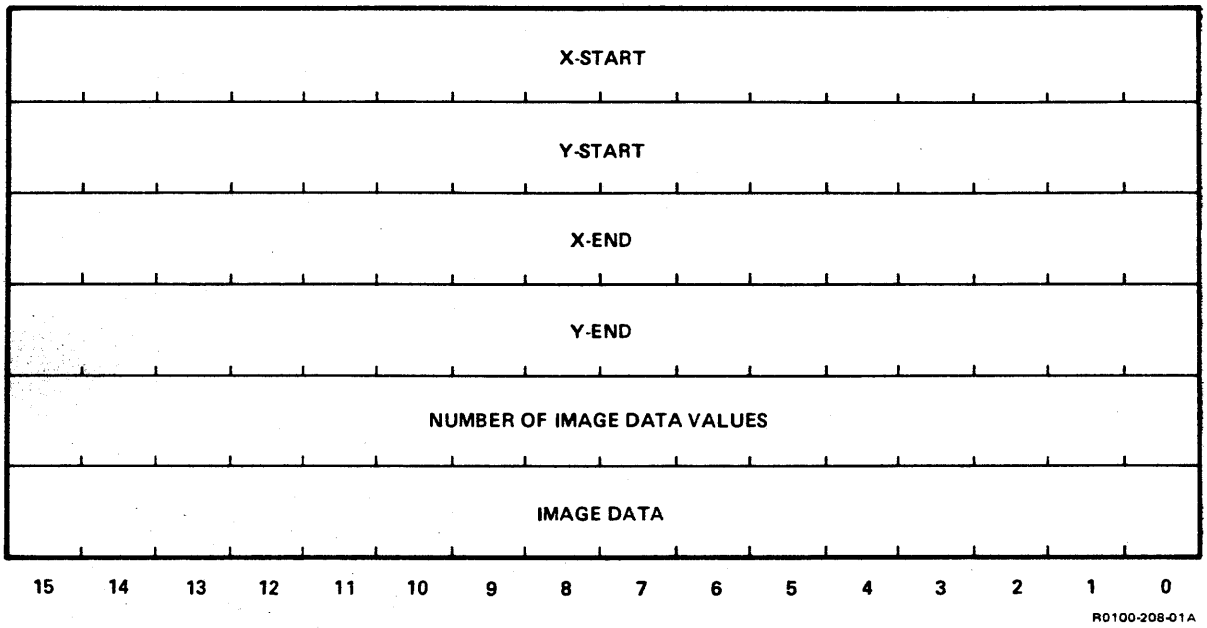
where

| | Word | Byte | Nibble |
|---|---|---|---|
| Mode Adjust 1 | 0 | 1 | 3 |
| Mode Adjust 2 | 1 | 2 | 4 |

B0100-234-01A

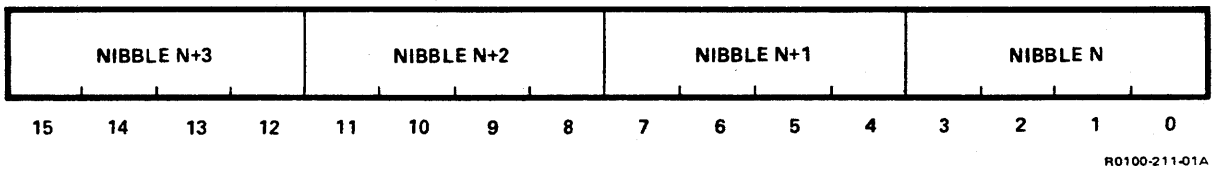
Mode Adjust 1 and 2 are used to convert bytes and nibbles to word counts by rounding up and dividing.

VECTOR FORMAT



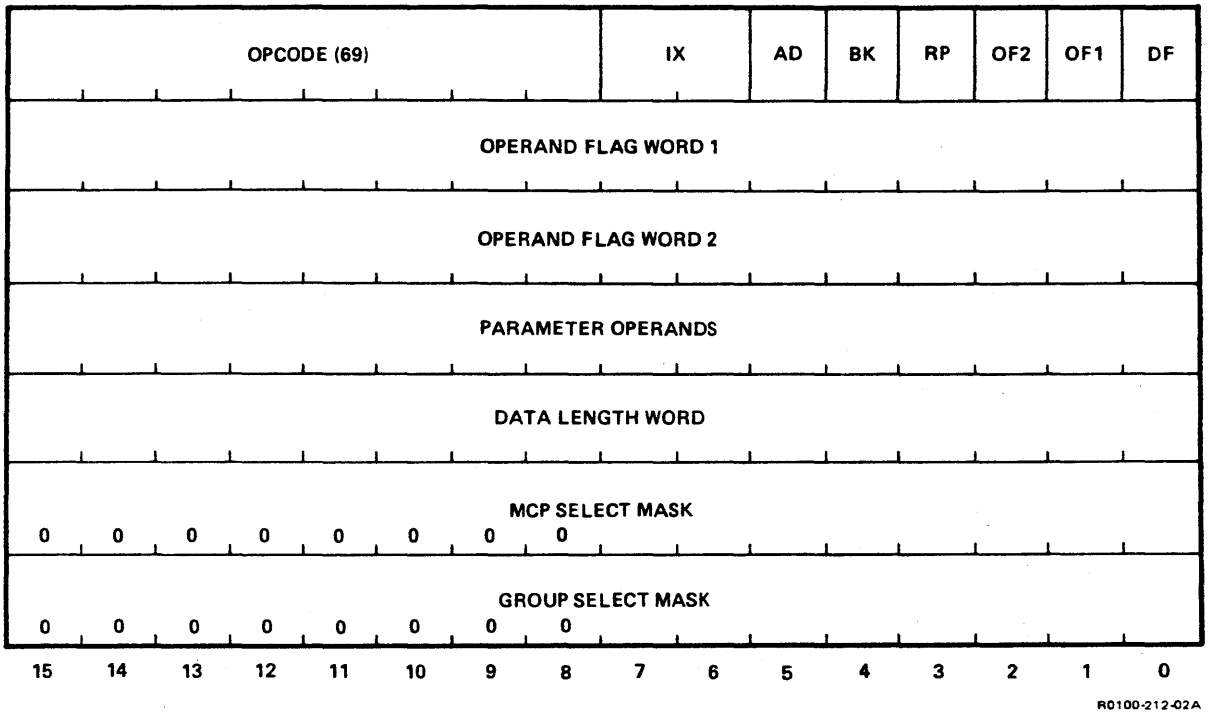

(X-START, Y-START) and (X-END, Y-END) describe the endpoint coordinates of the vector. The NUMBER OF IMAGE DATA VALUES describes how many pixel values are included for the vector. In word mode it describes the number of words, in byte mode it describes the number of bytes, and in nibble mode it describes the number of nibbles. Therefore, in byte and nibble modes there can be "residue" in the last word of the image data. In byte mode the last byte can be discarded, and in nibble mode, one, two, or three nibbles can be discarded.

| Errors | Description |
|---|---|
| Code = 8007 | Odd BYTE count. |
| Code = 8036 | Illegal PACKING DESCRIPTOR. |
| Code = 8038 | VECTOR COUNT equals zero. |




WRITE MASK = 0F00(H)
MODE = Nibble Mode = 2
SD = Right = 1
INITIAL SHIFT = 8 to shift NIBBLE N to align with memory bits 11-8.
PACKING DESCRIPTOR = 0182

## COPY IMAGE (COPY)




COPY is a normal-format instruction which can copy image data from a source MCP/group to a destination MCP/group. All LAF functions can be performed during copy. The image copy is performed in two operations. First it reads image data from the refresh memory belonging to the source MCP/group and stores it in Z80 or MC68000 RAM. Then it combines the data in Z80 or MC68000 RAM with the image data in the refresh memory belonging to the destination MCP/group, controlled by the LAF parameter.

The current MCP/group is the destination, and the source MCP/group is specified in the instructions. The START-POINT defines the starting pixel coordinate to copy from the source MCP/group. If START-POINT is not defined and SCAN is specified, the starting coordinate to copy from will be the FORMAT WINDOW origin determined by SCAN. If neither the START-POINT nor SCAN are defined, the COP resulting from the previous instruction is used as the starting pixel coordinate. The starting pixel coordinate of the destination MCP/group is always the FORMAT WINDOW origin determined by SCAN and video orientation. The FORMAT WINDOW defines a rectangular area into which data will be copied. The amount of image data read from the source MCP/group is the same as the size of the FORMAT WINDOW. In LAF replacement mode multiple MCPs and groups as destination are allowed. In all other cases of LAF operations only a single MCP/group as destination is permitted.

| Parameters | Description |
|---|---|
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, then writing to refresh memory is enabled; otherwise, writing to refresh memory is disabled. |
| IMAGE MODE | Defines the reading and writing mode used to copy image data. If FUNCTION is specified, the image-processing function is performed with the whole word of the current destination memory for all image modes.<br><br>Image Mode 0 : refresh memory plane = WMSK .AND. FFFF<br>Image Mode 1 : refresh memory plane = WMSK .AND. 00FF<br>Image Mode 2 : refresh memory plane = WMSK .AND. .FF00 |
| DATA LENGTH WORD | Defines number of bytes for MCP SELECT MASK and GROUP SELECT MASK. The value is 4. |

## Data Format

The word following the DATA LENGTH WORD is the MCP SELECT MASK for the source MCP. The bit set to one in the MCP SELECT MASK selects the corresponding MCP as the source MCP; for example, bit 0 set to one selects MCP 0, bit 1 set to one selects MCP 1, etc.

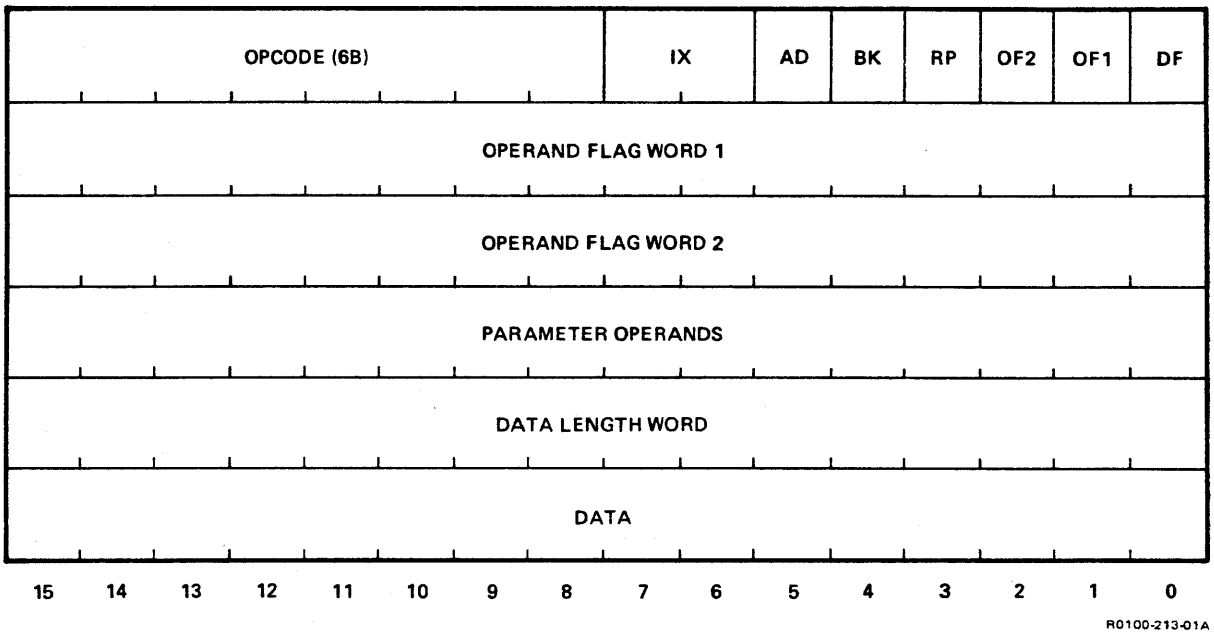
The word following the MCP SELECT MASK is the GROUP SELECT MASK for the source group associated with the selected source MCP. The bit set to one in the GROUP SELECT MASK selects the corresponding group as the source group.

## COP Movement

The resulting COP after completion of a copy instruction is the format window origin determined by SCAN and video orientation.

| Errors | Description |
|---|---|
| Code = 6902 | Not enough RAM available |
| Code = 6907 | Illegal byte count; not equal to 4. |
| Code = 6917 | No MCPs selected to read from or write into. |
| Code = 6918 | Too many MCPs selected to read from or write into. |
| Code = 6919 | No group selected to read from or write into. |
| Code = 691A | Too many groups selected to read from or write into. |

## COMBINE IMAGE (COMBI)




COMBI is a normal-format instruction which combines image data within the FORMAT WINDOW with the data supplied in the instruction. The image data combination is controlled by IMAGE mode and the Logic/Arithmetic Function code. The first image data value read from the host is combined with the pixel defined by START-POINT in refresh memory. If START-POINT is not specified, but SCAN is specified in the COMBI instruction, the first image data value is combined with the pixel at the FORMAT WINDOW origin determined by SCAN and video orientation. If neither START-POINT nor SCAN is specified, the first image data value is combined with the pixel at the COP as the result of the previous instruction. The successive image data values are combined with the pixels in refresh memory based upon the primary update direction defined by SCAN.

When a window boundary is encountered while combining image data values with successive pixels based upon the primary update direction, the COP is set to the opposite window boundary and incremented in a direction based upon the secondary update direction defined by SCAN. When a window boundary is crossed while incrementing in the secondary update direction, the COP is repositioned to the opposite window boundary.

When image data is loaded in byte mode, each image line is composed of an even number of bytes. If the FORMAT WINDOW parameter has defined a window with an odd number of pixels in the primary update direction, one extra pixel is written on each image line. It is possible that fewer lines will be written than expected. For example, if a square window were defined with 15 pixels on a side and a DATA LENGTH WORD of 225 bytes were sent to write all the pixels within the window, 16 pixels instead of 15 pixels would be written on each line; and instead of 15 full lines, 14 full lines and 1 pixel on the last line

| Parameters | Description |
|---|---|
| FUNCTION | Defines one of 13 possible image-processing functions to be applied as each value is stored in refresh memory. A value of zero defines a read-write with no processing operation, while the other 12 nonzero values represent read-modify-write operations. |
| START-POINT | Defines the coordinate of the first pixel to be written in with image data. If neither START-POINT nor SCAN is defined, the COP resulting from the previous instruction is used as the starting pixel coordinate. |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, then writing to refresh memory is enabled; otherwise, writing to refresh memory is disabled. |
| IMAGE MODE | Defines the writing mode used to transfer the host image data into refresh memory. If FUNCTION is specified, the image-processing function is performed using the whole word, low byte only, or high byte only for modes 0, 1, or 2, respectively. |
| DATA LENGTH WORD | Represents the number of bytes to be transmitted from the host processor to the RM-9460. In word mode image data is defined on a word basis, and DATA LENGTH WORD must always reflect an even number of bytes. If DATA LENGTH WORD is odd, the data is discarded and an illegal instruction interrupt is generated. In byte mode image data is defined on a byte basis so that this parameter may take any value, either odd or even. If DATA LENGTH WORD is odd, one extra byte is read from the host and discarded. For example, if DATA LENGTH WORD had a byte count value of 33 bytes, 17 words would be expected by the RM-9460, and the 34th byte would be discarded. |

Data Format

Image data in word mode is interpreted as word-per-pixel data; each word received from the host represents one pixel. Each bit of an incoming data word is written to its corresponding memory plane; that is, data bit 0 to memory plane 0, data bit 1 to memory plane 1, If RP = 1 within a COMBI instruction, the data bytes are reversed in the data word before being stored in refresh memory; the data-bit to memory-plane correspondence is subsequently modified.

| HEX | MNEMONIC | DESCRIPTION |
|---|---|---|
| 0500 | RSET | ;Clear Screen |
| 6B07 | COMBI+OF2+OF1+DF | ;Combine Image Instruction |
| 08C0 | LAF+SCN+WIN | ;Operand Flag 1 |
| 0020 | IMG | ;Operand Flag 2 |
| 0064 | 100 | ;Element Left Margin = 100 |
| 0064 | 100 | ;Line Top Margin = 100 |
| 0067 | 103 | ;Element Right Margin = 103 |
| 0067 | 103 | ;Line Bottom Margin = 103 |
| XXXX | XXXX | ;Scan Value from 0 through 7 |
| **** | **** | ;Logic/Arithmetic Function Codes |
| | | ; from 0 through C |
| 0000 | 0 | ;Image Mode = Word Mode |
| 0020 | 32 | ;Data Length Word = 32 |
| | D1 | ; |
| | D2 | ;32 Bytes (16 words) of Data |
| | . | |
| | . | |
| | . | |
| | . | |
| | D16 | |

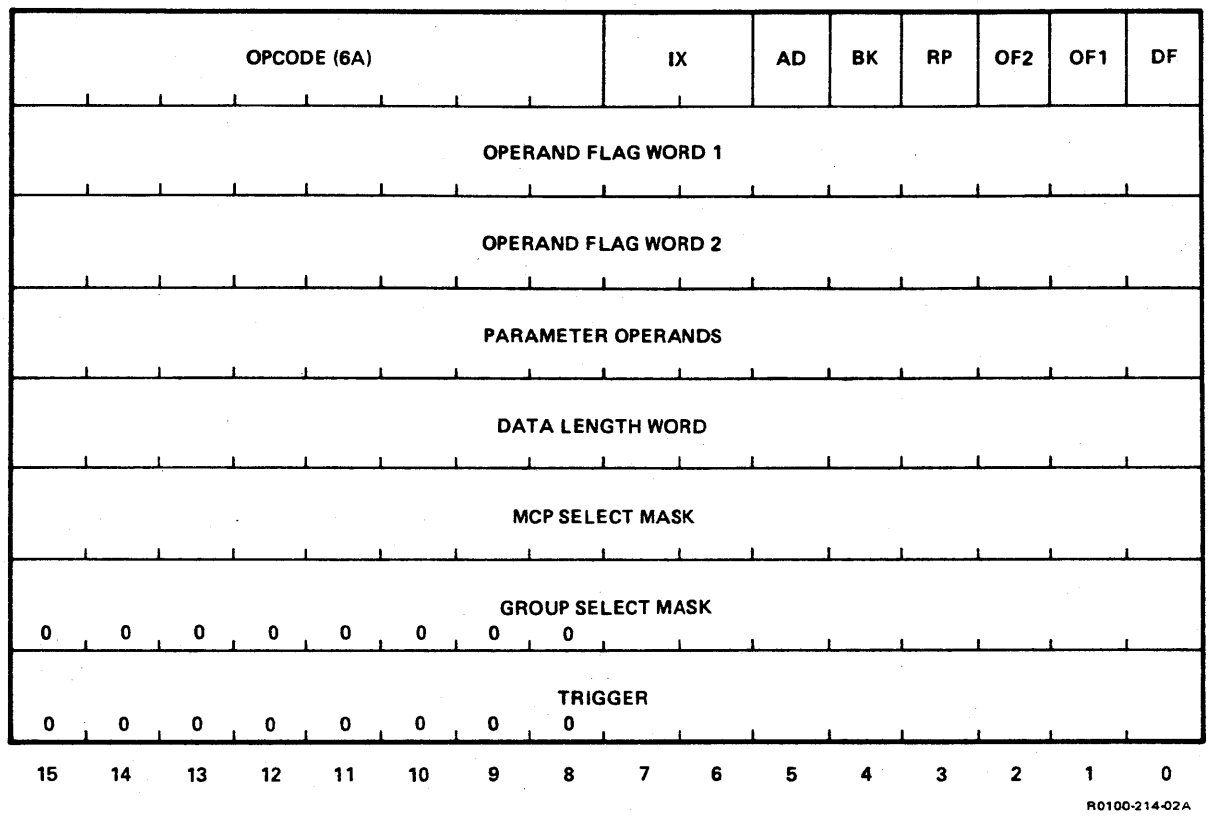
where XXXX defines the scan mode from 0 through 7
**** defines the Logic/Arithmetic Function codes from 0 through C

Note that the COP, after all image combines are complete, is identical to the starting COP. Both primary and secondary updates are exercised after data D16 is combined.

| HEX | MNEMONIC | DESCRIPTION |
|---|---|---|
| 0060 | RSET | ;Clear Screen |
| 6B07 | COMBI+OF2+OF1+DF | ;Combine Image Instruction |
| 08C0 | LAF+SCN+WIN | ;Operand Flag 1 |
| 0020 | IMG | ;Operand Flag 2 |
| 0064 | 100 | ;Element Left Margin X min |
| 0064 | 100 | ;Line Top Margin Y min |
| 0067 | 103 | ;Element Right Margin X max |
| 0067 | 103 | ;Line Bottom Margin Y max |
| XXXX | XXXX | ;Scan Mode |
| **** | **** | ;Logic/Arithmetic Function Codes |
| | | ; from 0 through C |
| 0001 | 1 | ;Image Mode = Low Byte Mode |
| 0010 | 16 | ;Data Length Word = 16 |
| | D1D2 | ; |
| | D3D4 | ;16 Bytes packed 2 per word |
| | | ;or 8 words |
| | D15D16 | |

where XXXX defines the scan mode 0 through 7
**** defines the Logic/Arithmetic Function codes 0 through C

## COPY IMAGE TRIGGERED (COPYT)




COPYT is a normal-format instruction which copies image data from source MCP/group to destination MCP/group, and performs all LAF functions.

TRIGGER is a refresh memory-plane mask used to decide whether the LAF function will be performed on each individual pixel. If the result of a logical AND operation of the destination refresh memory and TRIGGER is nonzero, the combine image operation will be performed in the same way as COPY. For LAF operations using replacement mode, multiple MCPs and groups are allowed as destinations. For all other modes, only a single MCP/group is permitted as a destination. Any restriction which applies to the COPY instruction will also be effective in COPYT.

| Flags | Description |
|---|---|
| IX | Defines the address mode in which INDEX 1, INDEX 2, FORMAT WINDOW, and START-POINT are evaluated. |

| Parameters | Description |
|---|---|
| IMAGE MODE | Defines the reading and writing mode used to copy image data. If FUNCTION is specified, the image-processing function is performed using the whole word of the current destination memory for all image modes.<br><br>Image Mode 0 : refresh memory plane = WMSK .AND. FFFF<br>Image Mode 1 : refresh memory plane = WMSK .AND. 00FF<br>Image Mode 2 : refresh memory plane = WMSK .AND. FF00 |
| DATA LENGTH WORD | Defines number of bytes for MCP SELECT MASK, GROUP SELECT MASK, and TRIGGER. The value is 6. |

## Data Format

The word following the DATA LENGTH WORD is the MCP SELECT MASK for the source MCP. The bit set to one in the MCP SELECT MASK selects the corresponding MCP as the source MCP; for example, bit 0 set to one selects MCP 0, bit 1 set to one selects MCP 1, etc.

The word following the MCP SELECT MASK is the GROUP SELECT MASK for the source group associated with the selected source MCP. The bit set to one in the GROUP SELECT MASK selects the corresponding group as the source group.

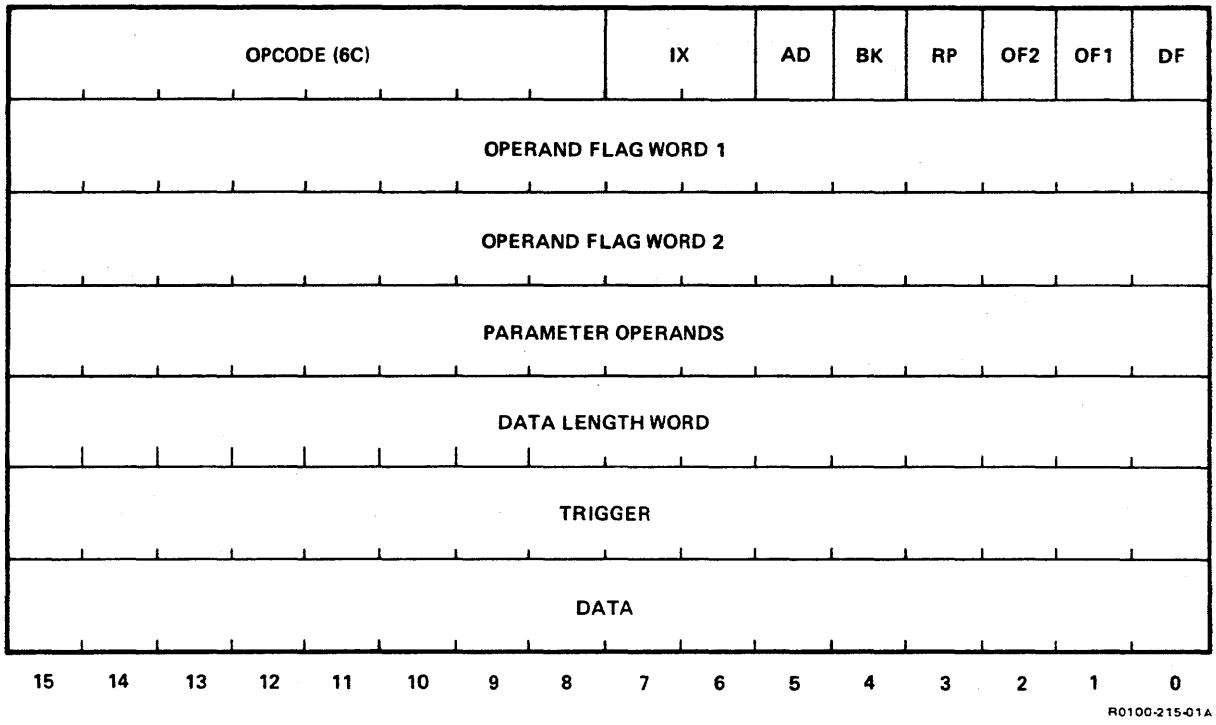
The word following the GROUP SELECT MASK is the TRIGGER value that determines whether the LAF operation will be performed on each individual trigger.

## COP Movement

After completion of the COPYT instruction, the COP is the starting point in the FORMAT WINDOW, determined by SCAN and video orientation.

| Errors | Description |
|---|---|
| Code = 6A02 | Not enough RAM available. |
| Code = 6A07 | Illegal byte count; not equal to 6. |
| Code = 6A17 | No MCPs selected to read from or write into. |
| Code = 6A18 | Too many MCPs selected to read from or write into. |
| Code = 6A19 | No group selected to read from or write into. |
| Code = 6A1A | Too many groups selected to read from or write into. |

## COMBINE IMAGE TRIGGERED (COMBT)




COMBT is a normal-format instruction combines image data within the FORMAT WINDOW with the data supplied in the instruction. The method of combining image data depends upon either word mode or byte mode and also is controlled by Logic/Arithmetic Function and TRIGGER.

TRIGGER is a refresh memory-plane mask used to decide whether the combined image operation will be performed. If the result of a logical AND operation of current refresh memory and TRIGGER is nonzero, then the combined image operation will be performed. All other functions of this instruction will be the same as those of COMBI instruction.

| Flags | Description |
|---|---|
| IX | Defines the address mode in which INDEX 1, INDEX 2, FORMAT WINDOW, and START-POINT are evaluated. |
| RP | Defines the byte-accessing order in byte mode and the byte significance in word mode as follows: |

| Parameters | Description |
|---|---|
| START-POINT | Defines the coordinate of the first pixel to be written with image data. If neither START-POINT nor SCAN is defined, the COP resulting from the previous instruction is used as the starting pixel coordinate. |
| WRITE-ENABLE WINDOW | Defines the area of global space where data may be written. |
| W.E.W. OFFSET | Defines the area of refresh memory that overlays global space where data may be written. |
| DISPLAY CLASS | Defines the current DCL value. If this value is in one of the defined display-class ranges, then writing to refresh memory is enabled; otherwise, writing to refresh memory is disabled. |
| IMAGE MODE | Defines the writing mode used to transfer the host image data into refresh memory. If FUNCTION is specified, the image-processing function is performed using the whole word, low byte only, or high byte only for modes 0, 1, or 2, respectively. |
| DATA LENGTH WORD | Represents the total number of bytes of data to be transmitted from the host processor to the RM-9460 plus two bytes of TRIGGER. In word mode imaged data is defined on a word basis and DATA LENGTH WORD must always reflect an even number of bytes. If DATA LENGTH WORD is odd, the data is discarded and an illegal instruction interrupt is generated. In byte mode image data is defined on a byte basis so that this parameter may take any value, either odd or even. If DATA LENGTH WORD is odd, one extra byte is read from the host and discarded. For example, if DATA LENGTH WORD had a byte count value of 33 bytes, 17 words would be expected by the RM-9460, and the 34th byte would be discarded. |

## Data Format

Image data in word mode is interpreted as word-per-pixel, so each word received from the host represents one pixel. Each bit of an incoming data word is written to its corresponding memory plane; that is, data bit 0 to memory plane 0, data bit 1 to memory plane 1, If RP = 1 within a COMBT instruction, the data bytes are reversed in the data word before being stored in refresh memory; the data-bit to memory-plane correspondence is subsequently modified.

Image data in byte mode is interpreted as byte-per-pixel data, so each word received from the host represents two pixels. The correspondence between each byte's data bits and memory planes is determined by the IMAGE MODE parameter:

Where XXXX defines the scan mode from 0 through 7.

The COPY and FUNCTION operation will be performed to pixels in Group 0 whenever bit 1 in on in the destination pixel (that is, TRIGGER value of 2 has only bit 1 on). If the TRIGGER was 010(H), the COPY and FUNCTION operation will be performed if bits 8 or 0 are on.

Note that the COP, after all the image combined in complete, is identical to the starting COP. Both primary and secondary updates are exercised after value D16 is combined.

Example:

The following example demonstrates the use of the COMBT instruction byte mode. Each data value (D0-D16) is a byte rather than a word. The window is defined by the coordinate corners (100, 100) and (103, 103). Byte mode is selected, and 16 bytes of data are combined into WINDOW, based upon the Logic/Arithmetic Function code and TRIGGER.

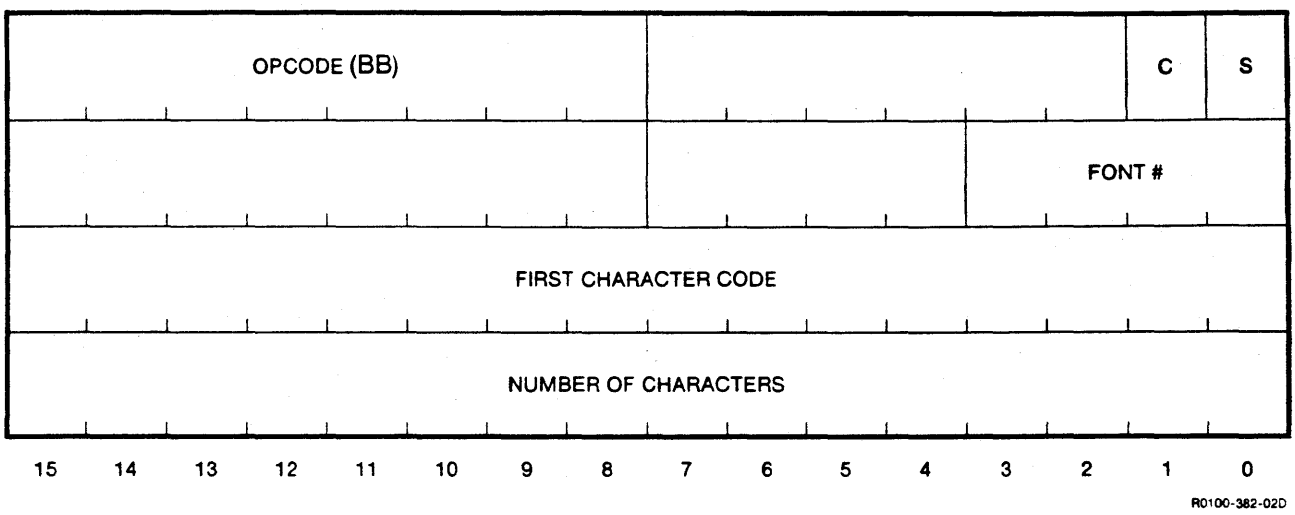
```
HEX          MNEMONIC              DESCRIPTION

0500           RSET                ;Clean Screen
6B07      COMBT+0F2+0F1+DF         ;Combine Image Triggered Instruction
08C0        LAF+SCN+WIN            ;Operand Flag 1
0020            IMG                ;Operand Flag 2
0064            100                ;Element Left Margin
0064            100                ;Line Top Margin
0067            103                ;Element Right Margin
0067            103                ;Line Bottom Margin
XXXX           XXXX                ;Scan Mode
****           ****                ;Logic/Arithmetic Function Codes
                                   ;  from 9 through C
0001              1                ;Image Mode = Low Byte Mode
0010             16                ;Data Length Word = 16
                  2                ;TRIGGER = 2
                D1D2               ;
                D3D4               ;16 Bytes packed 2 per word
                  .                ;or 8 words
                  .
                  .
               D15D16
```

Where XXXX defines the scan mode from 0 through 7

When **** defines the Logic/Arithmetic Function codes from 0 through C

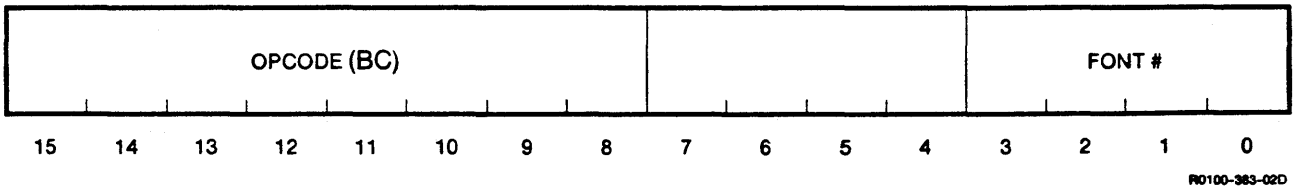
## ALLOCATE STROKED FONT (ASF)




ASF is a special-format instruction which allocates RAM from the free RAM pool for storing general information about the specified font.

| Parameters | Description |
|---|---|
| S | Indicates whether the characters codes are 16-bit word values (S=1) or 8-bit word values (S=0). |
| C | If set, indicates that font #0 is to be copied to the font specified by FONT #. |
| FONT # | Specifies the number of the allocated font (1-15). Font #0 is automatically allocated at power-up. |
| FIRST CHARACTER CODE | Specifies the character code of the first character defined in the allocated font. If the characters codes are 16-bit word values (S=1), this number can range from 0 to 5999. If the characters codes are 8-bit word values (S=0), this number can range from 0 to 255. |
| NUMBER OF CHARACTERS | Indicates the number of characters to be defined in the allocated font. If the characters codes are 16-bit word values (S=1), the maximum number of characters that can be defined is 256. If the characters codes are 8-bit word values (S=0), the maximum number of characters that can be defined is 6000. |

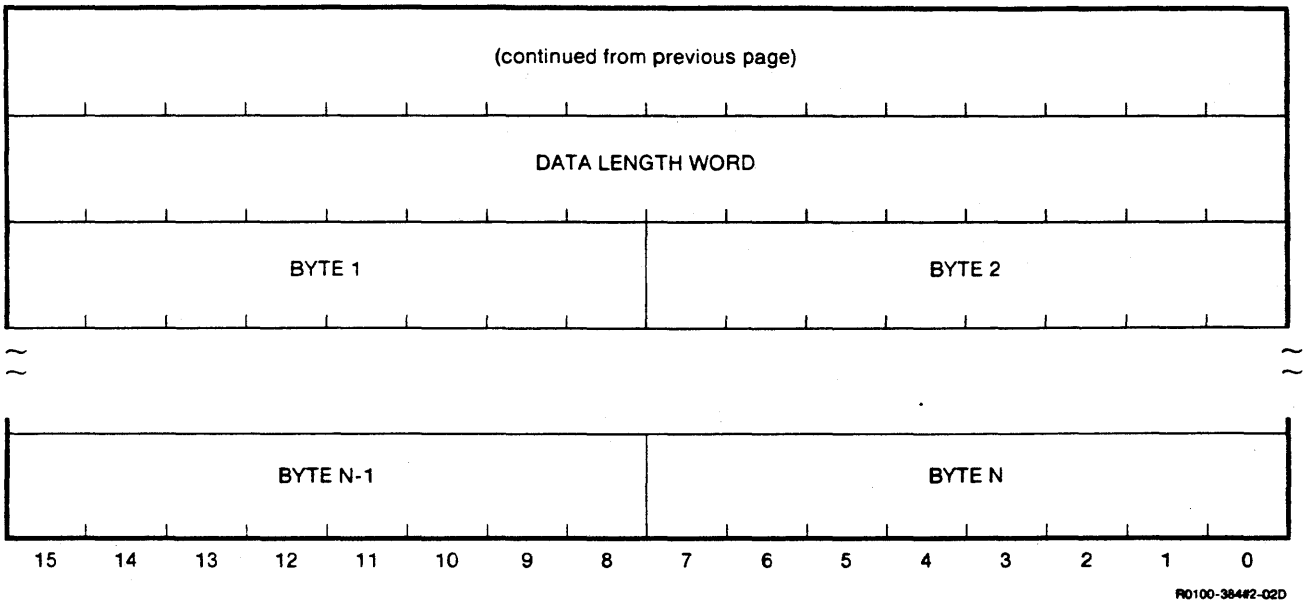
## DEALLOCATE STROKED FONT (DSF)




DSF is a special-format instruction which frees the RAM associated with the specified font number and frees the number for reallocation.

| Parameters | Description |
|---|---|
| FONT # | Specifies the number of the font to be deallocated (1-15) Font 0 cannot be deallocated. |

| Errors | Description |
|---|---|
| Code = BC01 | Attempt to deallocate font 0. |
| Code = BC0D | Attempt to deallocate a font that has not been allocated. |

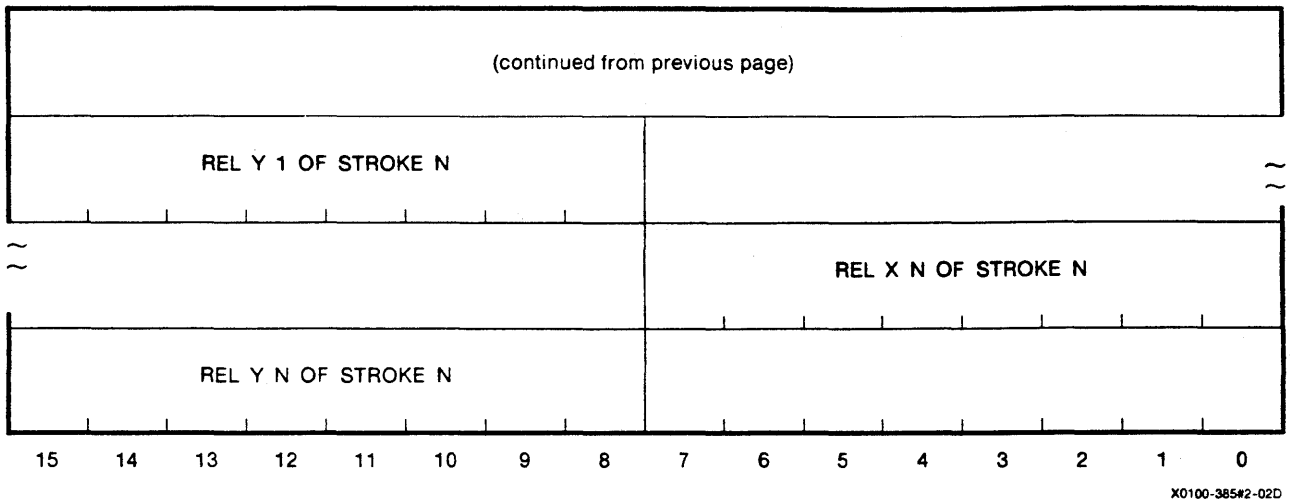



DEFSF is a special-format instruction used to define (add or overwrite) characters in the specified font. Each of the characters is formed by a collection of polylines defined in the font coordinate system. The origin of the font coordinate system is the lower left-hand corner of the character cell, with positive X going in the horizontal direction (right) and positive Y going in the vertical direction (up).

| Parameters | Description |
|---|---|
| FONT # | Specifies the number of the font (1-15) in which characters are being defined. Font #0 is automatically defined at power-up and cannot be redefined. |
| NUMBER OF CHARACTERS | Specifies the number of characters to be defined in the instruction. If the characters codes are indexed with 16-bit word values (S=1 in the ALLOCATE STROKED FONT instruction), this number can range from 0 to 5999. If the characters codes are indexed with 8-bit word values (S=0 in the ALLOCATE STROKED FONT instruction), this number can range from 0 to 255. |
| CHARACTER CODE | Specifies the character code of the character to be defined immediately following the data length word. |
| DATA LENGTH WORD | Defines the number of bytes of data defining the character specified by CHARACTER CODE. This number must be even. |

## Data Format of Character Descriptors

Each character specified by CHARACTER CODE in this instruction is defined in the following format:

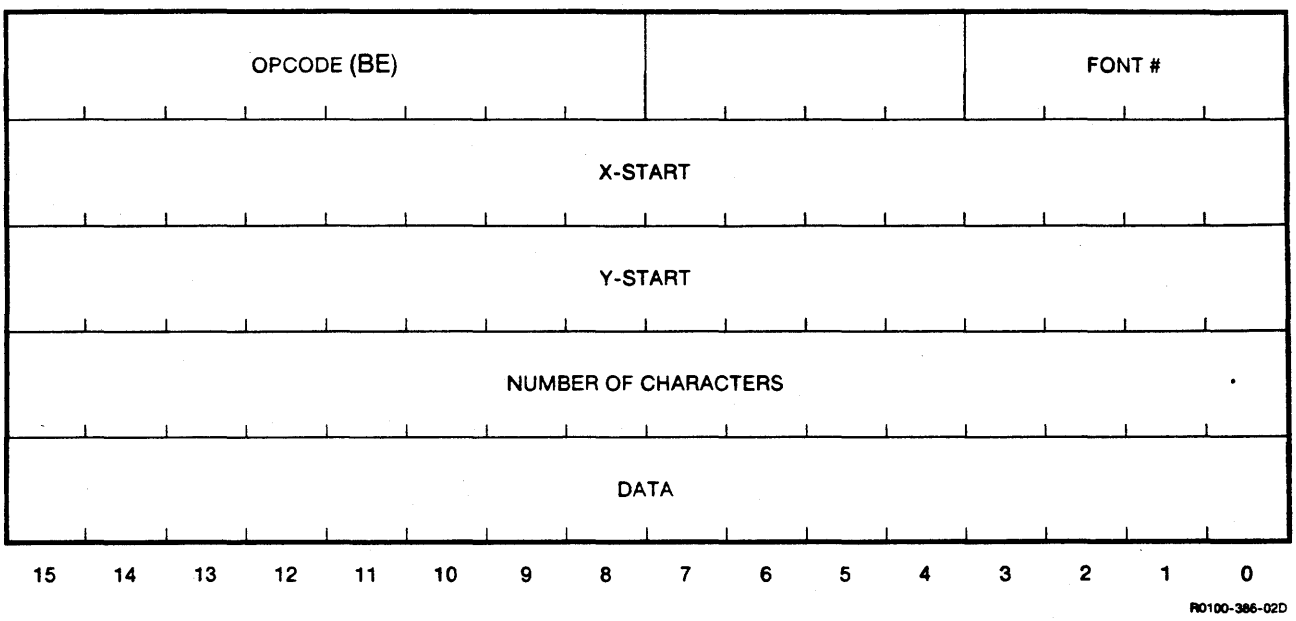



This definition of each character would fit into the instruction format in the bytes labeled BYTE 1 through BYTE N. A separate character descriptor would be supplied for each character code given.

| Parameters | Description |
|---|---|
| NUMBER OF STROKES | Specifies the number of strokes (linked vectors) used to describe the specified character. |
| HORIZONTAL CELL SIZE | Specifies the size of the character cell in the positive X direction of the font coordinate system. |
| VERTICAL CELL SIZE | Specifies the size of the character cell in the positive Y direction of the font coordinate system. |
| NUMBER OF VECTORS IN STROKE | Specifies the number of vectors that make up the stroke. |
| X-START, Y-START OF STROKE | Specifies the location in font coordinates where the stroke begins. |
| REL X, REL Y OF STROKE | Specifies the relative position (in font coordinates) of the end point of a vector from the end point of the previous vector (or start point of the current vector). |

| Errors | Description |
|---|---|
| Code = BD0D | Attempt to define a font that has not been allocated. |
| Code = BD0E | Attempt to redefine font #0. |

## WRITE STROKED TEXT (WST)



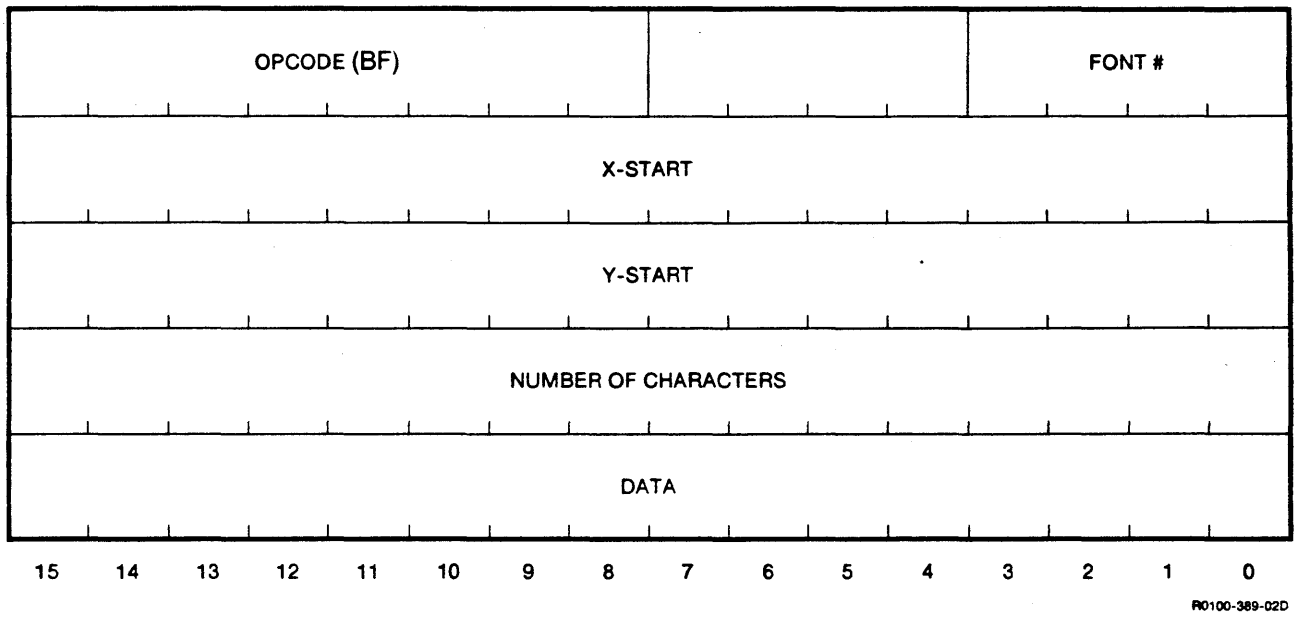

WST is a special-format instruction which displays a text string made up of specified characters of a specified font in a specified location. The character is always drawn with a FOREGROUND pixel value, and the character cell is not drawn.

| Parameters | Description |
|---|---|
| FONT # | Specifies the font to be used in this instruction. This number can range from 0 to 15. |
| X-START, Y-START | Position where the first character is to be drawn. |
| NUMBER OF CHARACTERS | Specifies the number of characters to be displayed. |

### Data Format

If the characters specified are indexed with 16-bit word values (S=1 in the ALLOCATE STROKED FONT instruction), the data provided will be in the following format:

## INQUIRE TEXT EXTENT (ITE)




ITE is a special-format instruction which reads a text string made up of specified characters of a specified font with a specified start point and returns the end point of the text string (in global coordinates) to the host computer.

| Parameters | Description |
|---|---|
| FONT # | Specifies the font to be used in this instruction. This number can range from 0 to 15. |
| X-START, Y-START | Position where the first character is drawn (lower left corner of the character cell). |
| NUMBER OF CHARACTERS | Specifies the number of characters in the string. |

### Data Format for Characters in the String

If the characters specified are indexed with 16-bit word values (S=1 in the ALLOCATE STROKED FONT instruction), the data provided will be in the following format:

The start point of the character string is specified in local coordinates, and the end point is returned in global coordinates. If a matrix other than the identity matrix is current when the instruction is sent, the string will undergo coordinate transformation. If either the start point or the end point lies outside the WRITE-ENABLE WINDOW after coordinate transformation, one of the following will occur:

1. If the specified start point or the calculated end point is outside the minimum X coordinate boundary, then the value returned for X will be 0.

2. If the specified start point or the calculated end point is outside the maximum X coordinate boundary, then the value returned for X will be FFFF.

3. If the specified start point or the calculated end point is outside the minimum Y coordinate boundary, then the value returned for Y will be 0.

4. If the specified start point or the calculated end point is outside the maximum Y coordinate boundary, then the value returned for Y will be FFFF.

| Errors | Description |
|---|---|
| Code = BF0D | Specified font not defined. |

# Chapter 3

# INTERFACE

## 3.1 INTRODUCTION

This chapter describes system controls and indicators and the interface between the RM-9460 system and the host CPU.

## 3.2 OPERATING CONTROLS

The RM-9460 system has only two operating controls; both are accessible at the front panel. They operate as follows:

- POWER ON: Set the switch to ON to apply power and initiate a reset. Set the POWER switch to the down position to remove system power.
- RESET: Press the RESET switch down to reset the system. This reset has the same effect as the reset which occurs each time the system is powered on.

## 3.3 INDICATORS

Two front panel indicator lamps are installed adjacent to the POWER and RESET switches. Their purpose is as follows:

- The lamp adjacent to the POWER ON switch is on when power is applied to the system and is off when power is removed.
- The SELF TEST lamp adjacent to the RESET switch is initially on when power is first applied and each time the RESET switch is pressed. During the on period a checksum internal test is performed. If the self-test is satisfactory, the SELF TEST lamp goes off and remains off during normal system operation.

## 3.4 HOST CPU TO RM-9460 INTERFACE

The host CPU is connected to the RM-9460 via one cable, which provides both data transfer and control signals required for interfacing the system. The cable has a connector which mates with connector J1 on the rear of the RM-9460. An internal cable in the RM-9460 ties connector J1 to a general purpose interface PCB (hard GPIF) in slot A1. The hard GPIF was selected for this text because it is representative of several PCBs which may be used in particular systems. Figure 3-1 is a generalized interfacing diagram showing host CPU to RM-9460 system interface.

Table 3-1. Computer Interface Equipment

| MODEL NUMBER | DESCRIPTION | PREREQUISITE |
|---|---|---|
| RM-9460-40 | General purpose bidirectional interface. Operation in PIO and DMA modes. | |
| RM-9460-53 | DEC PDP-11 Series bidirectional interface. Operation in PIO and DMA modes. Utilizes one SPC slot. | Non-processor grant line at the interface slot for DMA operation |
| RM-9460-54 | DEC LSI-11 series bidirectional Q-bus interface. Operation in QIO and DMA modes. | Non-processor grant line at the interface slot for DMA operation |
| RM-9460-56 | Interdata 7/8XX Series bidirectional interface to programmed I/O multiplexer bus or SELCH. | M48-13 compatible card (provided) |
| RM-9460-59 | Univac AN/UYK-7 bidirectional interface to NAVY Type B (NTDS FAST) digital data interface (0 V, -3 V logic levels). Usually a DMA interface. See MIL-Std. 1397 (ships). | Navy Type B NTDS FAST digital data interface (I/O controller) |
| RM-9460-61 | HP-21MX Series bidirectional PIO/DMA interface. | |
| RM-9460-64 | Varian 620 and 73 Series bidirectional interface to buffered I/O controller. | Varian (BIOC) buffered I/O controller board (P/N E2832) and priority interrupt module (PIM) Model 620/i-16, and CPU for checkout |
| RM-9460-65 | Data General Nova and Eclipse bidirectional interface. | Data General Model 4192 cable in CPU |
| RM-9460-71 | IBM 2701 Parallel Data Adapter (PDA) bidirectional interface. | |

completion of a read data operation always initiates a new internal read cycle. When PRE is true, IOREADY goes false while the output register is loaded and the read operation is performed.

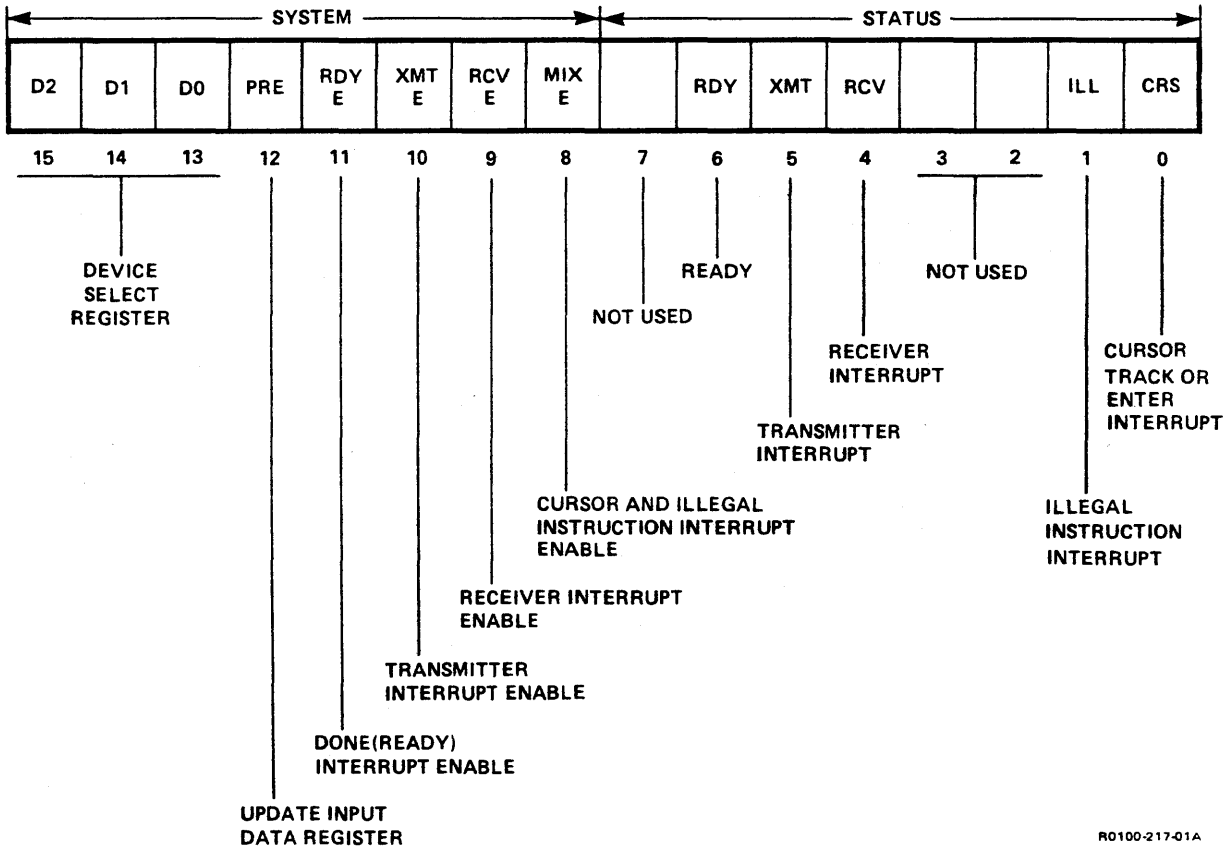
5. Bit 9 (DEV SEL). Select enable. When DEV SEL is true, bits 8-6 are loaded into the device select register, thus selecting a new device. When DEV SEL is false, bits 8-6 are ignored.

6. Bits 8-6 (D2, D1, D0). Device select register. D2, D1 and D0 select the various 9000-series systems. Set to false.

7. Bits 5-0. Set to false.

### 3.5.2 Status Register

The status register contains system control bits as well as interrupt and status. The bit assignments in the status register are described below.



### 3.6.3 Receiver Interrupt Request

The receiver interrupt request is generated when a key on the keyboard is depressed. The eight-bit code associated with the depressed key is stored in the FIFO (first-in-first-out) buffer associated with that particular keyboard and may be read by the host processor using the READ KEYBOARD instruction.

If any characters are present in the keyboard buffers after execution of a READ KEYBOARD instruction, a receiver interrupt request will be issued.

### 3.6.4 Transmitter Interrupt Request

The transmitter interrupt request is generated upon completion of a serial output transmission using the WRITE KEYBOARD instruction. Completion is indicated to the firmware by the serial link PCB. A transmitter interrupt request indicates that another WRITE KEYBOARD instruction to the interrupting output device is valid.

# Chapter 4

# COORDINATE SYSTEM AND TRANSFORMATION CONCEPTS

## 4.1 COORDINATE SYSTEM

The RM-9460 uses a dual-ported refresh memory to hold the actual display image. This memory can be written to or read from the memory control processor (MCP) at the refresh RAM access rate. It can also be read by RM-9460 video generation logic at effective rates using a multiple-word access method. The dual-access structure dictates two refresh memory addressing schemes, digital and video. The digital addressing scheme is used by the host computer (by the Z80 or the MC68000 microprocessor and the MCP) for generating coordinate addresses of such entities as vectors or characters. The video addressing scheme is used by the generation logic and deals with the image coordinate system of the CRT.

### 4.1.1 Digital Addressing

In the digital addressing scheme two coordinate systems, global and refresh, can be defined for write or read addressing of the image memory. The X-Y coordinate system referenced by the host computer when generating graphics and image displays is called the global coordinate system. (See figures 4-1 and 4-2.) A point in this X-Y coordinate system is referred to by the coordinate pair $(X_G,Y_G)$. Variables $X_G$ and $Y_G$ must obey the following constraints:

$$-16384 \leq X_G \leq +16383$$

$$-16384 \leq Y_G \leq +16383$$

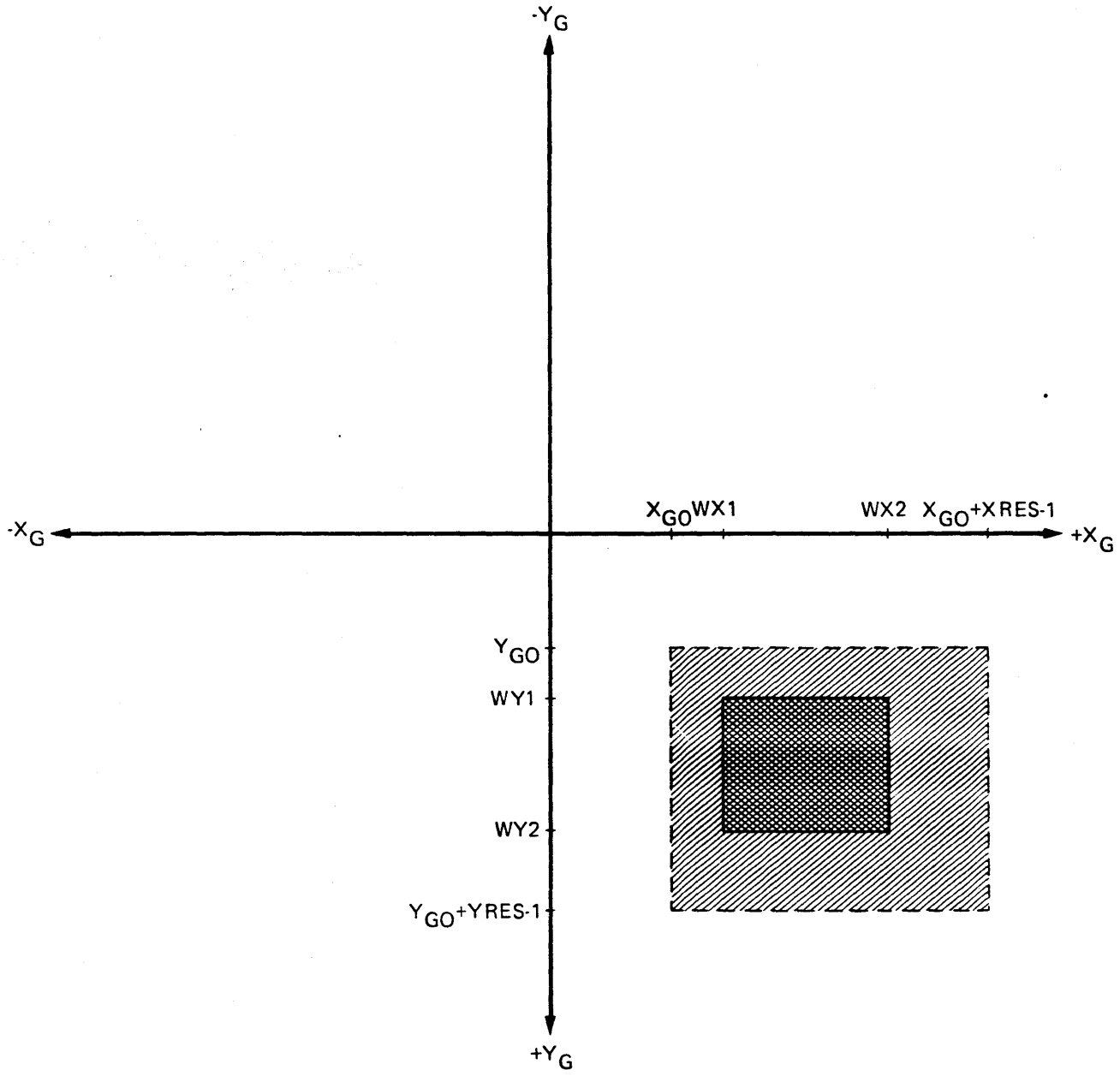
The refresh coordinate system is an X-Y coordinate system that refers to the absolute coordinate system used in the digital addressing scheme to access refresh memory pixels. A point in the refresh coordinate system is referred to by the coordinate pair $(X_R,Y_R)$. Variables $X_R$ and $Y_R$ must obey the following rules:

$$0 \leq X_R < XRES$$

$$0 \leq Y_R < YRES$$

where XRES is the element and YRES is the line resolution.

Additionally, a rectangular region within the global coordinate system, the WRITE-ENABLE WINDOW, can be defined. Data can be written into refresh memory only when the addressed pixel lies within this window. The window can be defined by the four 16-bit values WX1, WY1, WX2, and WY2. The values of WX1 and WX2 define the lower and upper element addresses of the WRITE-ENABLE WINDOW, and the values WY1 and WY2 define the lower and upper line addresses of the WRITE-ENABLE WINDOW.




THIS AREA REPRESENTS THE OVERLAP BETWEEN THE GLOBAL AND THE REFRESH COORDINATE SYSTEMS.

THIS AREA REPRESENTS THE WRITE ENABLE WINDOW.

Figure 4-2. Global and Refresh Coordinate Systems (Upper Left Video Origin)

Since the WRITE-ENABLE WINDOW can never exceed the boundaries of the refresh coordinate system, the following constraints must be placed on the values of ox, oy, WX1, WX2, WY1, and WY2:

$$ox + (WX2 - WX1) \leq XRES$$

$$oy + (WY2 - WY1) \leq YRES$$

If the host computer attempts to violate the constraints of the equation

$$ox + (WX2 - WX1) \leq XRES$$

the value of WX2 must be changed to the value WX2' where,

$$WX2' = WX2 - (ox + WX2 - WX1 - XRES)$$

or

$$WX2' = XRES + WX1 - ox$$

If the host computer attempts to violate the constraints of the equation

$$oy + (WY2 - WY1) \leq YRES$$

the values of WY2 must be changed to the value WY2' where

$$WY2' = WY2 - (oy + WY2 - WY1 - YRES)$$

or

$$WY2' = YRES + WY1 - oy$$

## 4.3 VIDEO ADDRESSING TRANSFORMATIONS

Video origin registers (XORG and YORG) allow the user to change the line and element refresh coordinates that correspond to the video coordinate system origin. In effect, these registers allow the user to perform nondestructive scrolling by changing the linear transformation equations between the refresh and video coordinate systems. The variables XORG and YORG must obey the following constraints:

$$0 \leq XORG < XRES$$

$$0 \leq YORG < YRES$$

If both conditions are true, the sets of transformations outlined for cases A and B can be defined.

Case A

$$X_V = X_G - WX1 + ox - XORG \quad \text{for} \quad X_R \geq XORG$$
$$= X_G - WX1 + ox - XORG + XRES \quad \text{for} \quad X_R < XORG$$
$$Y_V = Y_G + WY1 + oy - YORG \quad \text{for} \quad Y_R \geq YORG$$
$$= Y_G + WY1 + oy - YORG + YRES \quad \text{for} \quad Y_R < YORG$$

Case B

$$X_V = X_G - WX1 + ox - XORG \quad \text{for} \quad X_R \geq XORG$$
$$= Y_G - WX1 + ox - XORG + XRES \quad \text{for} \quad X_R < XORG$$
$$Y_V = YRES\text{-}1 - Y_G - oy + WY1 + YORG \quad \text{for} \quad Y_R \geq YORG$$
$$= -1 - Y_G - oy + WY1 + YORG \quad \text{for} \quad Y_R < YORG$$

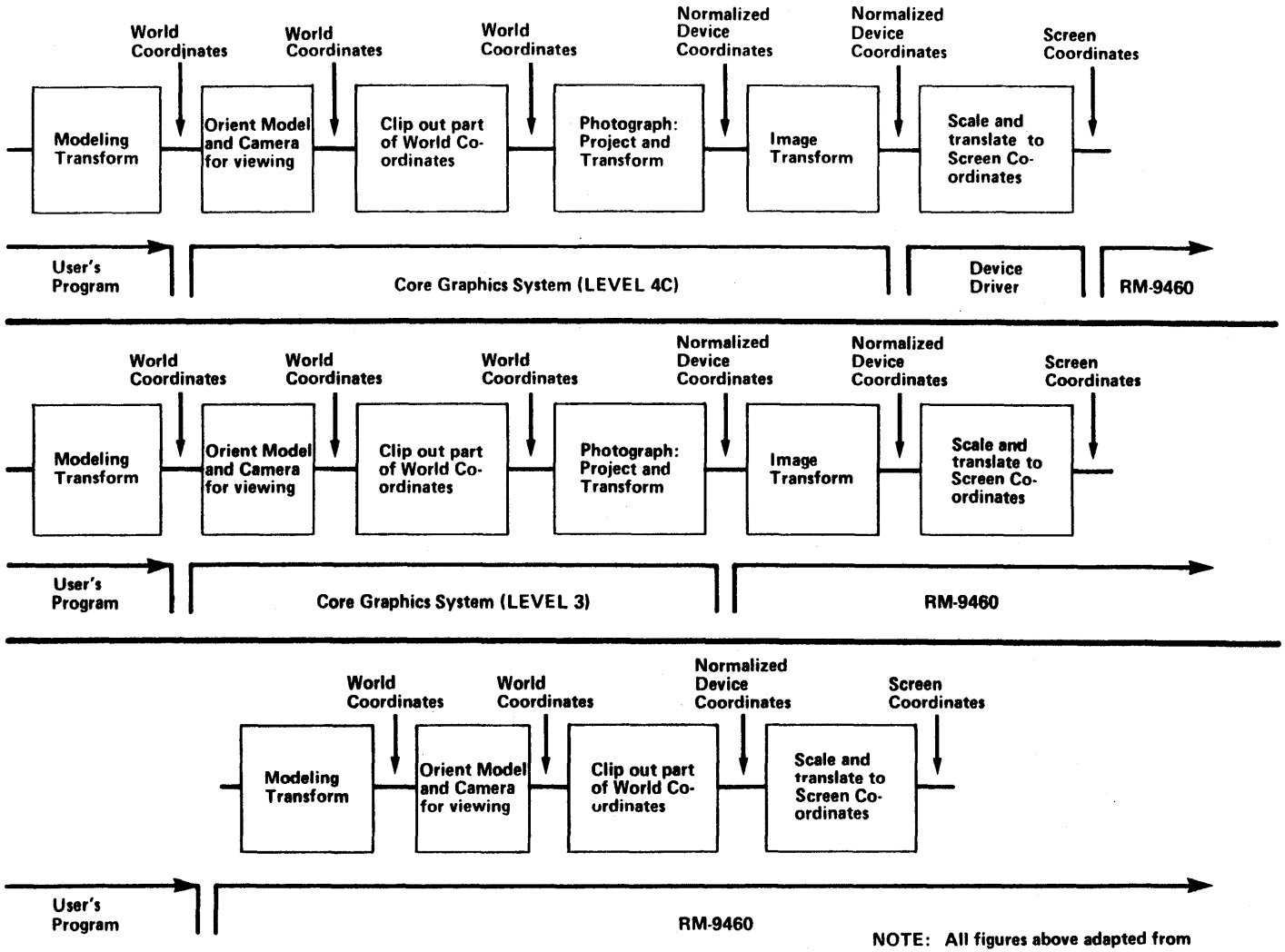
## 4.5 TRANSFORMATION CONCEPTS

The following paragraphs describe the concepts and implementation features of the coordinate transformation firmware of the RM-9460 system.

### 4.5.1 Default Conditions

To maintain compatibility with the RM-9000, the RM-9460 defaults the following parameters to values that emulate RM-9000 operation:

| | | |
|---|---|---|
| $X_{GO}$ | = 0 | Refresh Coordinate System Origin |
| $Y_{GO}$ | = 0 | |
| WX1 | = 0 | Write-Enable Window |
| WY1 | = 0 | |
| WX2 | = XRES-1 | |
| WY2 | = YRES-1 | |
| XORG | = 0 | Video Origin |
| YORG | = 0 | |
| ox | = 0 | Write-Enable Window Offset |
| oy | = 0 | |




NOTE: All figures above adapted from page II-100 status report of the Graphics Standards Planning Committee, Fall issue.

D0100-286-02A

Figure 4-3. Graphic Processing Pipeline

| | |
|---|---|
| LOAD CURRENT MATRIX m | Current matrix equals MATRIX m. |
| STORE CURRENT MATRIX m | MATRIX m equals current matrix. |
| MULTIPLY MATRICES m, n, o | MATRIX m equals MATRIX n times MATRIX o. |
| MULTIPLY MATRICES IMMEDIATE m (a,b,c,d,e,f) | MATRIX m equals supplied MATRIX times MATRIX m. |
| SCALE MATRIX m (x,y) | Changes the scale of the coordinate system defined by MATRIX m (independently for each axis). |
| TRANSLATE MATRIX m (x,y) | Moves the coordinate system of MATRIX m relative to current origin. |
| ROTATE MATRIX m (x,y) | Rotates the coordinate system of MATRIX m about its origin. |
| READ MATRIX m | Allows the host to retrieve the contents of MATRIX m. |

### 4.6.1 User Coordinate-to-Refresh-Memory-Mapping Transformation

The coordinates supplied by the user are passed through the current matrix into Ramtek global space. This is the most global of all coordinate spaces involved in the geometrical hierarchy. Its scale and rotation correspond directly to the refresh memory coordinate space; that is, a distance of one unit is the same as the distance between adjacent pixels, the X-axis being parallel to a video scan line.

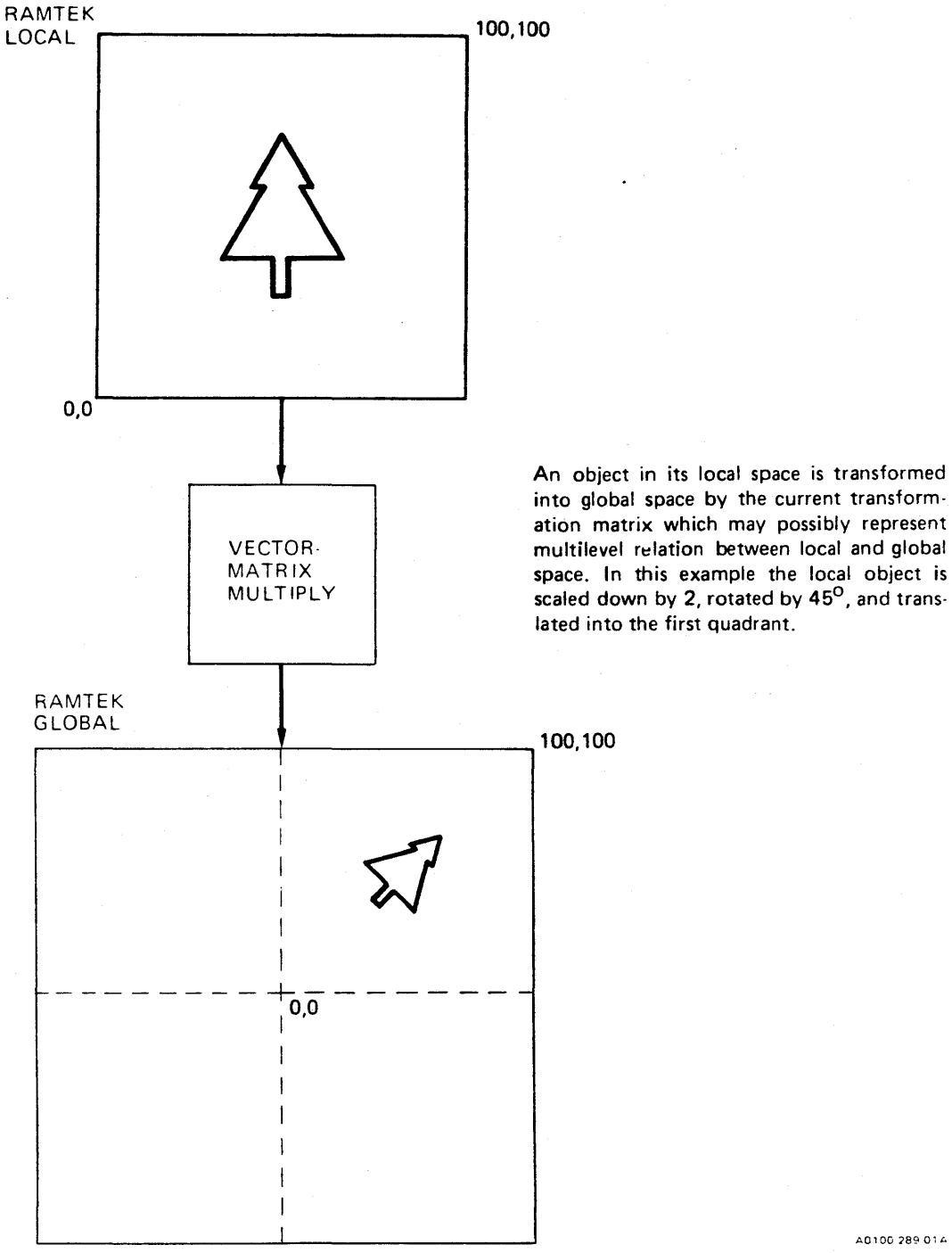
The final transformation maps a window of Ramtek global space onto a window of refresh memory space. These windows must be the same size and shape, and both must be aligned with the coordinate axes. This implies that the final (memory mapping) transformation involves only translation and clipping. (See figures 4-4 to 4-8.)

### 4.6.2 Instructions to Modify the Current Coordinate System

Through RM-9460 instructions the user specifies the geometric operation and geometric parameters. The RM-9460 performs the following:

- Forms the indicated transformational matrix
- Pre- or post-multiplies the current matrix by the new one
- Replaces the old matrix with the computed product

The operations provided to modify the current Ramtek local coordinate system are SCALE, TRANSLATE, and ROTATE.




An object in its local space is transformed into global space by the current transformation matrix which may possibly represent multilevel relation between local and global space. In this example the local object is scaled down by 2, rotated by 45°, and translated into the first quadrant.

Figure 4-5. Transformation from Local to Global Space

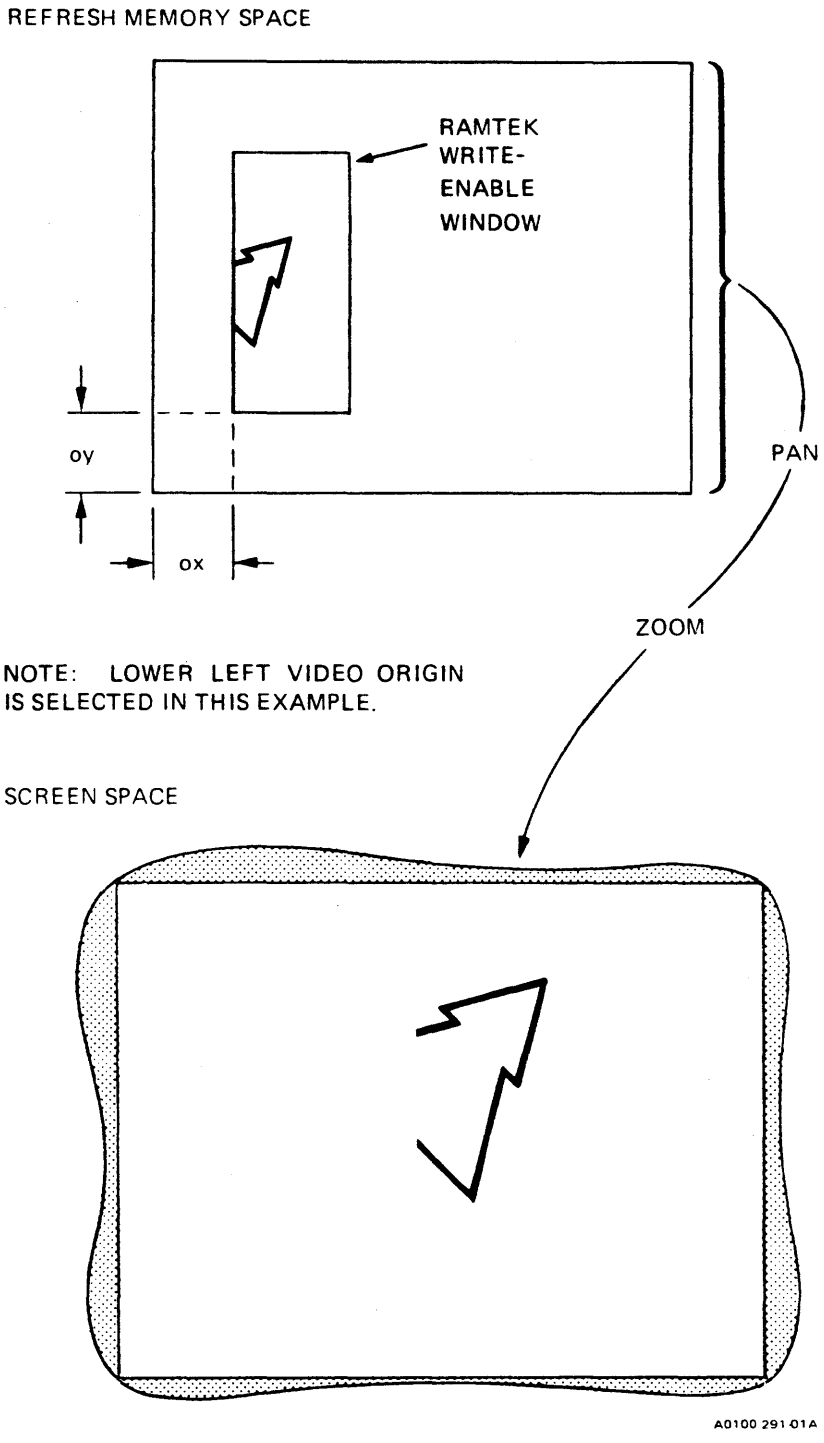



Figure 4-7. Refresh Memory for Screen Mapping Through Pan and Zoom

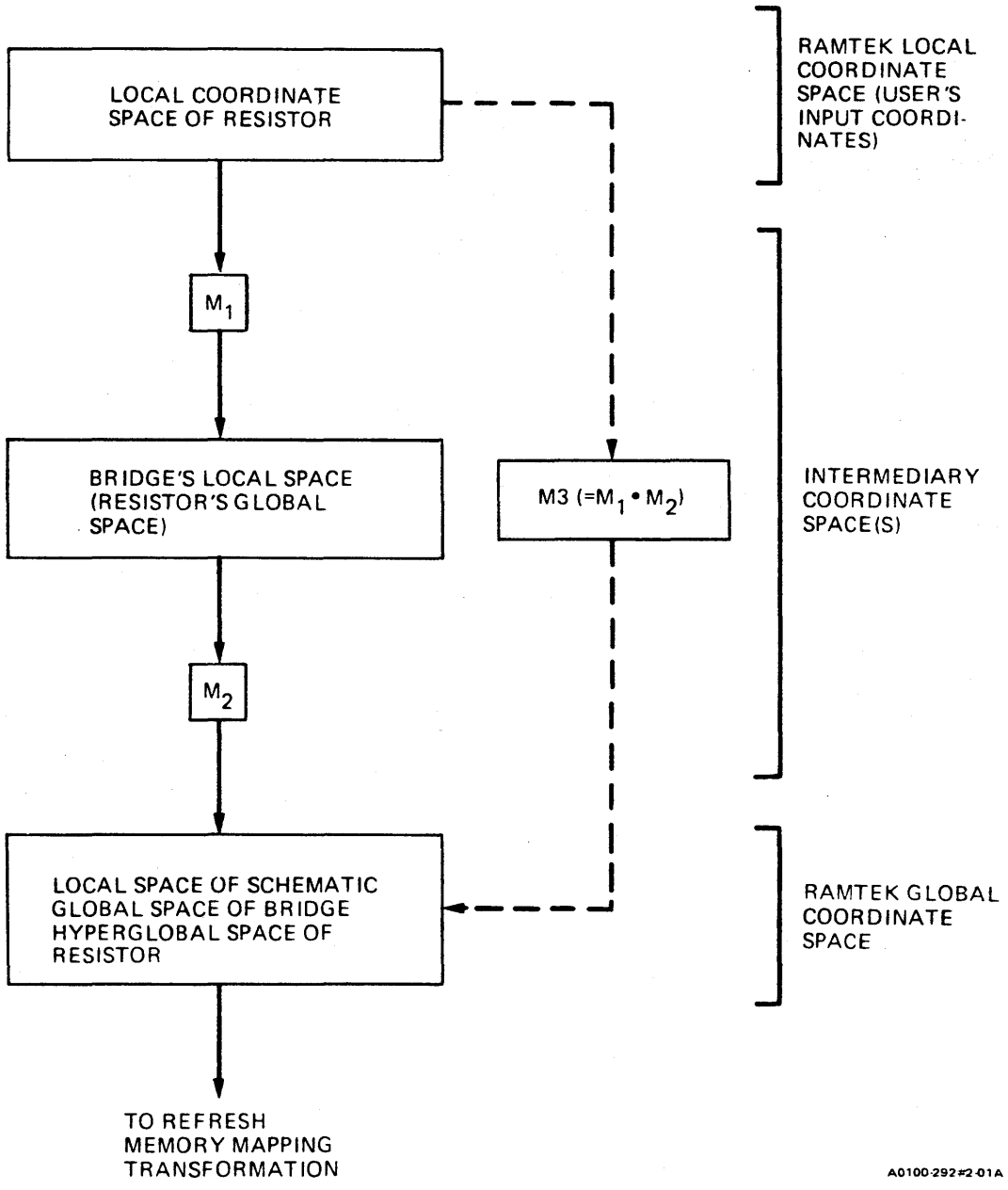
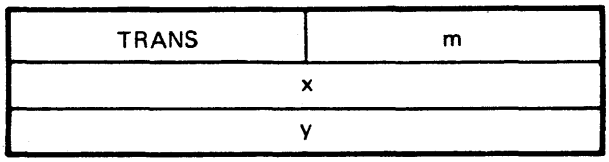



Figure 4-8. Nesting of Modeling Transformations and Resistor Mapping (Sheet 2 of 2)

### 4.6.4 Translate Instruction




B0100-227-01A

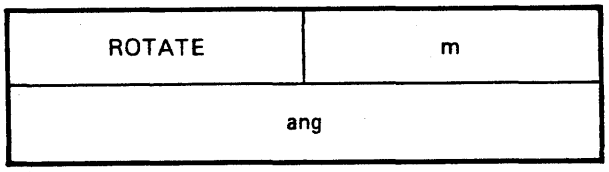
where: m = matrix register number
x and y are 16-bit two's-complement integer values.

The current coordinate system is translated along the vector (x,y). This is expressed as the following multiplication of the current matrix:

$$\underset{\text{Translation Matrix}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ M & N & 0 \end{bmatrix}} \times \underset{\text{Old Matrix}}{\begin{bmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{bmatrix}} = \underset{\text{New Matrix}}{\begin{bmatrix} a & d & 0 \\ b & e & 0 \\ aM+bN+c & dM+eN+f & 1 \end{bmatrix}}$$

where: M = x
N = y

### 4.6.5 Rotate Instruction




B0100-228-01A

where: m = matrix register number

This instruction causes the current coordinate system to be rotated about the current origin by an angle specified by ang. This number is a signed 16-bit two's-complement number. It represents the number of degrees to rotate. The instruction executes faster if ang is between 0 and 360. When it is necessary to specify angles more precisely than an integral number of degrees, the MULTIPLY MATRICES IMMEDIATE instruction can be used by the host computer.

A positive value of ang corresponds to a counterclockwise angle. The rotational transformation is implemented as the following matrix multiplication:

### 4.7.1 Clipping and Screen Management Parameter Setting

Once the user's coordinates have been transformed into Ramtek global space, the graphic elements are mapped through a clipping window (the WRITE-ENABLE WINDOW) that appears as a viewport on the display screen (figure 4-9). Because these windows must be the same size, six numbers are needed to describe the mapping.

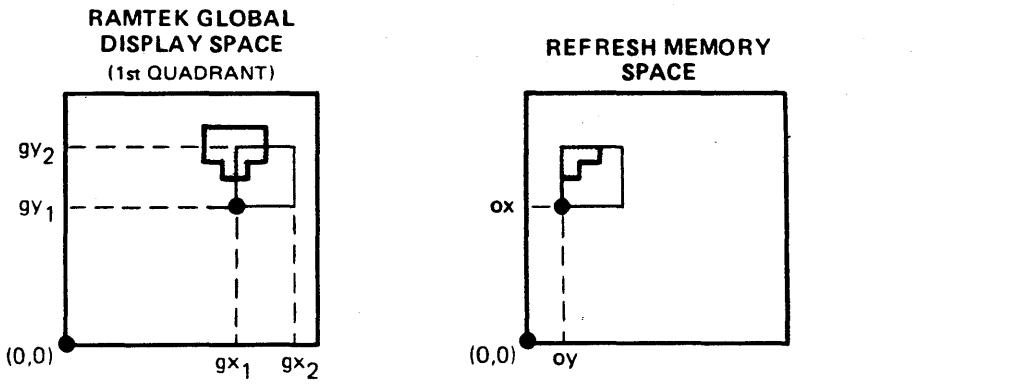



Figure 4-9. Global and Refresh Space

These six numbers are contained in two parameter blocks that can be set by any normal format instruction:

**WRITE-ENABLE WINDOW**

| |
|---|
| $gx_1$ (x minimum) |
| $gy_1$ (y minimum) |
| $gx_2$ (x maximum) |
| $gy_2$ (y maximum) |

**WRITE-ENABLE WINDOW OFFSET**

| |
|---|
| ox (minimum x) |
| oy (minimum y) |

B0100-229-01A

Specifically the coordinate transform matrix is set to be the identity matrix:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and the memory mapping parameters have these values:

$gx_1 = 0$

$gy_1 = 0$

$gx_2$ = refresh memory size$_x$ -1

$gy_2$ = refresh memory size$_y$ -1

$ox = 0$

$oy = 0$

B0100 295 01A

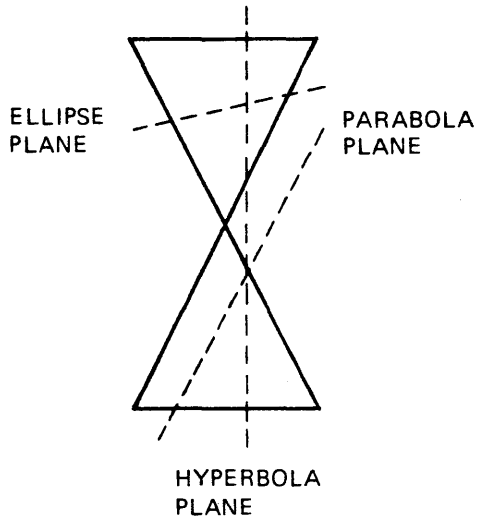
Figure 4-11. Default Transformation Values

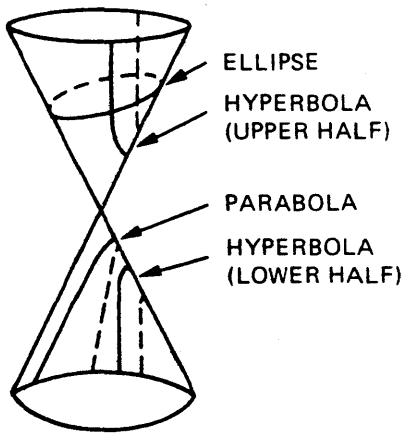
# Appendix A

# CONICS

## A.1 CONIC SECTION

A cone is the locus of a line rotated about an intersection line by a constant angle. A conic section, or more simply a conic, is defined as the intersection of the cone with a plane. Note that there are no planes in Euclidean space that do not intersect a given cone.




B0100 296 01A

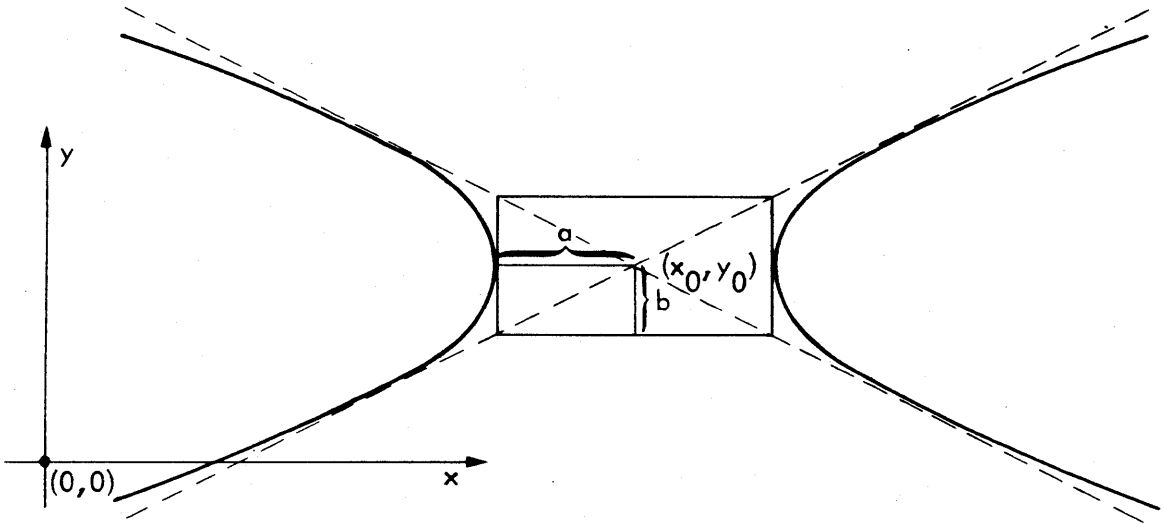
If a given point moves in a plane so that its distance from a fixed point (called the focus) divided by its distance from a fixed line (called the directrix) is a constant e (called eccentricity), then the curve described by the point is a conic.

There are three basic families of conics, differentiated as follows:

- e < 1 = ellipse
- e = 1 parabola
- e < 1 = hyperbola

A hyperbola can be described as:

$$\frac{(x - x_0)^2}{a^2} - \frac{(y - y_0)^2}{b^2} = 1$$




80100 298 01A

A parabola can be described as:

$(y-y_0)^2 = 4a(x - x_0)$ if the parabola opens to right and

$(y-y_0)^2 = -4a(x - x_0)$ if the parabola opens to left.

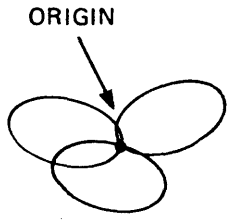
One advantage of using Cartesian coordinates is that all conic sections can be described in their plane of intersection as a second-order polynomial, of the form:

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0$$

If A = B = C = D = 0, the result is a straight line parallel to the x-axis. If E = 0 instead of D = 0, the line is parallel to the y-axis; if D and E are nonzero, the equation is that of some line slanted with respect to the axes.

If B and D are nonzero and A = C = E = F = 0, the result is a parabola that reflects across the x-axis.

The new constraint that the Cartesian origin must be chosen to lie on the conic locus does not bind to a single representation. There is still an infinite number of copies of the same eccentricity and orientation that pass through the origin as shown below:



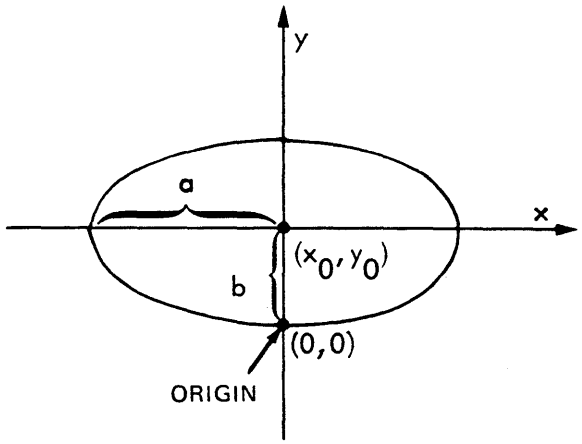

B0100 300 01A

One way to think of this is that the origin is being moved along the conic section. The origin can be placed anywhere along the conic and the corresponding equation fits the parameters. These new parameters are called Ramtek conic parameters, because this is the form accepted by Ramtek systems.

### A.2.1 Drawing an Ellipse

For simplicity, take as an example the case where the Cartesian origin is placed at the bottom of the ellipse. In this case, $x_0 = 0$ and $y_0 = b$.

Where: a = semimajor axis
b = semiminor axis
$(x_0, y_0)$ = center point



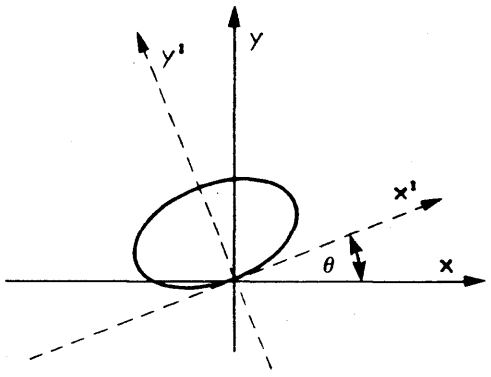

B0100 301 01A

### A.2.2 Translating the Ellipse

Translating the ellipse is easy. By changing the starting point (COP), the endpoint translates the ellipse. Note that changing the relative position of the starting point on the ellipse has the effect of translating the ellipse as well.

### A.2.3 Rotating the Ellipse

Rotating the ellipse requires changing the coefficients of the equation. Rotating the ellipse is equivalent to rotating the coordinate axes and drawing the ellipse again.




where:
$x' = x \cos \theta + y \sin \theta$
$y' = y \cos \theta - x \sin \theta$

80100 302 02A

Taking the old equation and putting it in terms of the new coordinates,

$$\frac{x'^2}{a^2} + \frac{y'^2}{b^2} - \frac{2y'b}{b^2} = 0$$

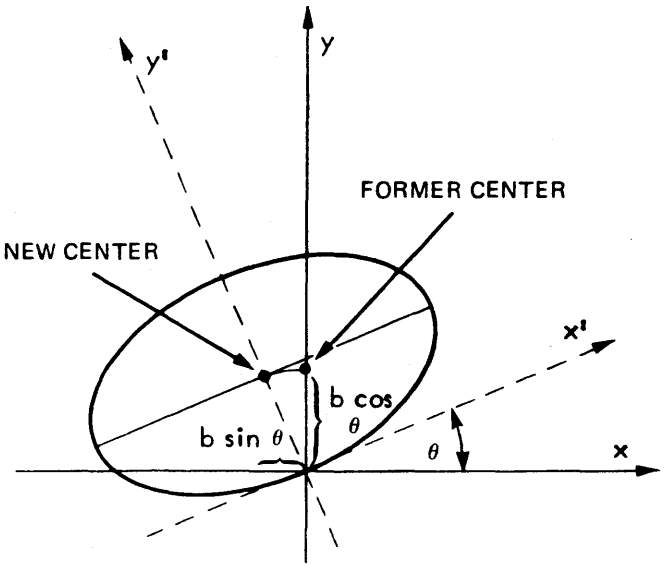
In terms of the old coordinates (the screen coordinates), the result is

$$\frac{(x \cos \theta + y \sin \theta)^2}{a^2} + \frac{(y \cos \theta - x \sin \theta)^2}{b^2} - \frac{2(y \cos \theta - x \sin \theta)b}{b^2}$$

The equation can be expanded to

$$\frac{x^2 \cos^2 \theta}{a^2} + \frac{y^2 \sin^2 \theta}{a^2} + \frac{2xy \cos \theta \sin \theta}{a^2} + \frac{y^2 \cos^2 \theta}{b^2} + \frac{x^2 \sin^2 \theta}{b^2}$$

$$- \frac{2xy \cos \theta \sin \theta}{b^2} - \frac{2y \cos \theta}{b} + \frac{2x \sin \theta}{b} = 0$$



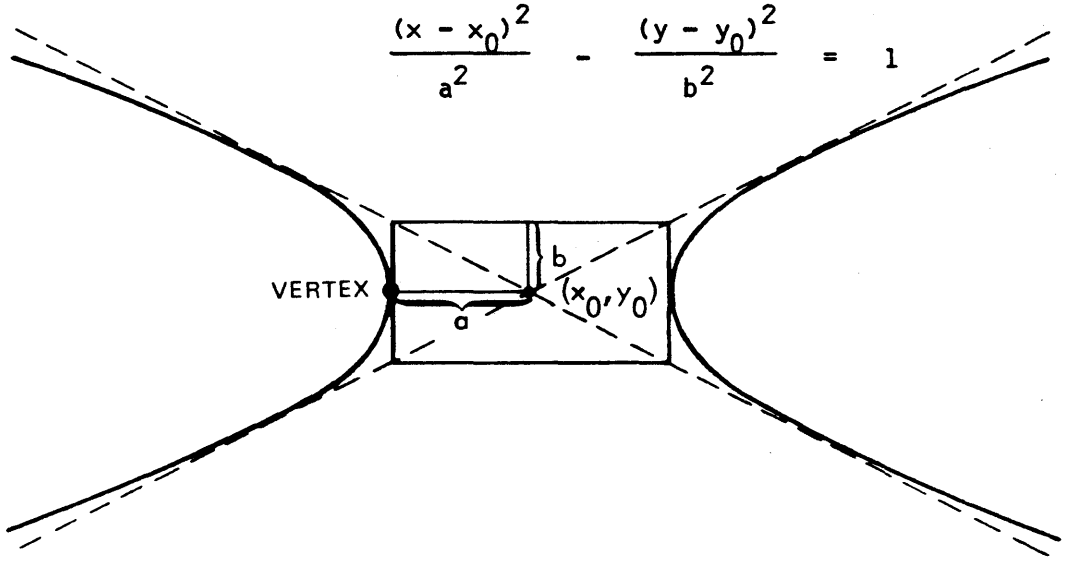

80100 303-02A

The center has been shifted by the last operation by b sin θ in the x-direction and b(1 - cos θ) in the y-direction. To rotate about the center, it must translate by this amount. The starting and ending point must be incremented by b sin θ in the x-direction and decremented by b(1 - cos θ) in the y-direction.

## A.3 THE HYPERBOLA

The same kind of analysis performed on the ellipse can be performed on the hyperbola. Recall the following equation for a hyperbola in terms of its semimajor and semiminor axes.

$$\frac{(x - x_0)^2}{a^2} - \frac{(y - y_0)^2}{b^2} = 1$$




80100 304-01A

At this point,

$$x_0 = a + d \quad \text{and} \quad y_0 = b\left(\frac{d^2}{a^2} + \frac{2d}{a}\right)^{1/2}$$

With this information put into the equation for the hyperbola, the result is

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} - \frac{2x\,(a + d)}{a^2} + \frac{2y}{b}\left(\frac{d^2}{a^2} + \frac{2d}{a}\right)^{1/2} = 0$$

which gives the coefficients in terms of the semimajor and seminor axes with the additional parameter of the distance along the axis from the origin to the vertex.

$$A = 1/a^2$$

$$B = -1/b^2$$

$$C = 0$$

$$D = -\frac{2(a + d)}{a^2}$$

$$E = \frac{2}{b}\left(\frac{d^2}{a^2} + \frac{2d}{a}\right)^{1/2}$$

## A.4 TRANSLATING AND ROTATING THE HYPERBOLA

Translating the hyperbola is simply a matter of where to start drawing on the screen. Rotation involves transforming the coefficients in the same way as before, substituting

$$x' = x \cos\theta + y \sin\theta$$
$$y' = y \cos\theta - x \sin\theta$$

thus,

$$\frac{x'^2}{a^2} - \frac{y'^2}{b^2} - \frac{2x'(a + d)}{a^2} + \frac{2y'}{b}\left(\frac{d^2}{a^2} + \frac{2d}{a}\right)^{1/2} = 0$$

## A.5 FINAL POINTS TO CONSIDER

Three assumptions were made in using Cartesian representations:

1. All points in space can be represented in terms of n variables, where n is the number of dimensions of the Cartesian space. These variables correspond to the tick marks along the orthonormal axes.

2. Space is infinite in extent.

3. Space is infinitely divisible.

Each of these three assumptions is violated on all computer graphics systems.

The first assumption, that an orthonormal coordinate system is being dealt with, is not accurate. It is not always a 1:1 screen ratio. This leads to severe problems when considering that object circles can appear to grow and retreat as the angle increases. It is usually necessary to consider the screen aspect ratio when determining how the conic section is to be drawn.

The second assumption of our Cartesian representation, that space is infinite, is obviously limited. The screen can only be drawn on. Although internal software allows the conic to curve back onto the screen after once leaving, be aware that ending the conic off-screen leaves the COP off-screen, which is generally undesirable.

The third assumption, that any interval is infinitely divisible, is perhaps the most troublesome. Actually it is a quantized space that simulates Cartesian space as closely as possible. Sometimes this is not good enough. Mathematical calculation may show that a certain point should be activated by issuing a certain conic command, but when the command is given, the actual pixel activated may be off by one. If the desired endpoint is never activated, the drawing automatically stops if the desired point was not reached in 1280 attempts. This value can be increased or decreased depending on the user's needs.

# Appendix B

# VIDEO GENERATORS

## B.1 INTRODUCTION

The video generators transform stored pictures into industry-compatible video, and signals that drive Ramtek and other commercially available CRT monitors, large screen projectors, and hardcopy printers. They process data on a pixel-by-pixel basis through PROM- or RAM-defined lookup tables that assign output color and/or intensity. This is simply done as each pixel indexes the lookup table while the data is scanned from refresh memory. The contents of the addressed cell in the lookup table are then passed to the digital-to-analog converters (DAC) and/or video amplifiers that produce the output video signals.

Cursor and overlay mixing is performed either in the lookup table or at the DAC by clamping the output voltage to a minimum or maximum scale. All video generators incorporate a blink frequency generator that allows output color and/or intensity to be defined as static or blinking.

## B.2 PROGRAMMABLE LOOKUP TABLES

Two of the three standard video generators incorporate programmable video lookup tables that can be loaded by the host computer. While the lookup table dimensions vary according to video option and system resolution, the mechanism for loading and retrieving data does not.

The video lookup tables are randomly accessible by the load and read auxiliary memory instructions (LAM and RAM). These special-format instructions carry an 8-bit code that selects the appropriate lookup table and a 16-bit field that presets the lookup table address register that controls host computer access to the lookup table. The lookup table address register is automatically incremented by one after each data word is transferred to or from the lookup table. Thus, the address register points to location n+1 after a load or read sequence, where n points to the location where the last word of the sequence was accessed. This is an important consideration in systems where fewer than the maximum number of refresh memory bit planes are processed by the lookup table since the unused lookup table inputs are provided by the corresponding bits in the lookup table address register. Thus, the lookup table is partitioned into two or more subtables, with one subtable being selected (for video processing) by the lookup table address register at any given time. The length and number of subtables is a function of the number of refresh planes being processed by the lookup table and the physical length of the lookup table itself, respectively. For example, if eight planes (0-7) are processed by a 2048 lookup table, then any of eight 256-word subtables may be selected by lookup table address register bits 8 through 10. Because the lookup table address register is incremented by one after each transaction, the programmer is cautioned that the next sequential subtable is automatically selected (for

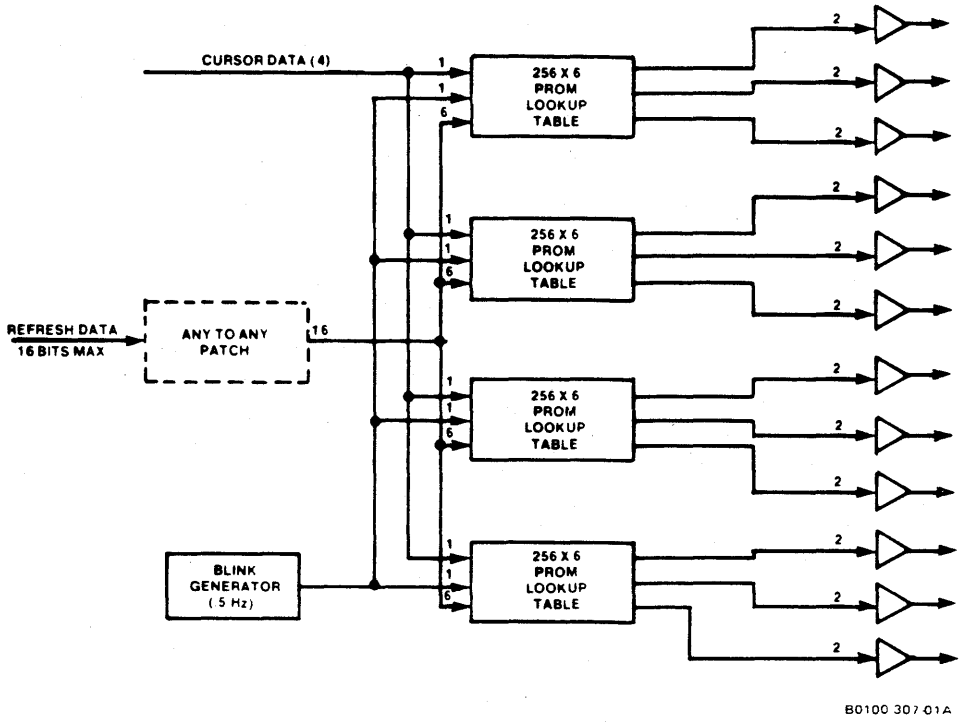



Figure B-1. RM-9460-V1 Video Generator

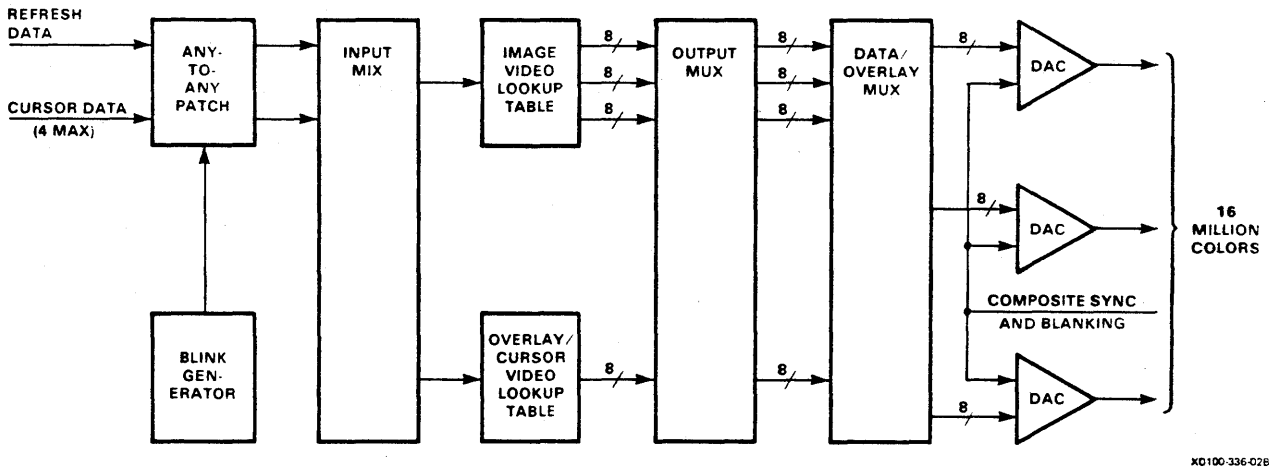



Figure B-2. RM-9460-V6 Video Generator

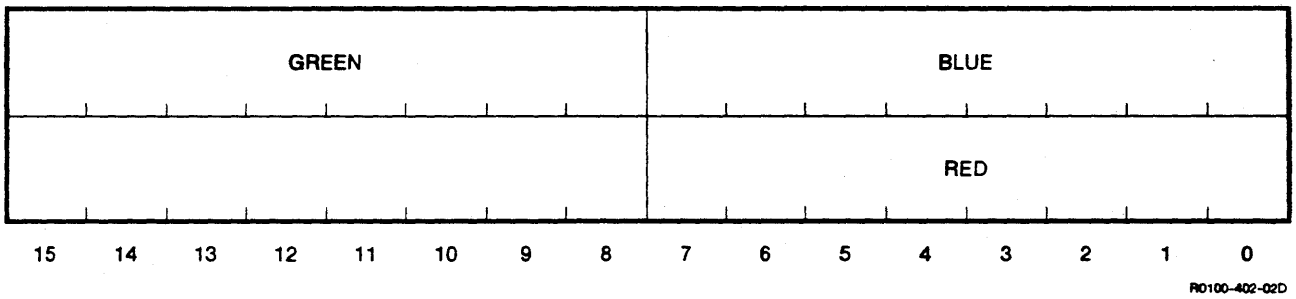
## B.5 TYPE 6B VIDEO GENERATOR (RM-9460-V6B)

The RM-9460-V6B (V6B) is designed for general imaging and sophisticated graphics applications (figure B-3). It provides for hardware blink and mixes an overlay and up to four independent cursors with the video outputs.

The V6B drives three 8-bit (256 level) outputs to a single RGB color display through two lookup tables (LUTs), either of which can be disabled, or function as either an image or overlay lookup table. At power-up, both lookup tables are enabled, with LUT 2 functioning as the overlay lookup table and LUT 1 functioning as the image lookup table. This configuration can be changed by sending the LAM instruction with a data length word of zero and the following starting addresses:

| Starting Address | Result |
|---|---|
| 8000(H) | LUT 2 will overlay LUT 1. |
| 8001(H) | LUT 2 will be disabled. |
| 8002(H) | LUT 1 will be disabled. |
| 8003(H) | LUT 1 will overlay LUT 2. |

Each lookup table is 1024 X 24 bits, allowing the user to program any of 256 output intensities for each of the primary colors, thus providing a selection of more than 16 million colors. The data format for loading the lookup tables is:

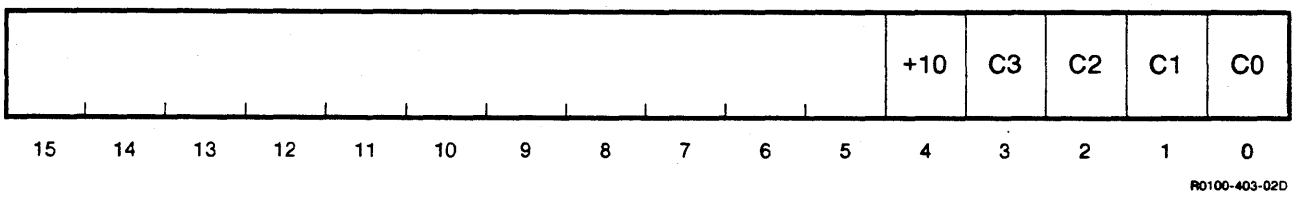



LUT 1 consists of physical addresses 0 - 2047. LUT 2 consists of physical addresses 2048 - 4095. Each refresh memory input value (logical address) will access two host-specified 16-bit color data words. Therefore, to change the color associated with any particular refresh memory input value to LUT 1, the user must send two color data words in the LAM instruction, using double the input value as the LUT address. To change the color associated with an input value to LUT 2, the user must send two color data words in the LAM instruction, using double the input value plus 2048 as the LUT address. Each table, from the user-loading viewpoint, looks like a 2048 X 16-bit table, with two sequential locations describing the 16 bits of color data for each refresh memory input value.

The V6B has three mask registers. These may be loaded or disabled using the LAM instruction with specified starting addresses, as follows:

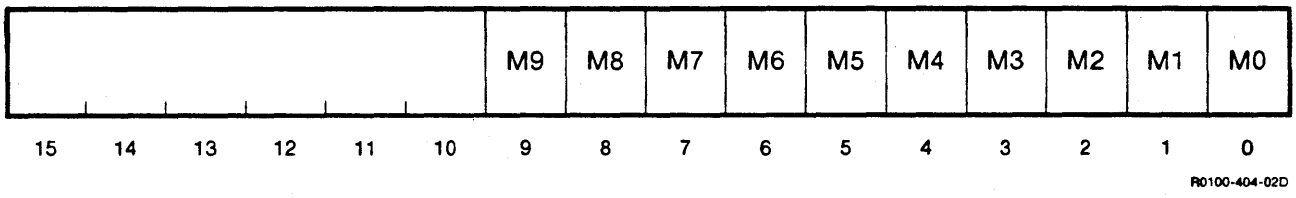
| Starting Address | Result |
|---|---|
| 8004(H) | Loads Cursor/+10% mask |
| 8005(H) | Loads LUT 2 mask |
| 8006(H) | Loads LUT 1 mask |
| 8007(H) | Disables all masks |

The data format for loading the Cursor/+10% mask is as follows:



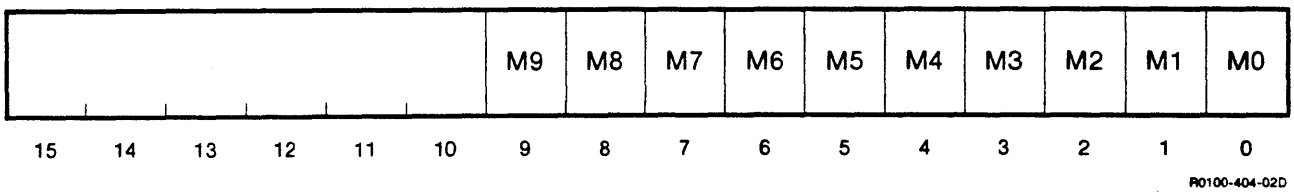

where bit 4 (+10), if not set, will mask the +10% intensity of the overlay, and bits 0-3 (C0-C3), if not set, will mask cursors 0-3, respectively.

The data format for loading the LUT 2 mask is as follows:




where bits 0-9 (M0-M9), if not set, will mask memory planes 0-9, respectively, to LUT 2.

The data format for loading the LUT 1 mask is as follows:




where bits 0-9 (M0-M9), if not set, will mask memory planes 0-9, respectively, to LUT 1.

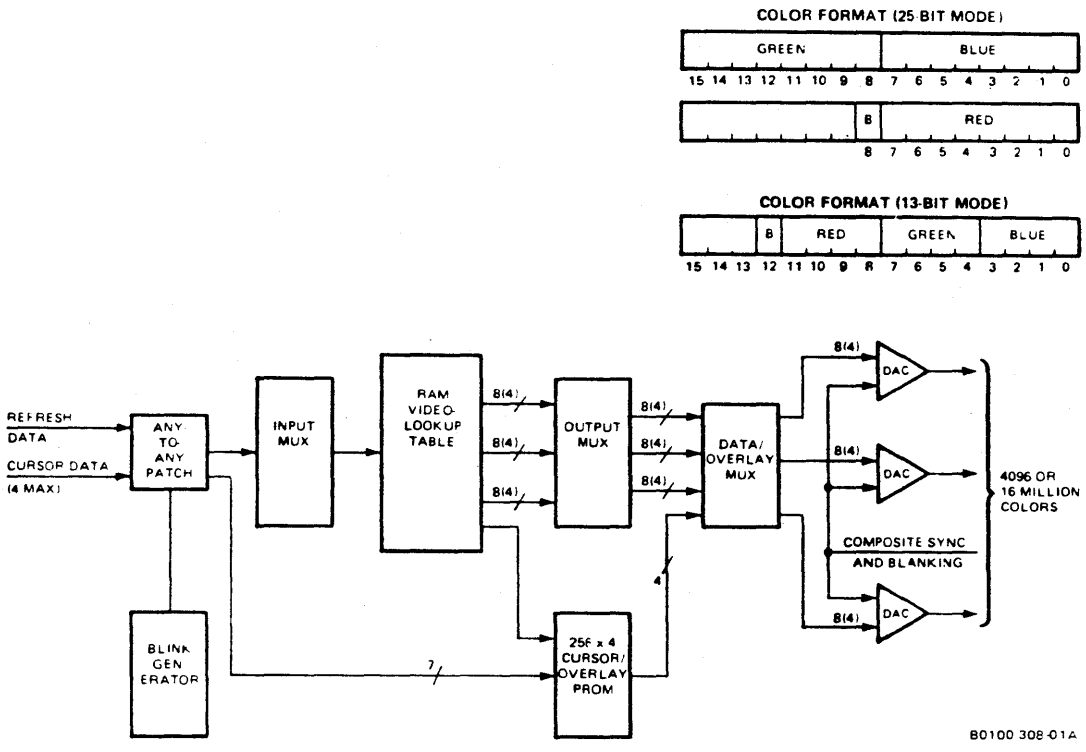
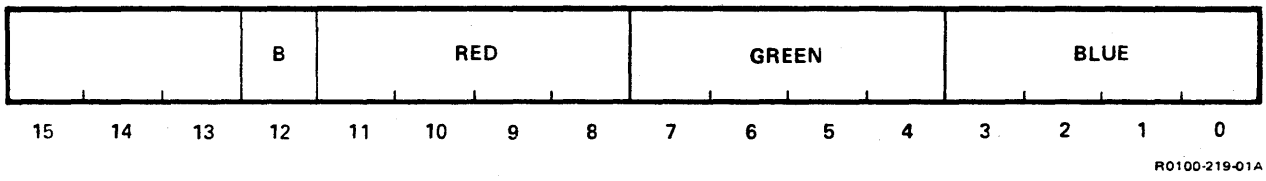



Figure B-4. RM-9460-V7A/B Video Generator

The lookup table format for the 13-bit format is:




The lookup table contains an additional bit. When true, this bit causes the specified color to blink to black at a 2-Hz rate.

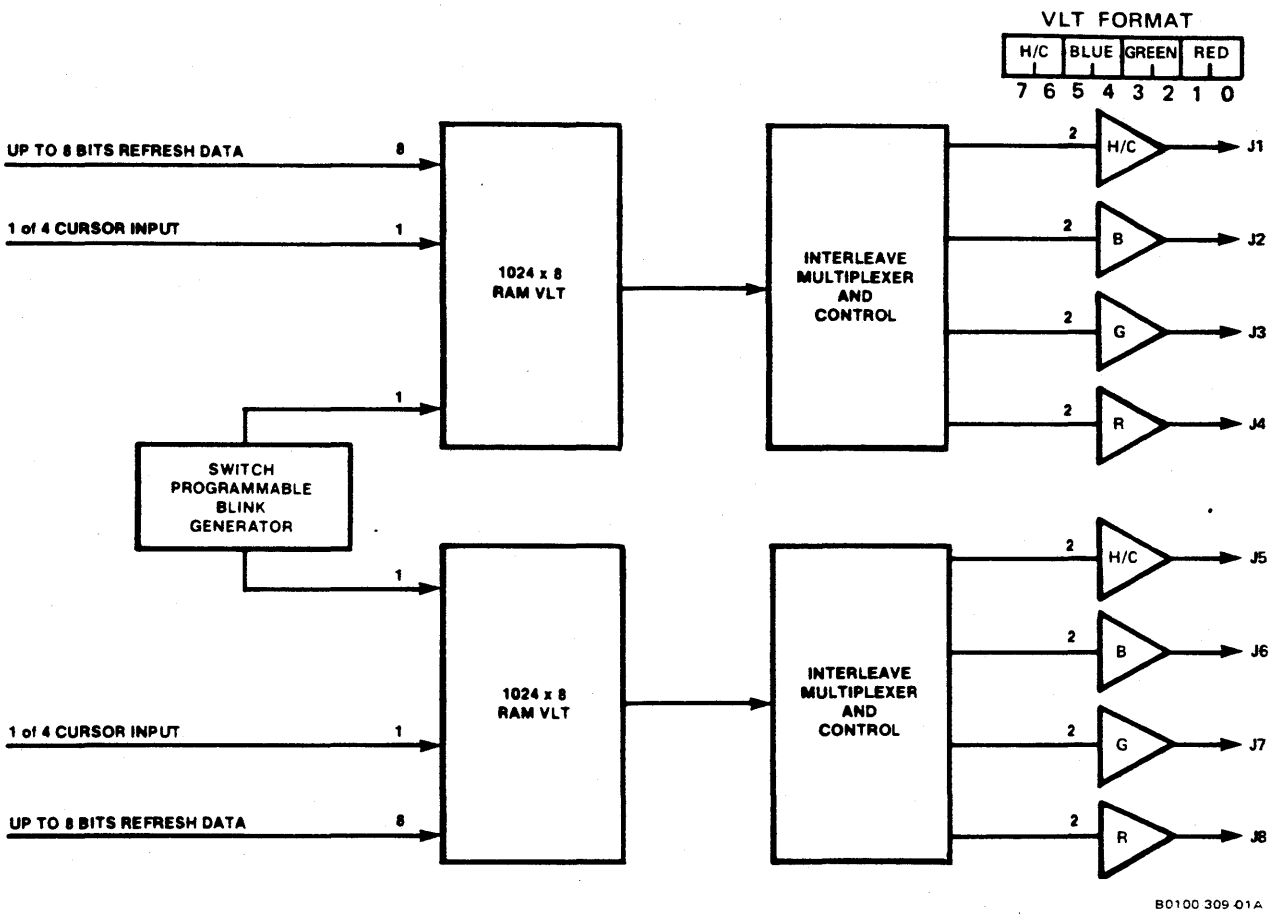



Figure B-5. RM-9460-V8 Video Generator

Cursor mixing and blink control are accomplished in the lookup table. For example, blink frequency (from blink generator) and the cursor symbol (from the cursor generator) are input directly to the lookup table as address bits 9 and 8, respectively. This allows the programmer not only to define the cursor color, but also to specify both phases of the blink cycle. For example, a color might be defined as flashing between high and low intensity red, or between red and yellow, etc. Refresh data and lookup table address register data are input to the lookup table as address bits 0 through 7; thus, the lookup table is divided into four sections as shown in figure B-6. Partitioning, when fewer than 8 refresh planes are processed, further divides each 256-word section of the lookup table.

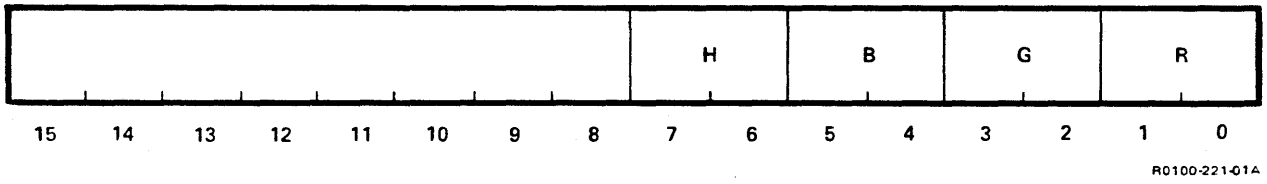
## B.8 TYPE 12 VIDEO GENERATOR (RM-9460-V12A/B)

The RM-9460-V12 generates video (graphics) and text (character) analog outputs from digital inputs. Video and text outputs can be displayed on a common monitor or on separate monitors. If the video and text outputs are displayed on separate monitors, both monitors must be high resolution monitors. The RM-9460-V12 provides hardware blink for video and text independent of each other. Jumpers allow the user to select either one overlay and one of four cursor inputs or two overlays and no cursor.

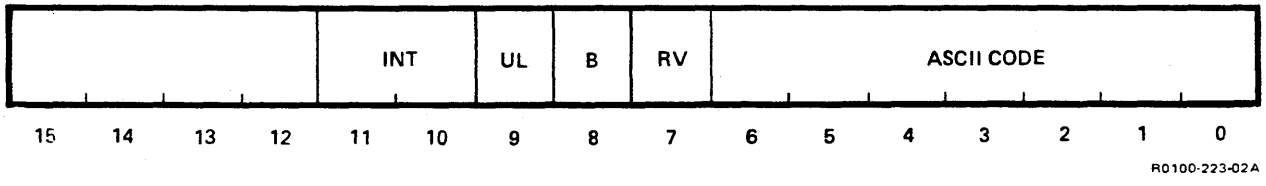
### B.8.1 RM-9460-V12A

The RM-9460-V12A (figure B-7) contains four 2-bit (4-level) digital-to-analog converters (DACs) and operates in one of two modes. In mode 1 three DACs drive a color monitor and one DAC drives a hardcopy device. In this mode the color monitor displays both video and text outputs. In mode 2 three DACs drive a color monitor and one DAC drives a monochrome monitor. In this mode the color monitor displays refresh memory video output and the monochrome monitor displays the text generator output.

The RM-9460-V12A contains a 256 x 8-bit video lookup table (VLT). The VLT allows the user to select from any of 4 output intensities for each primary color, for a total of 64 different colors. In mode 1 the VLT also allows the user to select from any of four output intensities for a hardcopy device. VLT data input format consists of eight bits that describe the relative intensity of four analog outputs: red (R), green (G), blue (B), and hardcopy (H). The format is as follows:



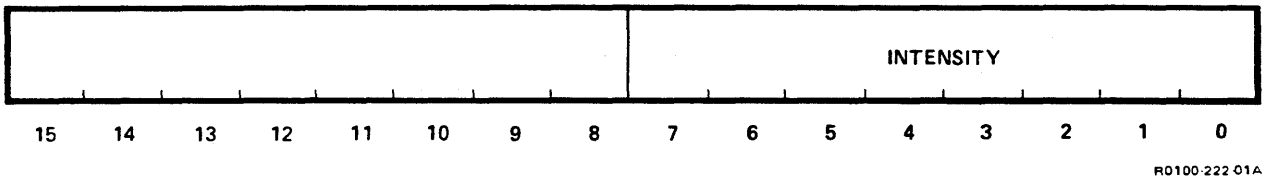

The RM-9460-V12A contains a character-generator lookup table consisting of a 2048 by 12-bit RAM array. The character-generator lookup table allows the user to select from 128 characters using ASCII code and to specify any of the following attributes for each character: intensity (INT), underline (UL), blink (B), and reverse video (RV). The LAM instruction data input format for the character-generator lookup table is as follows:



### B.8.2 RM-9460-V12B

The RM-9460-V12B (figure B-8) contains one 8-bit (64-level) DAC and one 2-bit DAC and can operate in one of two modes. In mode 1 the 8-bit DAC drives a monochrome monitor and displays the video and text outputs. In mode 2 the 8-bit DAC drives a monochrome monitor displaying only video output and the 2-bit DAC drives a separate monochrome monitor displaying only text output.

The RM-9460-V12B contains a VLT identical to the RM-9460-V12A VLT except that the RM-9460-V12B VLT allows the user to select from any of the 256 gray scale intensity levels. The VLT data input format for the LAM instruction is as follows:

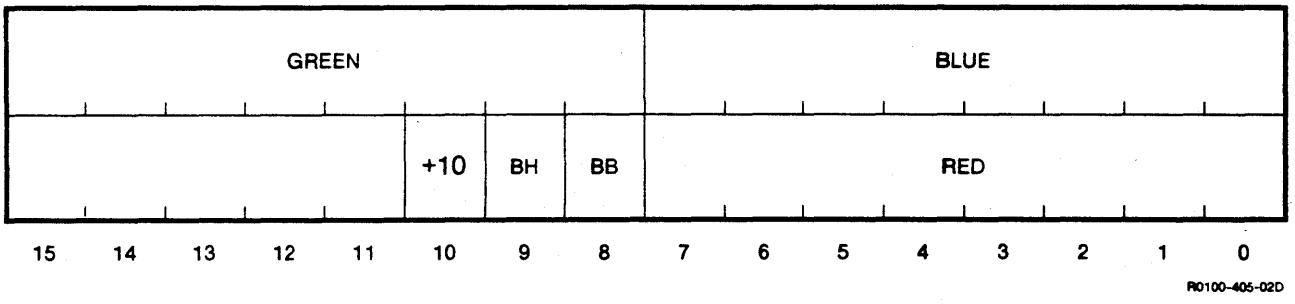



The RM-9460-V12B contains a character-generator lookup table identical to that contained in the RM-9460-V12A.

### B.9 TYPE 17 VIDEO GENERATOR (RM-9460-V17)

The RM-9460-V17 (V17) is designed for general imaging and sophisticated graphics applications on a RM-9465/04 (figure B-9) It provides hardware blink and mixes up to four independent cursors with the video outputs. Cursors can be either fixed or programmable.

The V17 drives three 8-bit (256 level) outputs to a single RGB color display through an image lookup table and optionally a cursor map table. The 1024 X 28-bit programmable image lookup table allows the user to program any of 256 output intensities for each of the primary colors, thereby allowing the user to select from more than 16 million colors. The data format for the image lookup table is:



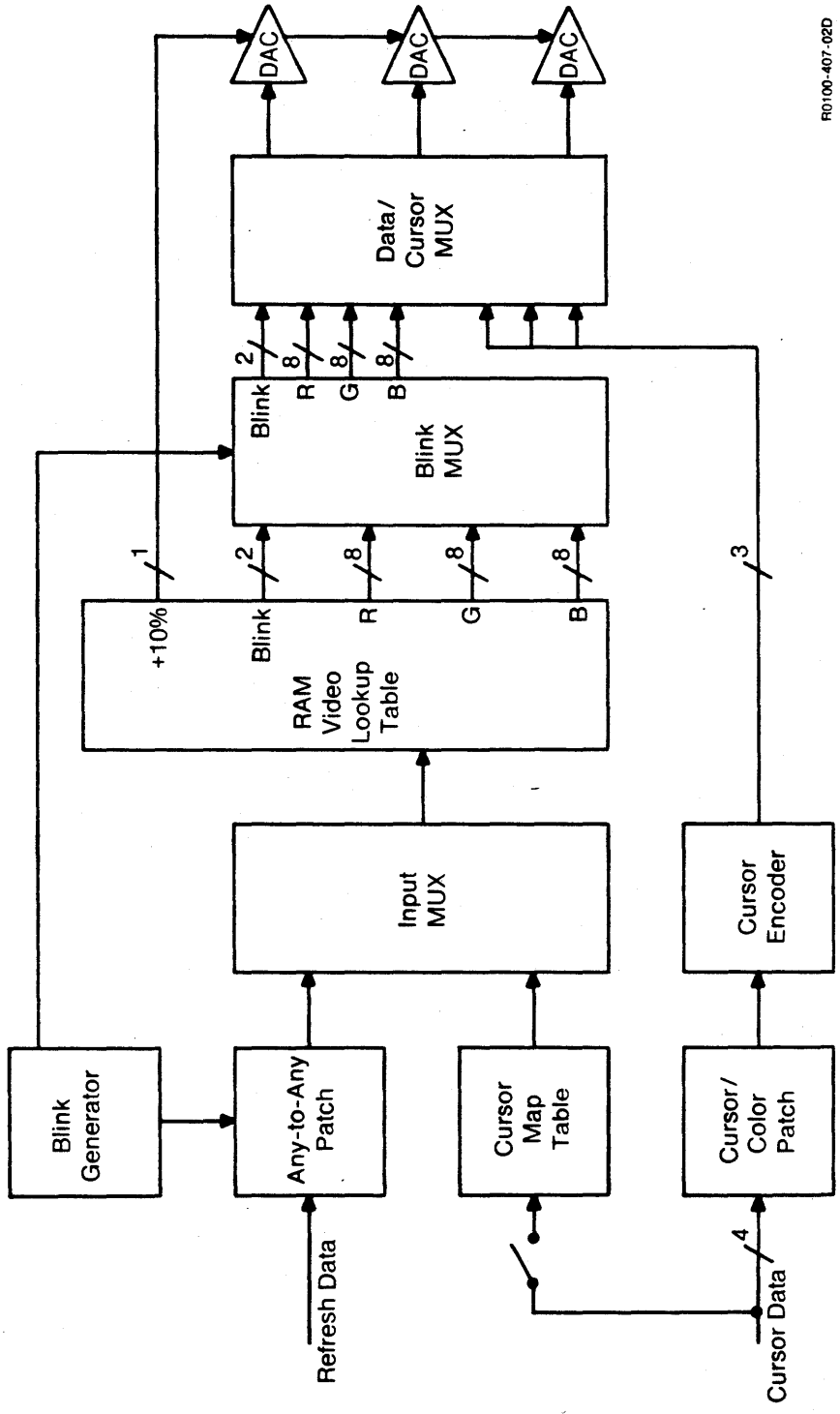



Figure B-9. RM-9460-V17 Video Generator

| Logical Address | Cursor Combination | Physical Address |
|---|---|---|
| 0 | none | 2048 |
| 1 | 0 | 2049 |
| 2 | 1 | 2050 |
| 3 | 0, 1 | 2051 |
| 4 | 2 | 2052 |
| 5 | 0, 2 | 2053 |
| 6 | 1, 2 | 2054 |
| 7 | 0, 1, 2 | 2055 |
| 8 | 3 | 2056 |
| 9 | 0, 3 | 2057 |
| 10 | 1, 3 | 2058 |
| 11 | 0, 1, 3 | 2059 |
| 12 | 2, 3 | 2060 |
| 13 | 0, 2, 3 | 2061 |
| 14 | 1, 2, 3 | 2062 |
| 15 | 0, 1, 2, 3 | 2063 |

# Appendix C

# INTERACTIVE PERIPHERALS

## C.1 INTRODUCTION

The RM-9460 supports a variety of interactive peripheral devices through serial ports on the basic processor and optional serial link PCBs. The support logic is provided as a combination of hardware and control software (firmware) that simplifies the man-machine interface. Support is provided for general-purpose keyboards, joysticks, trackballs, light pens, and digitizing tablets.

## C.2 BASIC CAPABILITY

The basic capability of the RM-9460 depends on whether the RM-9460 contains a Z80 or an MC68000 system processor. The basic RM-9460 with a Z80 system processor supports a single keyboard and either a joystick or trackball cursor controller. The basic RM-9460 with an MC68000 system processor supports a single keyboard and a single cursor. The cursor can be controlled by either a joystick/trackball or a graphic tablet. On both the Z80 and MC68000 systems, a single PROM-defined cursor symbol is produced; that is, normally a crosshair with the target element(s) missing. A cursor-tracking algorithm is implemented so that the operator can steer about the face of the display without host computer interaction and without affecting the picture in refresh memory. Separate controls on the cursor controller allow the operator to interrupt the host computer either on a momentary basis (ENTER switch) or on a continuous basis (TRACK switch) each time that the cursor position changes.

### C.2.1 Peripheral Ports on System Processor PCB

Both the Z80 and MC68000 system processors support a keyboard device, which must be connected to peripheral port J1, a differential port. Both the Z80 and MC68000 system processors support a joystick/trackball device, which must be connected to peripheral port J2, a differential port. Only the MC68000 system processor supports a graphic tablet device, which must be connected to peripheral port J3, an RS-232 port. Peripheral port J3 is not used on the Z80 system processor. All peripheral ports on the Z80 and MC68000 system processors are set up at 9600 fixed baud rate.

Two switches in dip switch pack 4X on the MC68000 system processor control the graphic tablet device; switch 5 controls the existence of the tablet device, and switch 4 controls whether the tablet is in menu or non-menu mode. Table C-0 lists the selections of switches 4 and 5 on dip switch pack 4X.

Table C-2. Device Selection

| DEVICE | SWITCH LOCATION | | |
|---|---|---|---|
| | 7C | 7F | 7K |
| Keyboard | OFF | OFF | OFF |
| Joystick/Trackball | ON | OFF | OFF |
| Lightpen | OFF | ON | OFF |
| Tablet (without menu) | ON | ON | OFF |
| Tablet (with menu) | OFF | OFF | ON |
| Unused | ON | OFF | ON |
| Unused | OFF | ON | ON |
| Unused | ON | ON | ON |

Baud rate selection (table C-3) is accomplished by setting switches at locations 7M and 7R to correspond to desired baud rate.

Table C-3. Baud Rate Selection

| BAUD RATE | SWITCH LOCATION | |
|---|---|---|
| | 7M | 7R |
| 1200 | OFF | OFF |
| 2400 | ON | OFF |
| 4800 | OFF | ON |
| 9600 | ON | ON |

### C.3.2 Programming Considerations

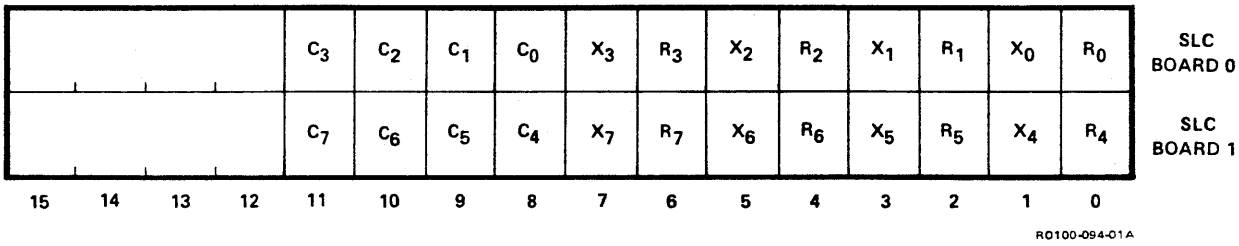
In addition to the capabilities provided by the basic RM-9460, the serial link PCB allows the user to program the cursor symbol, specify the device-to-cursor association, specify the cursor-to-video output association, and use either a light pen or graphic tablet cursor control device.

### C.3.3 Display List Firmware

The RM-9460 display list firmware allows local processing of keyboard and cursor control input whether or not an serial link PCB is installed. For example, local subroutines may be downline-loaded into display list memory and linked to function keys on the general purpose keyboard. Thus, the linked subroutine is executed when the corresponding function key is pressed. Additional logic allows local echoing of alphanumeric input and local access to cursor input.

## C.7 PERIPHERAL INTERRUPTS

Peripheral activity represents one of several interrupts transmitted to the host computer. To determine the nature of the received interrupt, the host computer must first read the INTERFACE STATUS WORD (ISW). The ISW format varies according to the computer interface but typically indicates the peripheral activity versus illegal instruction, etc. By reading the ISW, the interrupt bits reset to zero and further interrupts are enabled from the display system. After determining that a peripheral interrupt has occurred, the interrupt service routine next senses peripheral status by executing a SENSE PERIPHERAL STATUS (SPS) instruction for each of the installed serial link PCB options or for the basic system, which is treated as SLC0. The format of the response is as follows:




where $C_n$ = 1 flags cursor activity, $R_n$ = 1 flags keyboard input, and $X_n$ = 1 flags transmitter output completed (normally to a keyboard).

The interrupt service routine completes its function by reading or writing to the corresponding cursor, receiver, or transmitter register. You must send the SPS instruction to the RM-9460 only on an instruction boundary. Generally the interrupt service routine will queue I/O requests to perform the above actions and complete the peripheral interrupt processing when the current I/O is completed.

## C.8 GC-106 JOYSTICK CURSOR CONTROLLER

The joystick cursor controller (figure C-1) is an interactive peripheral device used to position a cursor upon a video graphic display. The cursor controller consists of a joystick, four status switches (ENTER, TRACK, VISIBLE and BLINK), four channel select switches, and a power switch. The controller interactively positions the cursor through the joystick, controls cursor status with the VISIBLE and BLINK status switches, and informs the computer of current coordinates and status by the ENTER and TRACK switches.

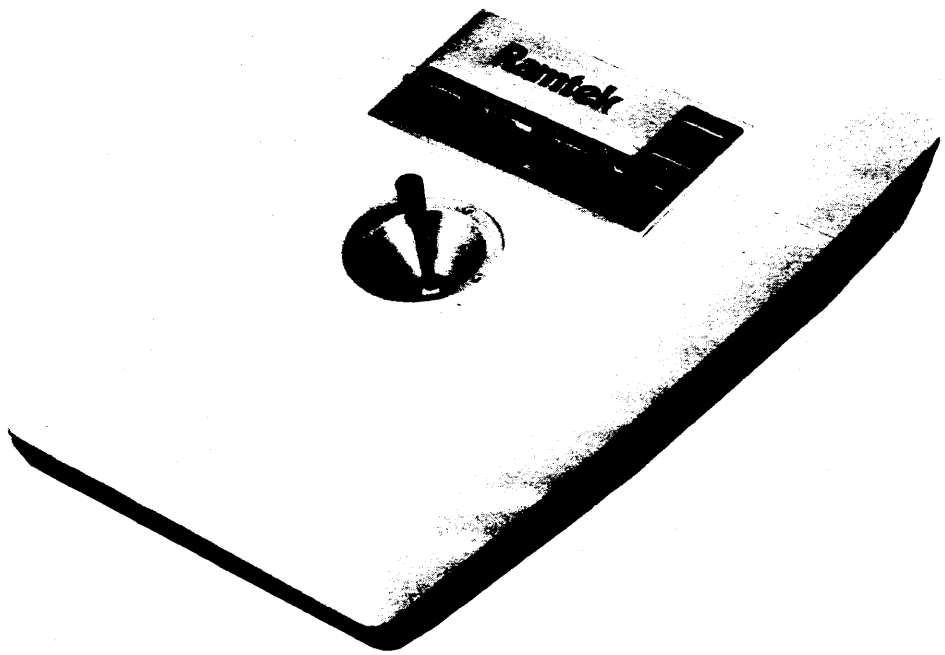



Figure C-1. GC-106 Joystick Cursor Controller

The cursor controller and trackball operate with the serial link option in an identical manner. Both use serial transmission lines to send data. The basic processor or serial link option stores cursor coordinates and status while generating the cursor video image. The cursor controller does not store cursor coordinates but issues increment/decrement commands to the serial link board which, in turn, updates the cursor position on the screen. Since the amount of displacement of the joystick from the center position changes proportionately the rate of increment/decrement commands issued by the cursor

### C.8.1 Joystick Status Control Switches

Four status switches determine the status of the cursor on the screen and control host computer interrupt generation. These switches are as follows:

- VISIBLE - an alternate action switch that turns the cursor ON or OFF. Cursor coordinates are not affected by the position of this switch.

- BLINK - an alternate action switch that, when ON, causes the cursor to blink at approximately 1-Hz rate. When BLINK is OFF, the cursor remains steady on the screen. Cursor coordinates remain unaffected by the position of the BLINK switch.

- ENTER - a momentary-action switch that causes a cursor interrupt, when enabled, to be sent to the computer regardless of the position of any status switch or the position of the joystick. If the ENTER switch is held ON, the cursor controller ceases to function until the switch is released. As soon as the ENTER switch is released, the cursor controller resumes normal operation.

- TRACK - an alternate action switch that, when ON, causes every movement of the cursor to generate a host computer interrupt. Every movement of the cursor is defined as a change in coordinates. When the TRACK switch is OFF, the cursor continues to move on the screen, but the cursor interrupt is not issued to the host computer.

As long as the joystick is centered and the status switches are stationary, the channel-select switches can be changed with no effect. Power does not need to be OFF to change the channel select switches.

### C.8.2 Joystick Cursor Selection Switches

Using the four channel-select switches, the operator can control up to four cursors simultaneously with one cursor controller unit. These alternate-action switches cause the output of the controller to be distributed to the output channel(s) selected by the switches. When a switch for any channel is ON, the output of the controller appears on the serial output for that channel. When the switch is OFF, the serial output for that channel goes to an idle or no transmission mode. Any combination of switches can be ON simultaneously, including all ON or all OFF.

**NOTE**

Channel-select switches should never be changed while moving the cursor with the joystick or while switching the status switches. Since the controller operates with a serial output line changing the channel-select switches while the unit is transmitting may cause unpredictable results or cursor movement or status.

## C.10 GC-105 LIGHT PEN CURSOR CONTROLLER

The GC-105 Light Pen (figure C-3) differs from other cursor control devices in that the operator points a hand-held pen directly at the CRT. The pen itself consists of a photodetector that senses light emissions from the CRT (as phosphors are excited by the electron beam) and a tip-mounted actuation switch. Ancillary electronics receive a composite sync signal from the display generator and the actuation and detect pulses from the light pen. These electronics digitize and serially transmit the actuation event and pen position to cursor control ports on the serial link/cursor option.

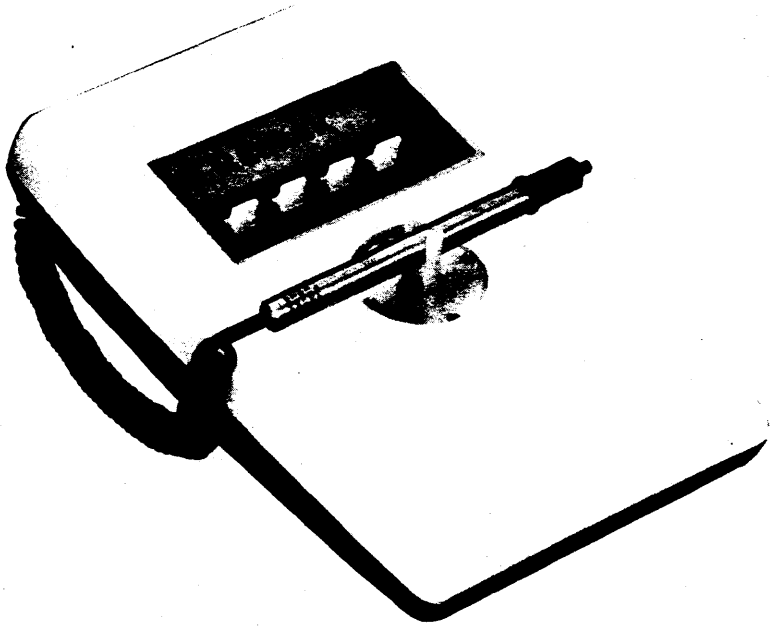
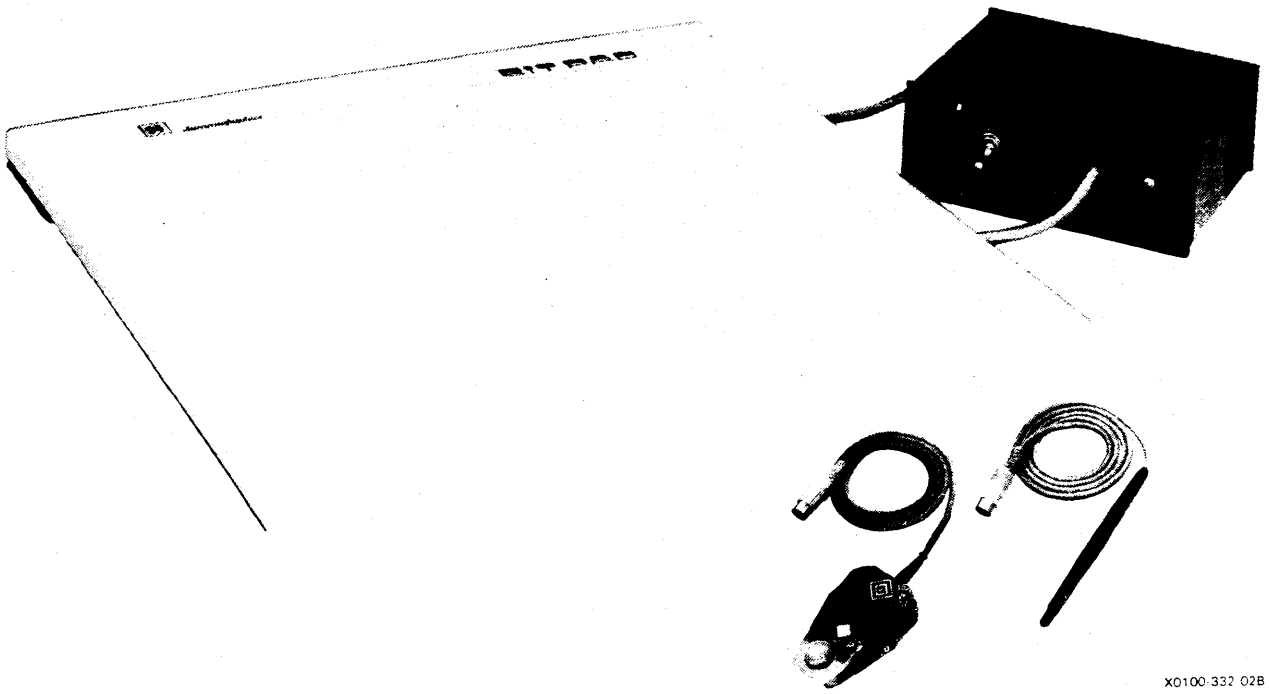


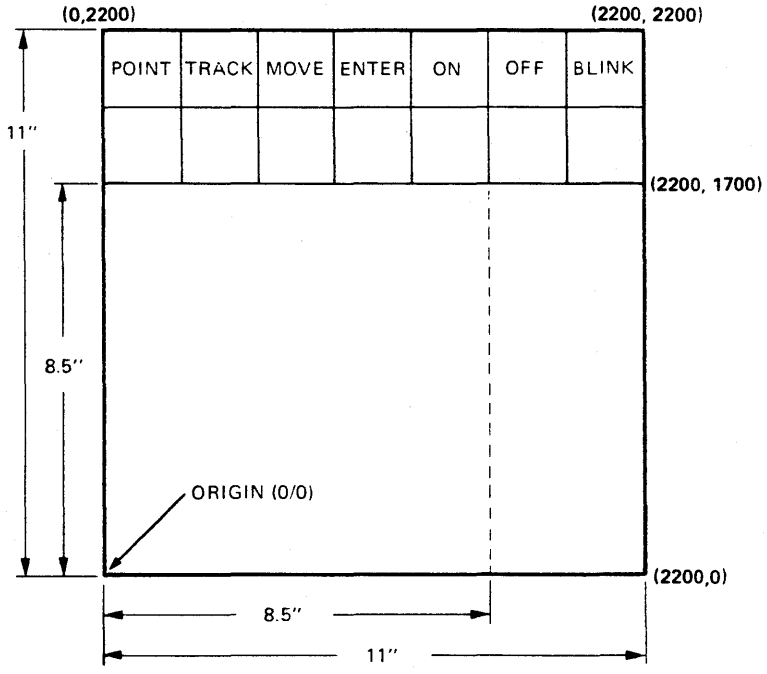
Figure C-3. GC-105 Light Pen Cursor Controller

Actuation occurs when the pen tip is pressed against the display surface. This arms the light-pen electronics to receive a detect pulse from the pen. It is also transmitted to the serial link PCB and its support software, which begins a single frame timeout. If light pen detection is not sensed within this time, the screen is momentarily flooded to the cursor color (for one frame time) ensure light pen detection. When the detect pulse is sensed by the light-pen electronics, the position of the CRT's electron beam (as derived from composite sync) is digitized and transmitted to the serial link PCB, which positions the cursor at that spot and interrupts the host computer to report cursor position. By momentarily flooding the screen, the light pen can detect dark regions of the screen and thus can serve as a display creation device; however, single pixel accuracy cannot be achieved except when the screen is highly magnified. For this reason, the light pen serves best as a menu selection device.



X0100-332 02B

Figure C-4. GC-108 Graphic Tablet Cursor Controller




B0100-313-01A

Figure C-5. GC-108 Graphic Tablet Layout

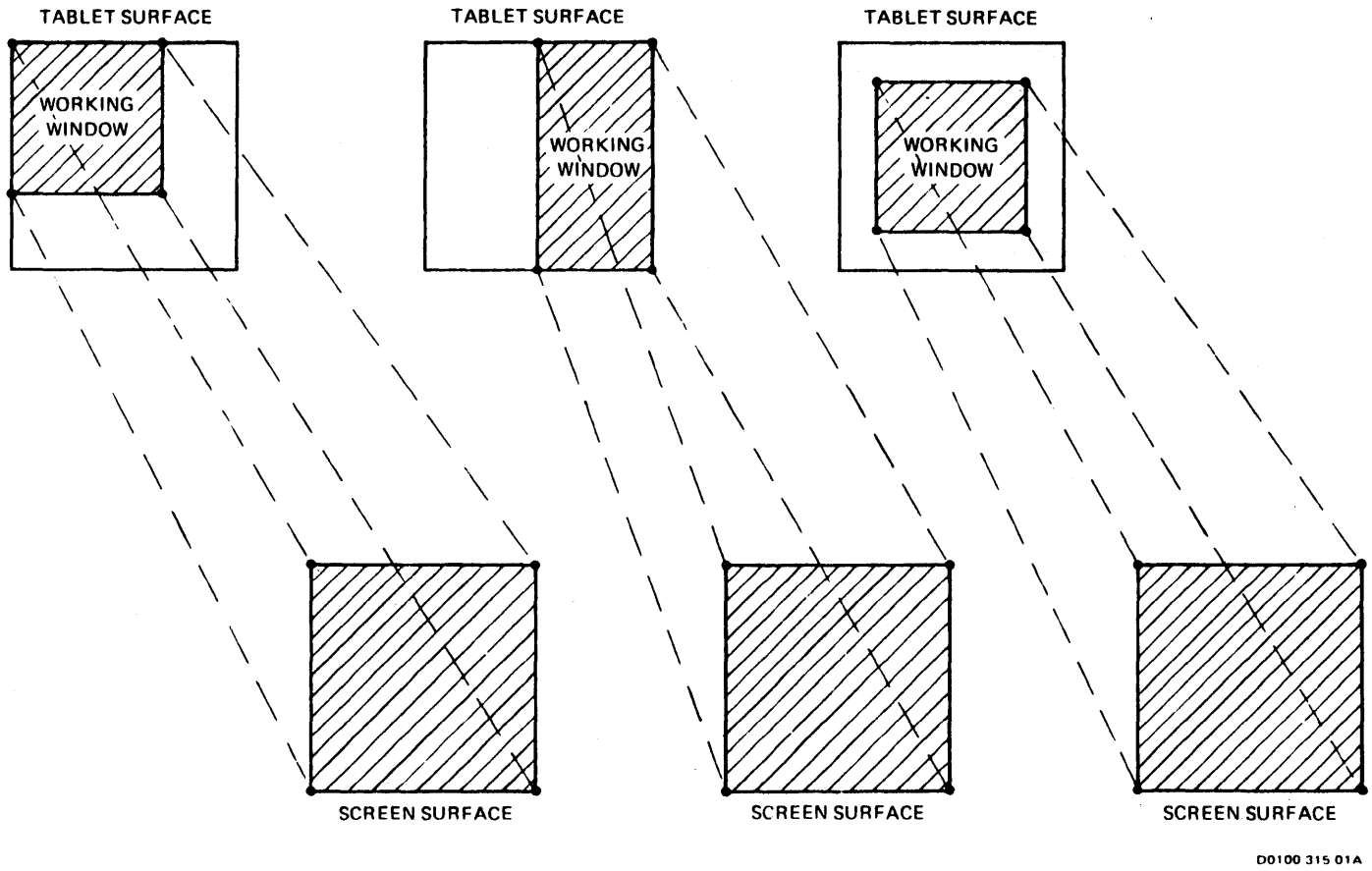
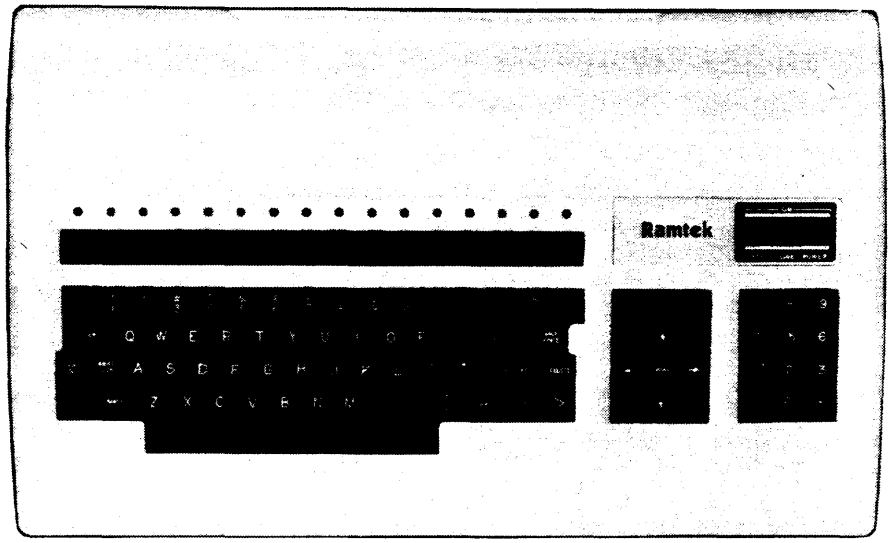



Figure C-7. Mapping Transformation in Non-Menu Mode




B0100-241-02A

Figure C-8. GK-120 General Purpose Keyboard

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT | LF | VT | FF | CR | SO | SI |
| 1 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ECS | FS | GS | RS | US |
| 2 | SP | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | | | | ~ | DEL |
| 8 | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ | $f_8$ | $f_9$ | $f_A$ | $f_B$ | $f_C$ | $f_D$ | $f_E$ | $f_F$ | $f_{10}$ |
| 9 | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ | $f_{16}$ | HOME $f_{17}$ | $f_{18}$ | $f_{19}$ | $f_{1A}$ | $f_{1B}$ | $f_{1C}$ | $f_{1D}$ | $f_{1E}$ | $f_{1F}$ | $f_{20}$ |
| A | $f_{21}$ | $f_{22}$ | $f_{23}$ | $f_{24}$ | $f_{25}$ | $f_{26}$ | $f_{27}$ | $f_{28}$ | | | | | | | | |
| B | | | | | | | | | | | | | | | | |
| C | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ | $f_8$ | $f_9$ | $f_A$ | $f_B$ | $f_C$ | $f_D$ | $f_E$ | $f_F$ | $f_{10}$ |
| D | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ | $f_{16}$ | HOME $f_{17}$ | $f_{18}$ | $f_{19}$ | $f_{1A}$ | $f_{1B}$ | $f_{1C}$ | $f_{1D}$ | $f_{1E}$ | $f_{1F}$ | $f_{20}$ |
| E | $f_{21}$ | $f_{22}$ | $f_{23}$ | $f_{24}$ | $f_{25}$ | $f_{26}$ | $f_{27}$ | $f_{28}$ | | | | | | | | |
| F | | | | | | | | | | | | | | | | |

ASCII (H)
ALPHANUMERIC SECTION
NUMERIC PAD
UNCONTROLLED
CURSOR PAD
FUNCTION PAD
SPECIAL FUNCTIONS
STATUS LIGHTS
CONTROLLED
NON-ASCII (H)

B0100-316-01A

Figure C-9. GK-120 TTY Mode Control

The shift and control keys determine the coded output of all keys. For example, consider the key labelled A:

| Shift | Control | Function | Hex. Code Output |
|---|---|---|---|
| - | - | a | 61 |
| x | - | A | 41 |
| - | x | SOH | 1 |
| x | x | SOH | 1 |

B0100-318-01A

The control key allows ASCII-defined control characters [00(H)-1F(H) and 7F(H)] as well as those codes that are not defined by ASCII [80(H)-FF(H)] to be output serially. All other codes are inhibited from appearing at the serial outputs when the control key does not lock out any codes. As an example of the control key locking out the above indicated codes, consider the key labeled with the numeral 3.

| Shift | Control | Function | Hex. Code Output |
|---|---|---|---|
| - | - | 3 | 33 |
| x | - | 3 | 23 |
| - | x | | Not Outputted |
| x | x | | Lockout Operation |

B0100-319-01A

Notice that both the shift and control keys determine the coded output of the keys in the alphanumeric section, but for all other sections (cursor pad, numeric/function pad and special functions), the shift key has no affect. Only the control key determines the coded output for the keys in these sections. There are two shift keys on the keyboard with one alternate action shift lock. Only one control key is provided.

Placing the keyboard in TTY mode causes the logic to suppress lower-case alpha characters. When in the TTY mode, the keyboard generates a 93-character subset of ASCII. Only the keys in the alphanumeric section are affected. The codes for all other sections of the keyboard (cursor pad, numeric/function pad and special function) remain unaffected by TTY mode. In TTY mode the codes are altered using the following rules:

- Codes 00(H) to 5F(H): retain as is.
- Codes 60(H) to 7E(H): subtract 20(H) to convert these codes to uppercase characters; that is, convert 60(H) through 7E(H) to 40(H) through 5E(H).

# Appendix D

# INSTRUCTION TIMING

# (Z80 SYSTEM PROCESSOR ONLY)

## D.1 INTRODUCTION

The following paragraphs describe the execution times of the RM-9460 display instruction set for the Z80 system processor. Timing information for the MC68000 system processor is not yet available.

In some cases, the instruction times are approximate due to the variable nature of the parameters used as input to the instruction. In these cases (which are noted in the instruction's execution time description), the equation describing the execution time becomes extremely complex and cumbersome, so that a reasonable approximation to the equation has been made.

## D.2 OTHER TIMING CONSIDERATIONS

Other factors must be considered to obtain an accurate overall time calculation for any particular display. These factors are host transfer delay times, overlay switching times, timing adjustment factors, and parameter operand processing times.

### D.2.1 Word Transfer Times

Timings have been made with no allowance for transfer delays from the host computer. Include transfer delay when calculating timings, or inaccurate values will result. The transfer delay is the elapsed time from when the RM-9460 sets ready (or acknowledge) to the next word written to the RM-9460. It is associated with every word transferred to the RM-9460, whether an opcode word, data word, or interface command. When executing from a display list, the transfer delay is 20 microseconds per word. In the following discussions, when the transfer delay time is used explicitly, it will be denoted TDT.

### D.2.2 Overlay Switching Times

The PROM overlay structure in the RM-9460 allows a virtually unlimited expansion capability. If the instruction to be executed is not in the fixed section or the currently resident overlay, switching adds to the execution time. To minimize this additional time, instructions should be grouped, if possible, so that several are executed in one overlay. The time required to switch to the new overlay is related to the order in which the overlays are placed on the Z80 PCB, with each sequential overlay taking an additional time increment. The trending option, when installed, occupies the first slot on the processor expansion PCB, and has a sequential overlay number which is one greater than the last sequential overlay number on the Z80 PCB. Table D-1 lists each instruction and its respective overlay.

Overlay switching time = 1120 usec + [ON * 140 usec]
where ON = sequential overlay number

Table D-1. Overlay Instruction Groupings (Continued)

| OPCODE | MNEMONIC | INSTRUCTION | OVERLAY |
|---|---|---|---|
| 30 | LCF | Load Cursor Font | STDFW |
| 31 | LCCIO | Load Cursor to COP/Index/Origin | STDFW |
| 32 | CWC | Center Window at Cursor | STDFW |
| 33 | DDCC | Define Device/Cursor Configuration | STDFW |
| 34 | DCVC | Define Cursor/Video Configuration | STDFW |
| 35 | WVU | Write Vector Unlinked | GRAF1 |
| 36 | WPP | Write Plot Point | GRAF1 |
| 37 | WPV | Write Plot Vector | GRAF1 |
| 38 | WPT | Write Point | GRAF1 |
| 39 | WRP | Write Random Pixel | GRAF1 |
| 3A | FILL | Fill | EG |
| 3B | ALDL | Allocate Display List | DL |
| 3C | DEDL | Deallocate Display List | DL |
| 3D | RESCON | Reset Context | STDFW |
| 3E | WBLK | Write Keyboard Block | STDFW |
| 3F | WBLKRP | Write Keyboard Block Reverse Packing | STDFW |
| 40 | RTC | Read Tablet Coordinates | STDFW |
| 41 | CDL | Call Display List | STDFW |
| 42 | RETDL | Return from Display List | STDFW |
| 43 | SDLR | Set Display List Register | STDFW |
| 44 | DDLR | Decrement Display List Register | STDFW |
| 45 | IDLR | Increment Display List Register | STDFW |
| 46 | SDLR | Subtract Display List Register | STDFW |
| 47 | ADLR | Add Display List Registers | STDFW |
| 48 | LDLR | Load Display List Register | STDFW |
| 49 | STDLR | Store Display List Register | STDFW |
| 4A | PLDLR | Parameter Load Display List Registers | STDFW |
| 4B | JDLR | Jump Conditional Upon Display List Register | STDFW |
| 4C | ALPF | Allocate Programmable Font | DL |
| 4D | DEPF | Deallocate Programmable Font | DL |
| 4E | ATTPF | Attach Programmable Font | DL |
| 4F | LMPF | Load Multiple Programmable Fonts | DL |
| 50 | PUSHM | Push Matrix | STDFW |
| 51 | POPM | Pop Matrix | STDFW |
| 52 | SETM | Set Matrix | STDFW |
| 53 | LM | Load Current Matrix | STDFW |
| 54 | SM | Store Current Matrix | STDFW |
| 55 | MM | Multiply Matrices | STDFW |
| 56 | MMI | Multiply Matrices Immediate | STDFW |
| 57 | IM | Initialize Matrix | STDFW |
| 58 | SCALE | Scale Matrix | STDFW |
| 59 | TRANS | Translate Matrix | STDFW |
| 5A | ROTAT | Rotate Matrix | STDFW |
| 5B | READM | Read Matrix | STDFW |
| 5C | ALCON | Allocate Context | STDFW |
| 5D | DECON | Deallocate Context | STDFW |
| 5E | SCON | Select Context | STDFW |

### D.2.3 Timing Adjustment Factors

Two additional factors must be taken into account when calculating timings: the pixel access time (PAT), which is the time required to write a pixel in refresh memory, and a frequency multiplier that is based on the resolution of the RM-9460. Both of these factors are detailed in table D-2. The PAT value is used in equations like write image where it affects the execution time directly. The frequency multiplier is used as an adjustment factor that will give an execution time based on the basic frequency of the RM-9460. After calculation of the execution time for all the instructions in a display sequence, multiplying by the frequency multiplier will give the true execution time for that display system.

Table D-2. Timing Factors

| SYSTEM TYPE in (Hz) | FRAME REFRESH RATE | ELEMENT RESOLUTION | LINE RESOLUTION | PIXEL ACCESS TIME ($\mu$sec) | FREQUENCY MULTIPLIER |
|---|---|---|---|---|---|
| RM-9460/8X | 30 | 1280 | 1024 | 1.49 | 0.990 |

80100-320-01A

- LOAD HARD REGISTER (LOAD)

  Execution Time = 650 usec

- READ SOFT REGISTER (READ)

  Execution Time = 650 usec

- LOAD AUXILIARY MEMORY (LAM)

  $$\text{Execution Time} = 1400 \text{ usec} + 16.67 \text{ msec} * \frac{[NW * DTR]}{5 \text{ msec}}$$

  where NW = Number data words
  DTR = Interface data transfer time

**NOTE**

To prevent video disruption, LUT loading or reading will occur only during horizontal or vertical retrace times. This allows a loading window of approximately 5 msec per 16.67 msec frame time.

- READ AUXILIARY MEMORY (RAM)

  $$\text{Execution Time} = 1400 \text{ usec} + 16.67 \text{ msec} * \frac{[NW * DTR]}{5 \text{ msec}}$$

- RESET (RSET)

  Execution Time = 1.7 sec

- INITIALIZE (INIT)

  Execution Time = 16.67 msec

- NO OP (INOP)

  Execution Time = 570 usec + OFT + PD + DFT + DD
  where OFT = 305 usec if OF1 or OF2
  OFT = 890 usec if OF1 and OF2
  PD = 80 usec per parameter word dumped
  DFT = 120 usec if DF = 1
  DD = 80 usec per data word dumped

¤ WRITE RANDOM COLORED TEXT (WRCT)

Execution Time = 895 usec + POPT + (525 usec * NCB) + (495 usec * NTS)
+ (NC * CT)

where NCB = Number of color blocks
NTS = Number of text strings
NC = Number of characters
CT = Maximum of 360 usec or CPT
CPT = [CHAR HEIGHT * CHAR WIDTH * (XSIZE + 1) * (YSIZE + 1)
* PAT] + [CHAR HEIGHT * (YSIZE + 1) * 20 usec]

¤ WRITE RASTER (WR)

Execution Time = 1000 usec + POPT + (615 usec * NL) + (105 usec * NB)
where NL = Number of lines of raster data
NB = Number of bytes of raster data

¤ WRITE VECTOR LINKED (WVL)

Execution Time = 1586 usec + POPT + NV * [(2 * TDT) + 35 usec +
(PAT * NP)]

where NV = Number of vectors
NP = Average number of pixels per vector
TDT = Transfer delay time
PAT = Pixel access time

¤ WRITE VECTOR UNLINKED (WVU)

Execution Time = 1490 usec + POPT + NV * [(4 * TDT) + 40 usec +
(PAT * NP)]

where NV = Number of vectors
NP = Average number of pixels per vector

¤ WRITE CONIC (WC)

Execution Time = 1135 usec + POPT + (NP * CPT)
where NP = Number of pixels in conic section
CPT = Conic pixel generation time: minimum = 280 usec
maximum = 700 usec
average = 300 usec

¤ WRITE CONIC 32 BIT PARAMETER (WC32)

Execution Time = 1860 usec + POPT + (NP * CPT)
where NP = Number of points in conic section
CPT = Conic pixel generation time: minimum = 515 usec
maximum = 1325 usec
average = 525 usec

¤ LOAD SUBCHANNEL ORIGINS (LOADSO)

Execution Time = 1800 usec

¤ WAIT FOR VERTICAL RETRACE (WAITVR)

Execution Time = 8.35 msec + [16.67 msec * Count] avg.
16.67 msec + [16.67 msec * Count] max.

¤ WAIT FOR VIDEO LINE (WAITL)

Execution Time = 8.35 msec + WT avg.
16.67 msec + WT max.

where $WT = \frac{\text{Line Number} * 16.67\ \text{sec}}{\text{Y-Resolution}}$

¤ ALLOCATE CONTEXT (ALCON)

Execution Time = 46.8 msec

¤ DEALLOCATE CONTEXT (DECON)

Execution Time = 2600 usec

¤ SELECT CONTEXT (SCON)

Execution Time = 2850 usec + PFT + DLT
where PFT = Programmable font time = 50 usec
DLT = Display list time = 100 usec

¤ RESET CONTEXT (RESCON)

Execution Time = 116.5 msec

¤ READ ALLOCATION STATE (READAS)

Execution Time = 6950 usec

¤ READ CONTEXT ASSOCIATIONS (RCON)

Execution Time = 4790 usec

¤ SENSE PERIPHERAL STATUS (SPS)

Execution Time = 630 usec

¤ SET DETECT PARAMETERS (SDP)

Execution Time = 2200 usec

¤ ENABLE DETECT (ED)

Execution Time = 625 usec

¤ DISABLE DETECT (DD)

Execution Time = 500 usec

¤ SUSPEND DETECT (SD)

Execution Time = 500 usec

¤ RESUME DETECT (RD)

Execution Time = 500 usec

¤ SET DETECT DATA (SDD)

Execution Time = 1000 usec

¤ SENSE DETECT STATUS (SDS)

Execution Time = 870 usec

¤ READBACK DETECT BUFFER (RDB)

Execution Time = 870 usec + [16 usec * NB]
where NB = Number of bytes read back

¤ ALLOCATE DISPLAY LIST (ALDL)

Execution Time = 1300 usec + [NB * 50 usec]
where NB = Number of 4K blocks allocated

¤ DEALLOCATE DISPLAY LIST (DEDL)

Execution Time = 2750 usec + [NB * 850 usec]
where NB = Number of 4K blocks deallocated

- INCREMENT DISPLAY LIST REGISTER (IDLR)

  Execution Time = 560 usec

- SUBTRACT DISPLAY LIST REGISTER (SUBDLR)

  Execution Time = 730 usec

- ADD DISPLAY LIST REGISTERS (ADLR)

  Execution Time = 730 usec

- LOAD DISPLAY LIST REGISTER (LDLR)

  Execution Time = 625 usec

- STORE DISPLAY LIST REGISTER (STDLR)

  Execution Time = 625 usec

- AND DISPLAY LIST REGISTERS (ANDDLR)

  Execution Time = 730 usec

- OR DISPLAY LIST REGISTERS (ORDLR)

  Execution Time = 730 usec

- XOR DISPLAY LIST REGISTERS (XORDLR)

  Execution Time = 730 usec

- LOAD MULTIPLE PROGRAMMABLE FONTS (LMPF)

  Execution Time = 1100 usec + [NC * CT]
  where NC = Number of characters
  CT = Character time:
  720 usec for 8 x 12 character
  2260 usec for 16 x 20 character

- PUSH MATRIX (PUSHM)

  Execution Time = 760 usec

- POP MATRIX (POPM)

  Execution Time = 800 usec

- SET MATRIX (SETM)

  Execution Time = 2700 usec

- LOAD CURRENT MATRIX (LM)

  Execution time = 775 usec

- STORE CURRENT MATRIX (SM)

  Execution Time = 735 usec

- MULTIPLY MATRICES (MM)

  Execution Time = 3930 usec

- INITIALIZE MATRIX (IM)

  Execution Time = 800 usec

- SCALE MATRIX (SCALE)

  Execution Time = 2450 usec

- TRANSLATE MATRIX (TRANS)

  Execution Time = 4260 usec

¤ WRITE ARC TYPE 3 (ARC3)

Execution Time = 1200 usec + NA * [11300 usec (NP * PT)]
where NA = Number of arcs
NP = Average number of pixels per arc
PT = Pixel generation time, 250 usec average

¤ SET LOCAL FUNCTION STATE (SLFS)

Execution Time = 565 usec

¤ SET LOCAL KEYBOARD FUNCTION (SLKF)

Execution Time = 3510 usec

¤ CLEAR LOCAL KEYBOARD FUNCTION (CLKF)

Execution Time = 3150 usec

¤ SET LOCAL CURSOR FUNCTION (SLCF)

Execution Time = 3460 usec

¤ CLEAR LOCAL CURSOR FUNCTION (CLCF)

Execution Time = 2910 usec

¤ SET KEYBOARD ECHO STATE (SKES)

Execution Time = 880 usec

¤ CLEAR KEYBOARD ECHO STATE (CKES)

Execution Time = 515 usec

¤ LOCAL FUNCTION ENQUE

Echo = 2750 usec
Cursor Controller = 2750 usec
Keyboard = 2875 usec min., 3325 usec max.

# Appendix E

## ERROR CODES

| ERROR CODE | OP CODE | DESCRIPTION |
|---|---|---|
| 0201 | READ | Attempt to execute READ from within a display list. |
| 0307 | LAM | Illegal byte count. |
| 0321 | LAM | No VLT corresponding to device number. All data discarded. |
| 0401 | RAM | Attempt to execute RAM from within a display list. |
| 0407 | RAM | Illegal byte count. |
| 0421 | RAM | No VLT corresponding to device number. |
| 0A07 | RI | Odd byte count in word mode. |
| 0B01 | RI | Attempt to execute RI from within a display list. |
| 0B07 | RI | Odd byte count in word mode. |
| 0B17 | RI | No MCPs selected to read from. |
| 0B18 | RI | Too many MCPs selected to read from. |
| 0B19 | RI | No groups selected to read from. |
| 0B1A | RI | Too many groups selected to read from. |
| 0E07 | WVL | Illegal value for DATA LENGTH WORD; not a multiple of four. |
| 0F07 | WC | Odd byte count. |
| 1007 | WPB | Odd byte count. |
| 1102 | SCRX | No (scratch) RAM available. |
| 1202 | SCRY | No (scratch) RAM available. |
| 130A | PUSHE | No room on environment stack. |
| 140B | POPE | No data on environment stack. |
| 1503 | LPF | Font RAM not allocated and attached to the current context; data discarded. |

| ERROR CODE | OP CODE | DESCRIPTION |
|---|---|---|
| 1F0D | LDLRP | DL does not exist; all data discarded. |
| 1F4D | LDLRP | Illegal COMMON display list address in extended mode. |
| 1F4E | LDLRP | Illegal normal display list address in extended mode. |
| 2001 | RDLRP | Attempt to execute RDLRP from within a display list. |
| 200C | RDLRP | Invalid DL NUMBER; no data readback. |
| 200D | RDLRP | DL does not exist; no data readback. |
| 204D | RDLRP | Illegal COMMON display list address in extended mode. |
| 204E | RDLRP | Illegal normal display list address in extended mode. |
| 2103 | LPF | Font RAM not allocated and attached to the current context; data discarded. |
| 2113 | LPF | Font to be loaded not 8 by 12; data discarded. |
| 2114 | LPF | Illegal ASCII CHARACTER CODE; data discarded. |
| 2307 | SETDC | DATA BYTE COUNT not a multiple of 4 or greater than 64. |
| 2308 | SETDC | One of the range pairs does not follow rule that [RANGE n DATA (HI)] $\geq$ [RANGE n DATA (LO)] |
| 2401 | READP | Attempt to execute READP from within a display list. |
| 2501 | RERR | Attempt to execute RERR from within a display list. |
| 2E01 | RCSP | Attempt to execute RCSP from within a display list. |
| 2F01 | RCSG | Attempt to execute RCSG from within a display list. |
| 3327 | DDCC | Illegal device/cursor association attempted. |
| 3428 | DCVC | Illegal cursor to video line association was attempted. |
| 3507 | WVU | Illegal value for DATA LENGTH WORD, not a multiple of eight. |
| 3607 | WPP | DATA LENGTH WORD does not reflect an even number of bytes. |
| 3707 | WPV | DATA LENGTH WORD does not reflect an even number of bytes. |
| 3807 | WPT | Illegal value for DATA LENGTH WORD; not a multiple of four. |
| 3907 | WPR | DATA LENGTH WORD not a multiple of six. |

| ERROR CODE | OP CODE | DESCRIPTION |
|---|---|---|
| 4610 | SUBDLR | DLR NUMBER out of range. |
| 4710 | ADLR | DLR NUMBER out of range. |
| 4810 | LDLR | DLR NUMBER out of range. |
| 484D | LDLR | Illegal COMMON display list address in extended mode. |
| 484E | LDLR | Illegal normal display list address in extended mode. |
| 4910 | STDLR | DLR NUMBER out of range. |
| 494D | STDLR | Illegal COMMON display list address in extended mode. |
| 494E | STDLR | Illegal normal display list address in extended mode. |
| 4A10 | PLDLR | DLR NUMBER out of range. |
| 4A11 | PLDLR | Illegal CONDITION TYPE was specified. |
| 4B0C | JDLR | Invalid DL NUMBER (display list execution terminated). |
| 4B0D | JDLR | DL NUMBER does not exist. |
| 4B10 | JDLR | DLR NUMBER out of range. |
| 4B20 | JDLR | Illegal CONDITION value. |
| 4B4D | JDLR | Illegal COMMON display list address in extended mode. |
| 4B4E | JDLR | Illegal normal display list address in extended mode. |
| 4C02 | ALPF | Attempt to allocate a font where not enough free RAM exists. |
| 4C0E | ALPF | Attempt to allocate a font that has already been allocated. |
| 4C12 | ALPF | PF NUMBER is out of range. |
| 4D03 | DEPF | Attempt to deallocate standard font. |
| 4D12 | DEPF | PF NUMBER is out of range. |
| 4E12 | ATTPF | The font defined by PF NUMBER is out of range. |
| 4E13 | ATTPF | The font defined by PF NUMBER has not been allocated. |
| 4F07 | LMPF | DATA BYTE COUNT not a valid multiple of either 14 or 42. |
| 4F12 | LMPF | PF NUMBER is out of range. |

| ERROR CODE | OP CODE | DESCRIPTION |
|---|---|---|
| 5E05 | SCON | Attempt to select a context that has not been allocated. |
| 5E16 | SCON | CONTEXT NUMBER out of range. |
| 5F01 | READAS | Attempt to execute READAS from within a display list. |
| 6001 | RCON | Attempt to execute RCON from within a display list. |
| 6701 | SDS | Attempt to execute SDS from within a display list. |
| 6801 | RDB | Attempt to execute RDB from within a display list. |
| 6902 | COPY | Not enough RAM available. |
| 6907 | COPY | Illegal byte count; not equal to 4. |
| 6917 | COPY | No MCPs selected to read from or write into. |
| 6918 | COPY | Too many MCPs selected to read from or write into. |
| 6919 | COPY | No group selected to read from or write into. |
| 691A | COPY | Too many groups selected to read from or write into. |
| 6A02 | COPYT | Not enough RAM available. |
| 6A07 | COPYT | Illegal byte count; not equal to 6. |
| 6A17 | COPYT | No MCPs selected to read from or write into. |
| 6A18 | COPYT | Too many MCPs selected to read from or write into. |
| 6A19 | COPYT | No group selected to read from or write into. |
| 6A1A | COPYT | Too many groups selected to read from or write into. |
| 6B07 | COMBI | Odd byte count in word mode. |
| 6C07 | COMBT | Odd byte count in word mode. |
| 6D07 | CIRC | Illegal value for DATA LENGTH WORD; not a multiple of six. |
| 6D35 | CIRC | Negative radius value specified. |
| 6E07 | ARC1 | Illegal value for DATA LENGTH WORD; not a multiple of 12. |
| 6E32 | ARC1 | Division by zero; no center can be found for the circle containing these points. |
| 6F07 | ARC2 | Illegal value for DATA LENGTH WORD; not a multiple of 10. |

| ERROR CODE | OP CODE | DESCRIPTION |
|---|---|---|
| 7616 | SKES | CONTEXT number out of range. |
| 7802 | ALTD | No RAM blocks available. |
| 780E | ALTD | TREND already exists. |
| 7823 | ALTD | Attempt to set up a trend with more than 12,000 trend points or more than 255 lines. |
| 7824 | ALTD | Illegal TREND number. |
| 7924 | DETD | Illegal TREND number. |
| 7A07 | ITD | BYTE COUNT will not allow an equal number of points to be loaded into each trend line.<br><br>BYTE COUNT defines a number of trend points greater than the number allocated for the trend. All data will be discarded. |
| 7A24 | ITD | Illegal TREND number. |
| 7B07 | LTDP | Illegal BYTE COUNT. All data will be discarded and an illegal instruction will be generated. |
| 7B24 | LTDP | Illegal TREND number. |
| 7C07 | UDTD | Illegal BYTE COUNT. The specified BYTE COUNT will not allow an equal amount of new data points to be added to each trend line. The data will be received and discarded, and an illegal instruction interrupt will be generated. |
| 7C24 | UDTD | Illegal TREND number. |
| 7D24 | ERSTD | Illegal TREND number. |
| 7E24 | DISTD | Illegal TREND number; TREND not allocated or does not exist. |
| 7E25 | DISTD | No trend database to display. |
| 7E26 | DISTD | START TIME is greater than the number of points per line. No change to the display will occur. |
| 7F07 | WPI | Illegal byte count; either odd byte count or not enough data to fill the last line. |
| 7F30 | WPI | Illegal packing parameter; either pixel length less than 2 or greater than 5, or pixel count * pixel length > 16. |
| 7F31 | WPI | COP not positioned at the FORMAT WINDOW starting edge of primary scan. |

| ERROR CODE | OP CODE | DESCRIPTION |
|---|---|---|
| 9246 | LCS | Illegal S2 record. |
| 9247 | LCS | Checksum fault. |
| 9307 | CCS | Illegal byte count; not a multiple of four. |
| 9344 | CCS | Too many CCS parameters (more than 16). |
| 9345 | CCS | Illegal start address. |
| 9407 | RSM | Illegal byte count (not equal to 2). |
| 9507 | COPYIR | Odd byte count. |
| 9517 | COPYIR | No MCPs selected to read from. |
| 9518 | COPYIR | Too many MCPs selected to read from. |
| 9519 | COPYIR | No group selected to read from. |
| 951A | COPYIR | Too many groups selected to read from. |
| 9607 | COPYIM | Odd byte count. |
| 9617 | COPYIM | No MCPs selected to read from. |
| 9618 | COPYIM | Too many MCPs selected to read from. |
| 9619 | COPYIM | No group selected to read from. |
| 961A | COPYIM | Too many groups selected to read from. |
| 964A | COPYIM | Bad magnification range (1-16 allowed). |
| 984B | FPOLY | Less than three vertices specified for polygon. |
| 994B | FCPOLY | Less than three vertices specified for polygon. |
| 9E07 | SPLN2 | Illegal DATA BYTE WORD. |
| 9E4C | SPLN2 | Illegal sub function code. SPLINE TYPE must be 0 or 1. |
| 9E50 | SPLN2 | GRAIN exceeds 4096. |
| 9F07 | SPLN3 | Illegal DATA BYTE WORD. |
| 9F4C | SPLN3 | Illegal sub function code. SPLINE TYPE must be 0 or 1. |
| 9F50 | SPLN3 | GRAIN exceeds 4096. |

| ERROR CODE | OP CODE | DESCRIPTION |
|---|---|---|
| A90D | ECDL | DL does not exist (display list execution is terminated). |
| A94D | ERDLRP | Illegal COMMON display list address in extended mode. |
| A94E | ERDLRP | Illegal normal display list address in extended mode. |
| A94F | ERDLRP | Illegal display list mode for instruction to operate. |
| AA10 | ELDLR | DLR NUMBER out of range. |
| AA4D | ERDLRP | Illegal COMMON display list address in extended mode. |
| AA4E | ERDLRP | Illegal normal display list address in extended mode. |
| AA4F | ERDLRP | Illegal display list mode for instruction to operate. |
| AB10 | ESTDLR | DLR NUMBER out of range. |
| AB4D | ESTDLR | Illegal COMMON display list address in extended mode. |
| AB4E | ESTDLR | Illegal normal display list address in extended mode. |
| AB4F | ESTDLR | Illegal display list mode for instruction to operate. |
| AC0D | EJDLR | DL NUMBER does not exist. |
| AC10 | EJDLR | DLR NUMBER out of range. |
| AC20 | EJDLR | Illegal CONDITION value. |
| AC4D | EJDLR | Illegal COMMON display list address in extended mode. |
| AC4E | EJDLR | Illegal normal display list address in extended mode. |
| AC4F | EJDLR | Illegal display list mode for instruction to operate. |
| AD03 | SLKFE | PROGRAMMABLE FONT does not exist. |
| AD05 | SLKFE | CONTEXT does not exist. |
| AD0D | SLKFE | DISPLAY LIST does not exist. |
| AD12 | SLKFE | PROGRAMMABLE FONT number out of range. |
| AD16 | SLKFE | CONTEXT number out of range. |
| AD33 | SLKFE | No room in local function table. |
| AD4D | SLKFE | Illegal COMMON display list address in extended mode. |

# Appendix F

# GLOSSARY

## F.1 INTRODUCTION

This glossary explains terms encountered in graphic display systems. Some entries are unique Ramtek hardware or software terms.

**Accumulator:** A 4-, 8-, 12-, 16-, or 32-bit register that temporarily stores arithmetic, logical, input, and output operations.

**Address:** An identification value (name, label, or number) assigned to a register, a unique memory location, or any other data source or destination.

**ALU:** (Arithmetic and logic unit) A functional element that performs addition, subtraction, multiplication, division, and exponentiation. This element also performs logic functions.

**Animation:** Sequenced movement of a displayed object, realized with techniques such as scrolling, coordinate transformation, or panning.

**Application program:** One or more tasks described in a computer language and executed by the user, when needed, for a specific purpose.

**Argument:** An independent variable; for example, when looking up a quantity in a table, the number, or any of the numbers, that identify the location of a desired value.

**Array:** Data structured such that each element is identified by one or more unique position indicators that reference a symbolic location in memory.

**ASCII:** (American Standard Code for Information Interchange) An industry standard code that has 128 octal values representing uppercase and lowercase characters, numerals, spacing, punctuation, special characters, and nonprinting machine or control commands.

**Attribute:** The quality that determines the appearance or the status of a graphic primitive, such as dashed or solid line, narrow or wide line, high or low intensity, or color.

**Baud rate:** A measure of data flow equal to the number of signal elements transferred per second. The size of the signal element varies from device to device. Baud rate equals bits per second (b/s) when each signal element carries one bit.

**Bit:** (Binary digit) The smallest discrete information unit stored in digital memory. Equal to either 1 or 0 in base 2 notation.

**CRT:** (Cathode ray tube) An electronic vacuum tube with a phosphorescent screen that stores and displays information. A rapidly modulated electron beam excites the phosphorescent screen, thus producing a display.

**Cursor:** A position indicator on a display device that indicates where the next data character will appear or where the next read operation should occur.

**Cursor interrupt:** See Peripheral interrupt.

**Cycle-stealing DMA:** An implementation of direct memory access (DMA) that exploits unused portions of the central processing unit (CPU) machine cycle to transfer data between devices. When correctly implemented, cycle-stealing DMA is totally transparent to the CPU and the user.

**DAC:** (Digital-to-analog converter) An electronic device that converts digitized information into an analog signal. In graphics, DACs output the digitized images to a CRT. For color imaging, three parallel DACs produce red, green, and blue signals. DACs commonly handle 2, 4, 6, or 8 bits of parallel data.

**Debugger:** A utility program that controls program execution. A debugger isolates execution errors by single stepping through instructions.

**Declutter:** Controlled visibility as a function of scale, rotation, or operator request. The instructions and display lists that generate an image are assigned display class numbers by the programmer. Selected types of data can then be shown or suppressed by enabling or disabling the instructions associated with a given range of display class numbers.

**Default:** The value assigned to a variable or action performed by a computer when either is left unspecified. For example, in the RM-9460 graphic displays, all memory planes are write-enabled by default. The WMSK instruction parameter lets the user change this default condition.

**Digitize:** A process that converts analog signals, such as audio or video, into a series of discrete digital levels. Conversion is accomplished with an analog-to-digital converter (ADC), and reconversion with a digital-to-analog converter (DAC).

**Display coordinate:** An address within global memory space whose display increment represents a single pixel in refresh memory. Refresh memory is registered against global memory space.

**Display list:** Buffered instructions and data stored by a display device for later execution. Display lists are called by number and loaded by the display processor into display device RAM for immediate or subsequent execution. Display lists are recursive and self-modifying, since they access a set of display list registers that may store or read data addresses or instructions. Common display lists are a special case because they always remain loaded and executable.

**Display list memory:** A dedicated portion of user RAM that stores display lists.

**I/O:** (Input/output) A generic term describing any device, program, or procedure for receiving and transmitting information.

**Interlaced scan:** A raster scan technique that activates every other row of phosphors during a vertical retrace, thus requiring two passes to refresh all pixels.

**Interrupt:** A special control signal that temporarily diverts a microprocessor to another job. Interrupts are a much more efficient than polling in a multi-tasking environment.

**Interrupt disable:** An instruction that sets or resets the interrupt-control flag. When this flag is false (disabled), the system ignores interrupt requests.

**Interrupt enable:** An instruction that sets or resets the interrupt-control flag. When this flag is true (enabled), the system acknowledges interrupt requests.

**Interrupt request:** An externally issued control signal that diverts the central processing unit from one task to another task.

**Interrupt service routine:** A program executed when an interrupt is issued. The service routine determines the interrupt source and what action to take.

**K-byte:** A unit (kilobyte) signifying 1024 bytes.

**LED:** (Light-emitting diode) A solid-state semiconductor device that lights when drawing current.

**Local space:** A 64K by 64K virtual picture coordinate space with the Cartesian origin (0,0) at the center. Pictures are written into local space before they are translated into global space.

**Loop:** An instruction sequence that repeats until some prespecified condition prevails.

**LSB:** (Least significant bit) The rightmost bit in a byte or word of digital data.

**Magnify:** A software-controlled destructive enlargement process that multiplies pixels in refresh memory before they are displayed. The program can magnify within an arbitrary window.

**Mask:** A binary data word that specifies which parts of another word the program operates on.

**Masking:** A technique that senses specific binary conditions and ignores others. The programmer specifies a bit mask for this purpose and then logically filters data through the bit mask. Bit mask positions that contain ones pass data. Bit mask positions that contain zeros block data.

**Operator:** The "what to do" portion of an operation; for example, "add" is the operator in the operation "add X." Also, the user who operates a computer.

**Overlaying:** A technique that discretely writes one image, or portion of an image, over another image on the screen. Normally, images are mixed when written on top of each other, but overlaying prevents this mixing. Character data is commonly overlaid on image data in process control displays.

**Palette:** A subset of colors available from a much larger set. When not specified, the palette usually defaults to a predefined set.

**Pan:** A highly interactive and nondestructive process that moves digitally represented objects stored in refresh memory by constantly updating the video origin. As a result, the object moves on the screen.

**PAT:** (Pixel access time) The time required to write a pixel in refresh memory.

**PCB:** (Printed circuit board) An epoxy and fiberglass substrate that supports analog or digital electronic components. Copper circuit traces substitute for electrical wires.

**Peripheral interrupt:** A control signal issued by a peripheral device that prompts the display to read cursor position or, in local functions, that prompts the display to execute a display list. In track mode, the peripheral device sends a stream of interrupts.

**Phosphor:** A rare earth element that glows when excited by an electron beam.

**Picture coordinate:** A point in global space as opposed to a point in refresh memory. Picture coordinates are issued as integer values that either represent or are transformed to refresh memory coordinates.

**PIO:** (Parallel I/O) An integrated circuit that does parallel input and output processing.

**Pixel:** (Picture element) The smallest addressable area on a display.

**Primary colors:** The three video colors, red, green, and blue. When combined, these three colors create all other colors.

**Primary scan:** See Primary update direction.

**Primary update direction:** The direction that text strings are initially written in until a format window boundary is encountered or the string ends. The primary update or scan direction is set by the SCAN parameter.

**Primitive:** A simple graphic element, like a line or polygon, having a specific appearance described by one or more attributes, including line width, orientation, and edge style. Primitive also denotes the algorithm that creates the graphic element.

**RGB:** (Red, green, and blue) Primary video colors. Secondary and tertiary colors are created by adding RGB colors together.

**ROM:** (Read-only memory) Partially or totally nonvolatile memory integrated circuit. A blank ROM integrated circuit exists as a mosaic of undifferentiated cells or gates until etched with a program, or data, or both.

**Rotation:** The angular displacement of an image through the display coordinate space of refresh memory without altering the relative distance between the points that constitute the image.

**Scaling:** The proportional displacement of picture and display coordinates by a fixed amount (the offset or scale factor). The spatial relationship of all points remains constant, so no distortion occurs.

**Scan orientation:** The direction in which an image is written out of refresh memory and onto the screen. Normal scan orientation is left-to-right, top-to-bottom.

**Screen:** The surface of a CRT that displays video information.

**Screen coordinate:** An X-Y intersection on the graphic display screen. This intersection may or may not relate to the point position in refresh memory.

**Scroll:** The process of displacing a screen image vertically, horizontally, or vertically and horizontally.

**Secondary scan:** See Secondary update direction.

**Secondary update direction:** The direction in which the graphic display system writes the remainder of long text strings after detecting a format window bounary. The secondary update or scan direction is set by the SCAN parameter.

**Spatial conversion:** The combined functions of coordinate transformation, registration, clipping, pan, and zoom. Picture coordinates are first converted to display coordinates via the coordinate transformation option that translates, rotates, and scales endpoint information as data is processed from the display list.

**String:** A contiguous set of characters,or numbers, or combinations thereof. Strings are often set off by a pair of delimiters that indicate origin and terminus.

**Subroutine:** A series of instructions that are part of a larger program or that represent an independent program. When a subroutine executes, the computer carries out a series of well-defined operations. Usually larger programs call subroutines to perform generic tasks. The main program hands control to the subroutine, which executes a task and then hands control back to the main program. Data is often passed between a subroutine and the main program as well.

**Wait:** A CPU state that suspends tasks that cannot execute immediately because of unfinished processing requirements.

**Window:** A rectangular area in display coordinate space that defines a display function boundary. Example display functions are clipping, scrolling, zooming, erasing, and entity detection.

**Z80 microprocessor:** The device name for an 8-bit microprocessor manufactured by Zilog, Inc.

**Zoom:** A highly interactive, nondestructive enlargement process that replicates pixels as they are projected from refresh memory onto the graphic display screen. Line thickness and character size, along with other details, increase as the picture is enlarged, but resolution decreases.

# Appendix G

# INDEX

## G.1 INTRODUCTION

This topical index facilitates finding information by referencing topics to corresponding manual pages. Entries and subentries are alphabetical. The chapter number reference is shown for the first page occurrence within a chapter. All subsequent page numbers within the chapter are separated by commas. Inclusive page numbers within a chapter are enclosed in parentheses.