



VECTRIX VX/PC COLOR CARD SET

USER'S GUIDE AND REFERENCE MANUAL

Revision B

Copyright 1984 by Vectrix Corp. All rights reserved.

VX/PC is a trademark of Vectrix Corporation.

The symbol IBM is used throughout this manual. IBM is a registered trade mark of International Business Machines Corporation. Radio Shack is a registered trademark of Tandy Corporation.

The illustrations on pages 1-7, 1-8, 1-9, 1-10, 1-12, 1-13, 1-15, 1-15, and 1-27 are reprinted from the IBM Options Installation manual by permission. All are copyright 1983 by International Business Machines Corporation.

All other illustrations in this manual were produced with the VX/PC card set, Vectrix Paint Pad software, and the Radio Shack CGP-220 Inkjet printer.

LIMITED WARRANTY

Vectrix Corporation warrants this product to be in good working order for a period of 90 days from the date of purchase from Vectrix or an authorized Vectrix dealer. Should this product fail to be in good working order at any time during this 90 day period, Vectrix will, at its option, repair or replace this product at no additional charge except as set forth below. Repair parts and replacement will be furnished on an exchange basis and will either be reconditioned or new. All replaced parts become the property of Vectrix. This limited warranty does not include service to repair damage to the product resulting from accident, disaster, misuse, abuse, or modification of the product by anyone other than Vectrix Corp.

Warranty service may be obtained by mailing or delivering the product in the original shipping cartons or the equivalent within the 90 day warranty period, and providing proof of the purchase date. When the product is delivered, you agree to insure the product and assume the risk of loss or damage in transit and to prepay shipping costs to the warranty service location. Contact Vectrix customer service for further information, by phone or at this address: Vectrix Corporation, Customer Services, 2606 Branchwood Drive, Greensboro, NC 27408.

All expressed and implied warranties for this product including the warranties of merchantability and fitness for a particular purpose are limited in duration to a period of ninety days from the date of purchase; no warranties, expressed or implied; will apply after this period. Some states do not allow limitations on how long an implied warranty lasts, so the above limitations may not apply to you.

If this product is not in good working order as warranted above, the sole remedy shall be repair or replacement as provided above. In no event will Vectrix Corporation be liable for any damages, including lost profits, lost savings, or other incidental or consequential damages arising out of the use of or the inability to use this product, even if Vectrix or an authorized Vectrix representative has been advised of such damages, or for any other claim by any other party. Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, so the above limitation may not apply to you. This warranty gives you specific legal rights and you may have other rights which vary from state to state.

PREFACE

The Card Set

The VX/PC Card Set is a two board graphics processor that is plug compatible with the IBM PC family of personal computers. It offers a resolution of 672 x 480 pixels, nine bitplanes, 2-D and 3-D operation, 4096 colors in the standard configuration, IBM color graphics card emulation, and high level graphics commands.

VX/PC's two on board processors, an INTEL 80188 and an NEC 7220, relieve the IBM of the burden of graphics calculations. The IBM PC sends a graphics command with its associated parameters to the VX/PC and then continues on to its next computing task without interruption. The VX/PC then independently performs all aspects of the graphics generation.

There are over 75 high level graphics commands stored in the VX/PC in firmware. Because the graphics library is resident on the VX/PC, little of the IBM PC's memory is used when executing graphics commands. There is no need to store graphics libraries in memory or to link procedures during program compilation.

The commands are readily available through character like commands that can be generated by any language's I/O statements. With this arrangement use of the command set is not limited to any particular language or operating system.

The screen display is laid out as a rectangle of 672 x 480 pixels. In 2-D mode, the pixels are numbered from 0 - 671 on the horizontal axis and 0 - 479 along the vertical axis. The coordinate 0,0 is in the lower left corner; the upper right is 671,479.

The VX/PC provides the capabilities of the IBM color graphics adapter card, and can be used in its place. The only functional differences are that the VX/PC does not draw border colors in IBM mode and the IBM color graphics adapter diagnostics cannot resolve the VX/PC's higher performance characteristics.

The VX/PC Disk

The disk supplied with the VX/PC card set contains the device driver needed to install VX/PC as a read/write device in the operating system. Other important files on the VX/PC disk include the print screen programs, hardware test programs, useful utilities, and a demonstration image. A complete list showing the name and function of each file on the disk is presented next.

FILE NAME	FUNCTION
MIDASDRV.COM	The VX/PC device driver
MIDASDRV.ASM	The source code for the driver
MPRINT.EXE	Screen Print program (Radio Shack)
MPRINT.ASM	Source code for MPRINT.EXE
QPRINT.EXE	Screen Print program (Canon, Quadjet)
APRINT.EXE	Screen Print program (ACT II)
CPRINT.EXE	ScreenPrint program(Canon PJ1080)
CODEFRAG.ASM	Source for GETCHR and PUTCHR
RESET.EXE	RESETs the VX/PC cards
SI.EXE	Switch to IBM mode program
SV.EXE	Switch to Vectrix mode program
TEST01.BAS	Test the VX/PC on installation
TEST01.DAT	Command file for TEST01
TEST02.BAS	Test the Light Pen
MODEKEYS	Redefines F9 as SI, F10 as SV
SPHERE	A demonstration command file
IODEMO.BAS	Manipulates color lookup table data
TODMA.OBJ	Linkable module for DMA transfers
TODMA.ASM	Source code for TODMA.OBJ
FROMDMA.OBJ	Linkable module for DMA transfers
FROMDMA.ASM	Source code for FROMDMA.OBJ
TEST0.EXE	Copies command files using DMA
TESTTO.ASM	Source code for TESTTO.EXE
TESTFROM.EXE	Captures images using DMA
TESTFROM.ASM	Source Code for TESTFROM.EXE

The Manual

This manual is designed to offer a practical introduction to full use of the VX/PC card set. The manual is divided into four main chapters: Installation, System Overview, Program Examples, and the Reference Guide.

Chapter 1, INSTALLATION, covers system requirements and options, installing the card set and options, and verifying proper installation. Checking individual hardware items is done by running supplied programs and comparing the results to examples in this manual.

Chapter 2, SYSTEM OVERVIEW, presents information on how the system works, general computer graphics theory, and background on the Vectrix implementation of commands. Program examples are provided to demonstrate various VX/PC features and commands.

Chapter 3, PROGRAMMING THE VX/PC CARD SET, covers general programming considerations, dealing with high level languages as well as assembly language.

Chapter 4, REFERENCE GUIDE, provides a structured way to find specific documentation of all VX/PC commands. Each command begins on a separate page describing command format and parameters, examples of its uses, relationship to other commands, and special characteristics.

APPENDIX A provides application notes on the color lookup table and bitplane management. APPENDIX B provides information on using DMA data transfers.

VX/PC USER'S GUIDE AND REFERENCE MANUAL

TABLE OF CONTENTS

Preface

Chapter 1 - INSTALLATION.....1-1

- Introduction
- VX/PC And Monitor Installation
- Installing The Software Driver
- Testing The VX/PC Card Set, Software, And Monitor
- Installing The Color Printer
- Installing The Light Pen
- Functional Specifications

Chapter 2 - SYSTEM OVERVIEW.....2-1

- How to Use This Chapter
- Vectrix and IBM Modes
- ASCII vs. Hex Mode
- Graphics Primitives
- Text Primitives
- Color Manipulations
- 3-D Transformations
- Pixel and RAM Commands
- Crosshair Commands
- Color Printing
- Exploring the System

Chapter 3 - PROGRAMMING THE VX/PC CARD SET.....3-1

- General Programming Overview
- High Level Language Programming
- Low Level Language Programming

Chapter 4 - REFERENCE GUIDE.....4-1

- How To Use This Chapter
- Command Arguments
- Command and Data Delimiters
- VX/PC Commands

- APPENDIX A - APPLICATION NOTES
- APPENDIX B - USING DIRECT MEMORY ACCESS
- GLOSSARY
- FURTHER READING
- INDEX



Chapter 1

HARDWARE INSTALLATION

Contents	Page
PREFACE	
1.1 INTRODUCTION	1-3
Packing Checklist	
System Requirements	
System Options	
1.2 VX/PC AND RGB MONITOR INSTALLATION	1-6
Introduction	
Summary Of VX/PC Installation	
Detailed Installation Procedures	
Connecting the RGB Monitor	
1.3 INSTALLING THE SOFTWARE DRIVER	1-19
1.4 TESTING THE VX/PC CARD SET, SOFTWARE, AND MONITOR ...	1-22
Testing The Installation	
Problem Determination	
SPHERE: A Demonstration Image Command File	
1.5 INSTALLING THE COLOR PRINTER	1-27
Installing the Printer	
Testing the Printer	
1.6 INSTALLING THE LIGHT PEN	1-29
Installing the Light Pen	
Testing the Light Pen	
1.7 FUNCTIONAL SPECIFICATIONS	1-30
RGB Monitor Specifications	
Light Pen Connector Pin Designations	

1.1 INTRODUCTION

This chapter leads you through all phases of the installation of your VX/PC card set. This process is similar to installing any card in your IBM Personal Computer. The installation steps are shown with both pictures and text. If questions arise concerning installation of any equipment, and are not covered in this chapter, please call Vectrix Customer Service.

The actual installation process is very similar to installing any peripheral card in the IBM PC/XT chassis. The one difference is that the VX/PC is a two card set, and the two cards should be installed in adjacent slots.

Please read all relevant sections in this chapter before attempting to install your Card Set. This will save both you and Vectrix time and trouble. Most questions are answered here; please reserve telephone contact for those that are not.

1.1.1 Packing Checklists

Before starting the installation procedure, use the checklist below to determine if all necessary items have been received.

The VX/PC Graphics System is shipped from Vectrix in several boxes under separate cover. There is a possibility the boxes will arrive at their destination at different times.

The boxes and their respective contents are listed next.

The VX/PC Card Set

_____ VX/PC GRAPHICS PROCESSOR (2 Cards)
_____ 2 PLASTIC CARD SUPPORT BRACKETS
_____ WARRANTY CARD

The VXM RGB Monitor

_____ VX/PC TO RGB MONITOR CABLE
_____ RGB MONITOR MANUAL
_____ WARRANTY CARD

This Manual and VX/PC Diskette

_____ VX/PC DISKETTE
_____ VX/PC USER'S GUIDE AND REFERENCE MANUAL

Remove all parts from their boxes and check for shipping damage. If any damage to the equipment is found, file a claim with the shipping agent. Vectrix will work with the shipper to quickly replace or repair the equipment.

If an item is missing, check the Vectrix packing slip for reference to a backordered item. Notify the shipping agent and call Vectrix Corporation, (919) 288-0520, and ask for Customer Service. Please have available the Work Order number, located in the upper right corner of the packing slip.

NOTE: All boxes, the foam packing material, and antistatic bags should be safely stored for later use in shipping your equipment.

This is a good time to fill out the Warranty/Registration Cards. This card is important to both you and Vectrix. Without the registration cards, Vectrix cannot honor your warranty.

1.1.2 System Requirements

The IBM PC XT computer requires these components to run the Vectrix VX/PC Graphics system:

VX/PC Card Set
High Quality Analog RGB Monitor
Card Set to Monitor cable

In addition, the IBM PC requires one of the following:

The IBM PC expansion chassis, or
IBM PC power supply upgrade to 130 watts.

The expansion chassis or power supply upgrade must be properly installed and in known working condition prior to the installation.

1.1.3 System Options

Monochrome Monitor and Adapter

The VX/PC card set provides a high quality graphics image and IBM Color Card emulation on a single RGB monitor. Both modes can use the same Vectrix monitor and do not require additional display cards or monitors. We will call a system with one monitor an RGB system.

The VX/PC also functions properly with an IBM monochrome monitor and display card installed in the system. We will call this an RGB and Monochrome system. Using two monitors, the monochrome monitor functions as the system console while the RGB monitor functions as the graphics display.

The only configuration not supported is both the IBM or equivalent color card and the VX/PC card set. This is because the VX/PC contains an IBM compatible color card. The IBM PC/XT does not allow two IBM color graphics cards to be in the system.

Color Printer

The five color printers currently supported by Vectrix for the VX/PC are these:

- Canon A1210 Color Inkjet Printer
- Radio Shack CGP-220 Color Inkjet Printer
- Quadram Quadjet Printer
- ACT II Color Inkjet Printer
- Canon PJ1080 Color Inkjet Printer

These printers require an IBM or equivalent parallel printer interface cards and an IBM to Centronics cable.

Light Pen

A light pen is available from Vectrix that is compatible with both the Vectrix light pen commands and the IBM commands. It operates when the VX/PC is in either Vectrix mode or in IBM mode.

1.2 VX/PC AND RGB MONITOR INSTALLATION

1.2.1 Introduction

Installing the VX/PC card set requires three basic steps:

- Step 1) Install the VX/PC card set and the RGB monitor. Since the two cards are physically connected, they are installed at the same time.
- Step 2) Install the software driver. The driver allows PC DOS to recognize and communicate with the VX/PC card set. Testing this software driver also tests the hardware installation.
- Step 3) Connect and test optional devices, a printer and light pen.

1.2.2 Summary Of VX/PC Installation

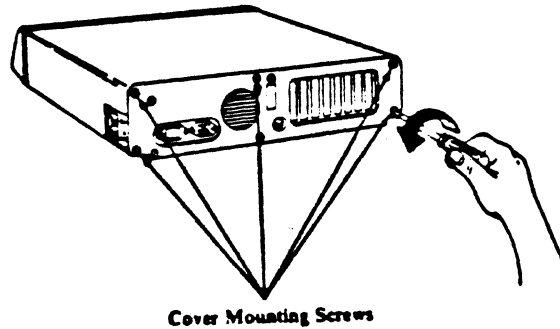
The installation process is summarized next, then detailed in the following pages.

1. Turn the computer off, disconnect any peripheral equipment, and remove the system unit cover.
2. Select two adjacent expansion slots to insert the two VX/PC cards. Remove the expansion slot covers.
3. Remove the VX/PC cards from their shipping container. Locate the card labeled "Memory" and the card labeled "Processor".
4. Set the DIP Switch Bank #1 on the IBM system board.
5. Insert the VX/PC card set into the system unit and replace the cover.
6. Verify RGB monitor switches, connect the RGB video cable from the VX/PC card to the RGB monitor, and reconnect any peripheral equipment removed in Step 1.
7. Install the software driver.
8. Test the installation of the VX/PC cards and the software driver.

1.2.3 Detailed Installation Procedures

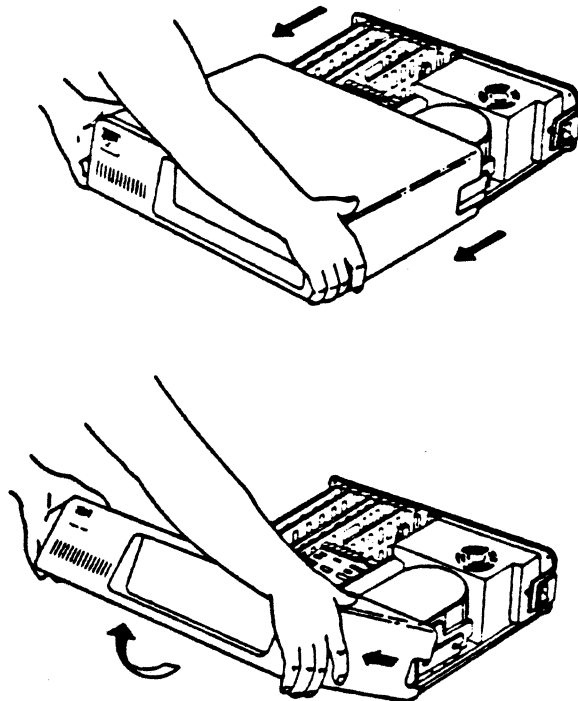
- 1a) Turn off the power and disconnect all cables.
- b) Remove peripheral equipment from the immediate area.
- c) Turn system so the back of the unit is facing forward.
- d) Remove the five retaining screws on the back of the system unit. (Illustration 1)

ILLUSTRATION 1 Removing The Cover Mounting Screws



- e) Carefully slide the cover toward the front of the unit. When the cover is almost off, it can be tilted upward (Illustration 2) to slide off completely.

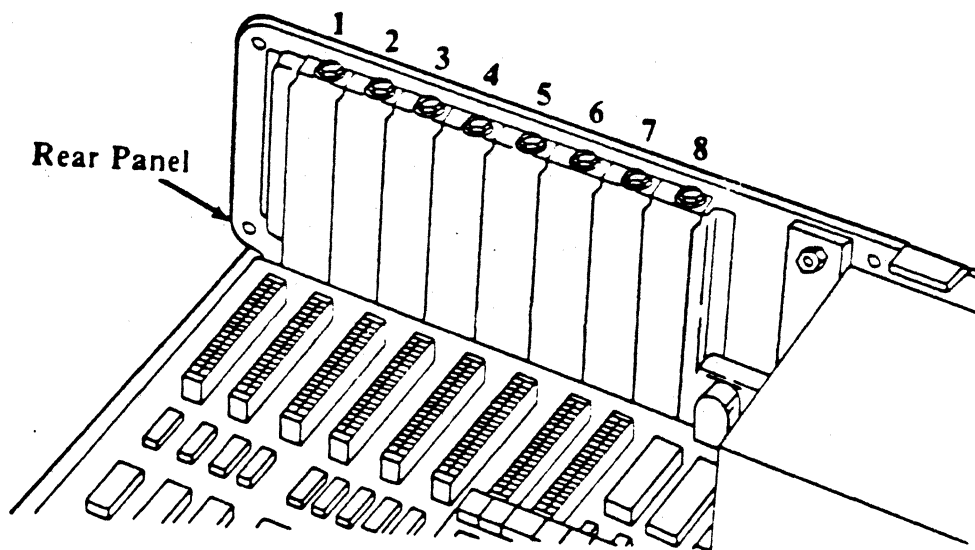
ILLUSTRATION 2 Removing The Cover



- 2a) Locate the eight expansion card edge connectors inside the system unit chassis, next to the power supply. Connectors #1 and #2, farthest from the power supply, will hold the VX/PC cards. If either slot contains other peripheral cards, the cards must be moved.

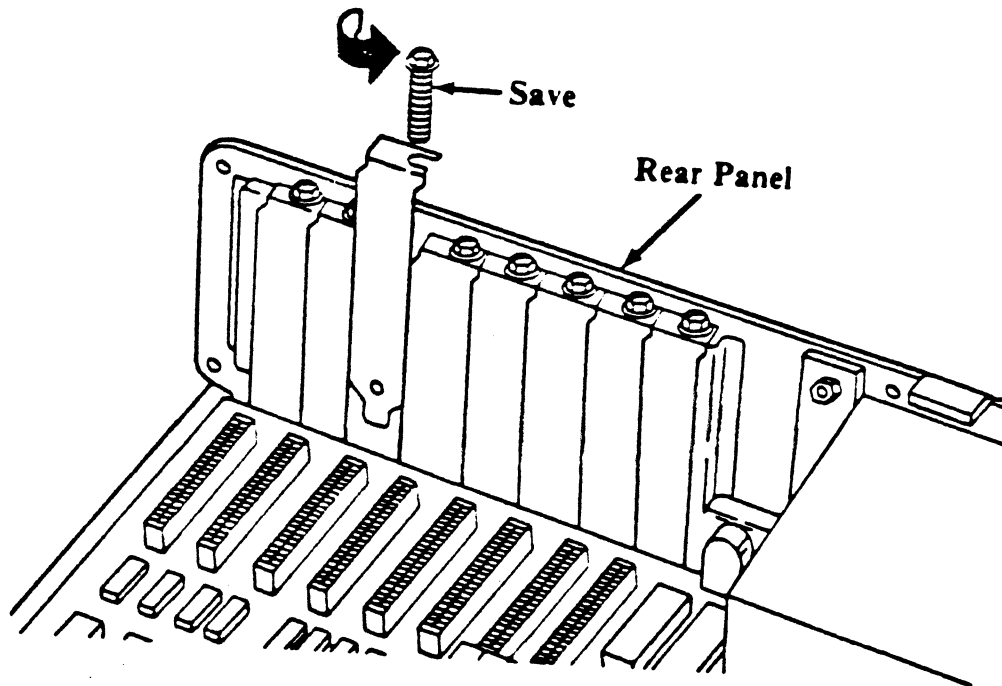
Note: The IBM PC has five card edge connectors. The two leftmost connectors are used for the VX/PC cards.

ILLUSTRATION 3
The IBM XT Expansion Connectors



- 2b) Remove the screws retaining the expansion covers and set aside. (Illustration 4) Save the screws for use with the cards. The slot covers are not used, but should be kept to cover the slots if the cards are removed.

ILLUSTRATION 4
Removing The Expansion Slot Covers



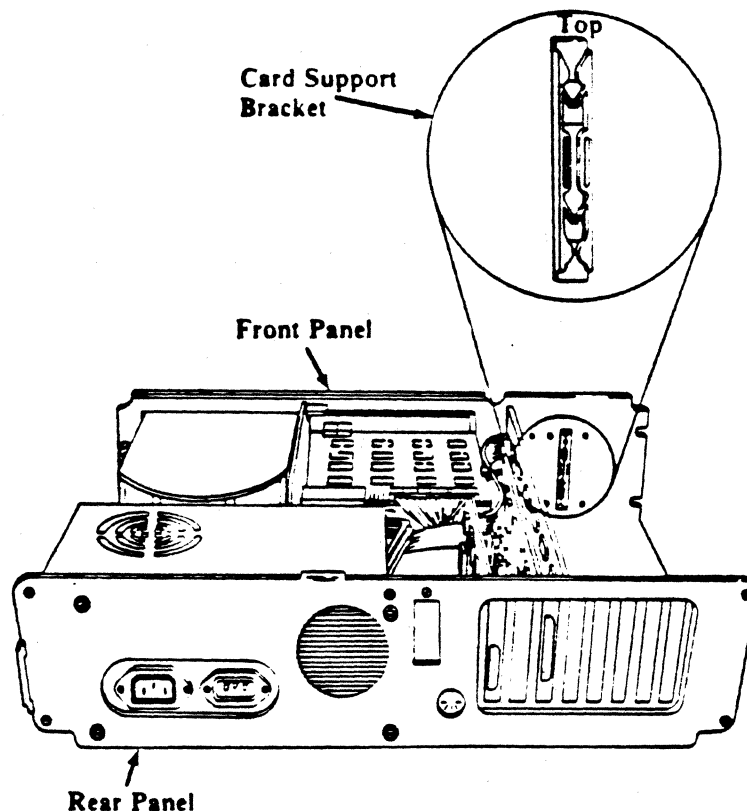
- 2c) Remove any existing color graphics display adapter card from the system. If the system now uses a color monitor, it will have a color graphics display adapter card. It must be removed.

The VX/PC has the equivalent of a built in IBM color graphics adapter card, so any other color graphics card in the system must be removed. If an IBM or compatible graphics card is left in the system, neither it nor the VX/PC card will work properly, and damage to either or both cards may result.

A monochrome display adapter will cause no problems.

- 2d) Install the plastic card support brackets in the front panel holes corresponding to the installation slots chosen. Since the machine is turned around, the front panel is the side farthest from you, containing the disk drives.

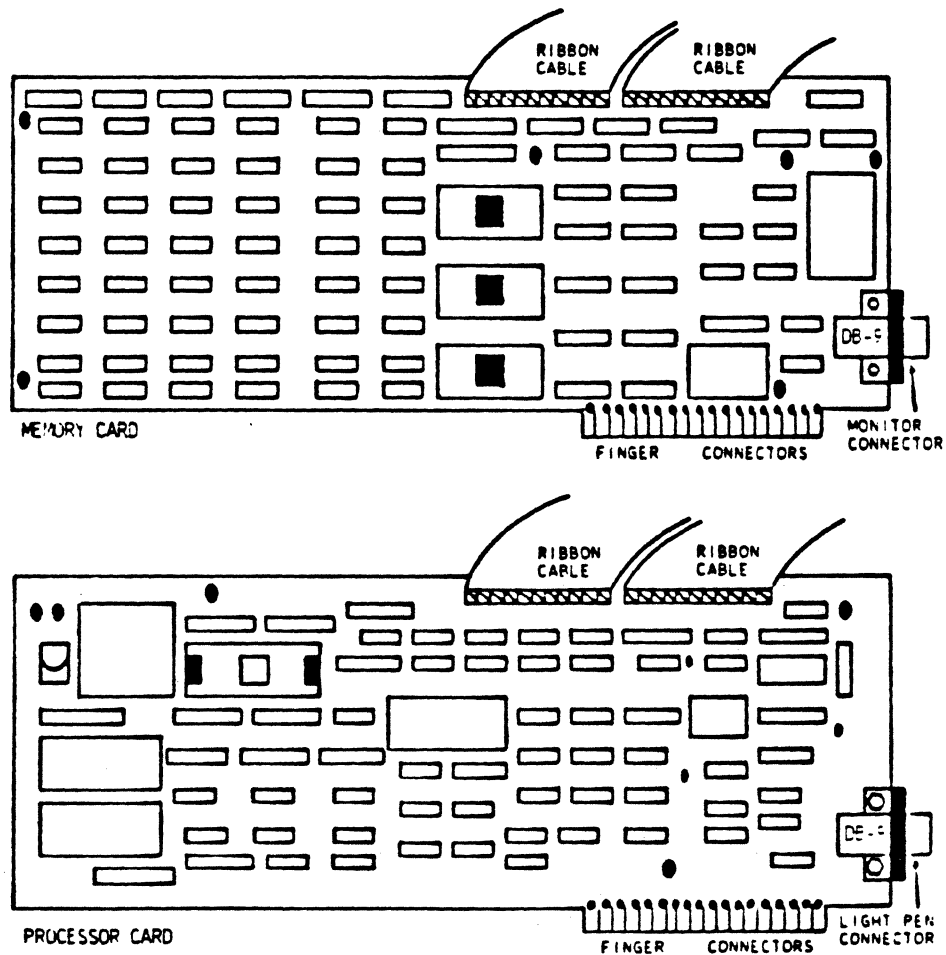
Illustration 1.5
Installing the Card Support Brackets



- 3) Remove the VX/PC cards from their box and study the the two cards. Compare the two boards with Illustration 1.6. The processor board has the legend "PROCESSOR BOARD" written at the bottom. The memory board has the legend "MEMORY BOARD" written at bottom center. The cards are connected by two ribbon cables.

Note that each board has a DB-9 connector at the back of the board, above and to the right of the card edge finger connectors. The processor board has a female DB-9 connector for an optional light pen. The memory card has a male DB-9 connector used to connect to an RGB monitor.

Illustration 1.6
VX/PC Card Set Diagrams



4) Set the DIP switch on the IBM system board.

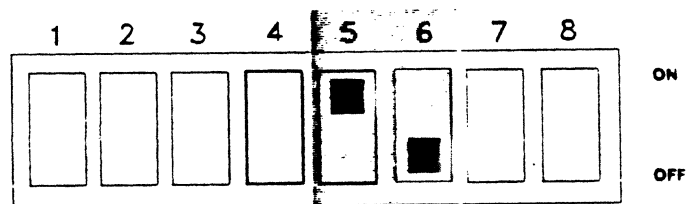
Switch Bank #1 on the IBM system board must be set to reflect the number and type of monitors installed. Illustration 1.7 shows the two possible settings for switches five and six on the system board DIP switch.

This small switch is located on the system board at the bottom of the cabinet, to the right of the power supply and directly behind the floppy disk drives. It is usually hidden by the disk drive cabling. The easiest way to move the individual switches is with a ball point pen or small screwdriver.

Note: The settings for switch #5 and #6 are the same as described in the IBM Installation Guide.

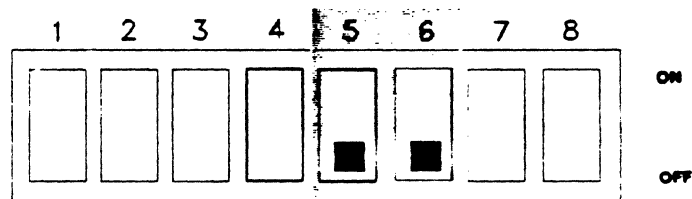
Using an RGB requires that switch #5 be set ON and switch #6 be set OFF.

Illustration 1.7a
Switch Settings For An RGB System



When a monochrome monitor and adapter card are used, the switches are set differently. Switches #5 and #6 must both be set to OFF. This allows use of both monitors, though only one can be active as the system console at a time.

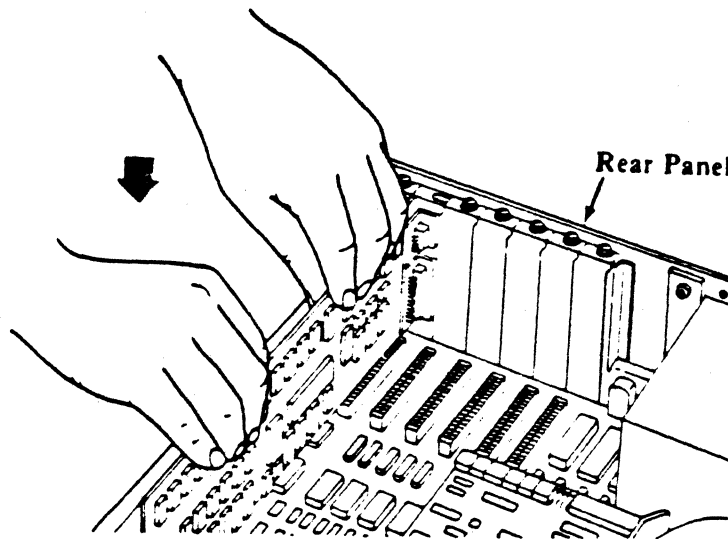
Illustration 1.7b
Switch Settings For An RGB and Monochrome System



- 5a) Insert the VX/PC cards in the two leftmost expansion slots on the IBM system board.

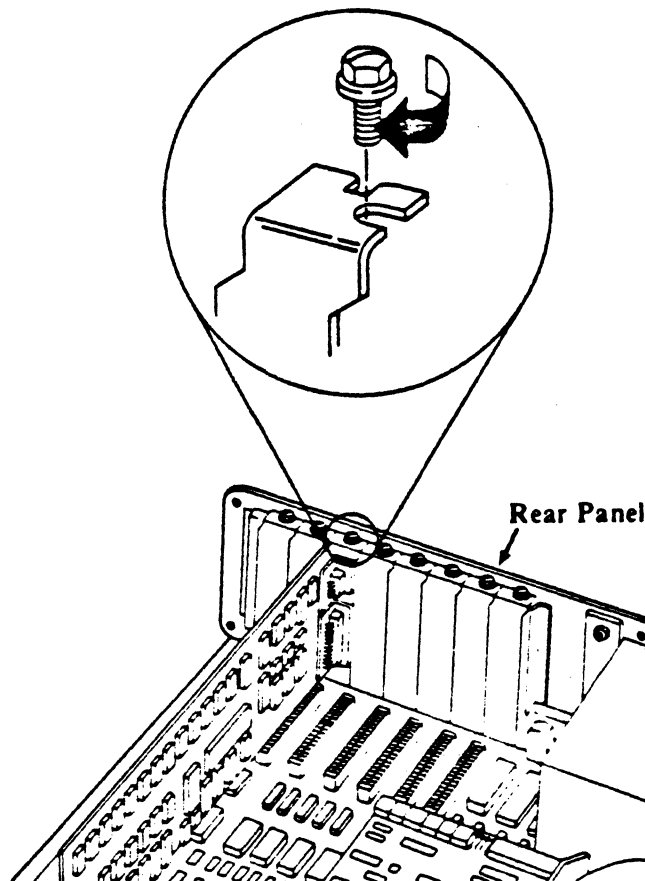
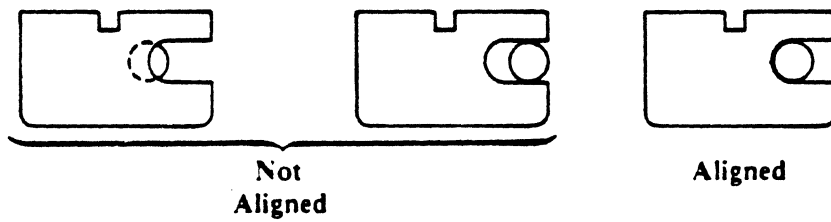
Holding the VX/PC cards (finger connectors facing down) over the card edge connector on the computer, align the cards with the card support brackets on the front and the metal brackets on the back. Center the cards over the card edge connector and push them into the connectors. Now press the cards firmly into the connectors. The gold finger edge connectors should be barely showing at the top of the connectors when the boards are pressed in correctly.

Illustration 1.8
Inserting the VX/PC Card Set



- 5b) Align the holes in the retaining bracket and the holes in the cards' brackets. Insert the screws and tighten.

Illustration 1.9
Aligning the Retaining Brackets

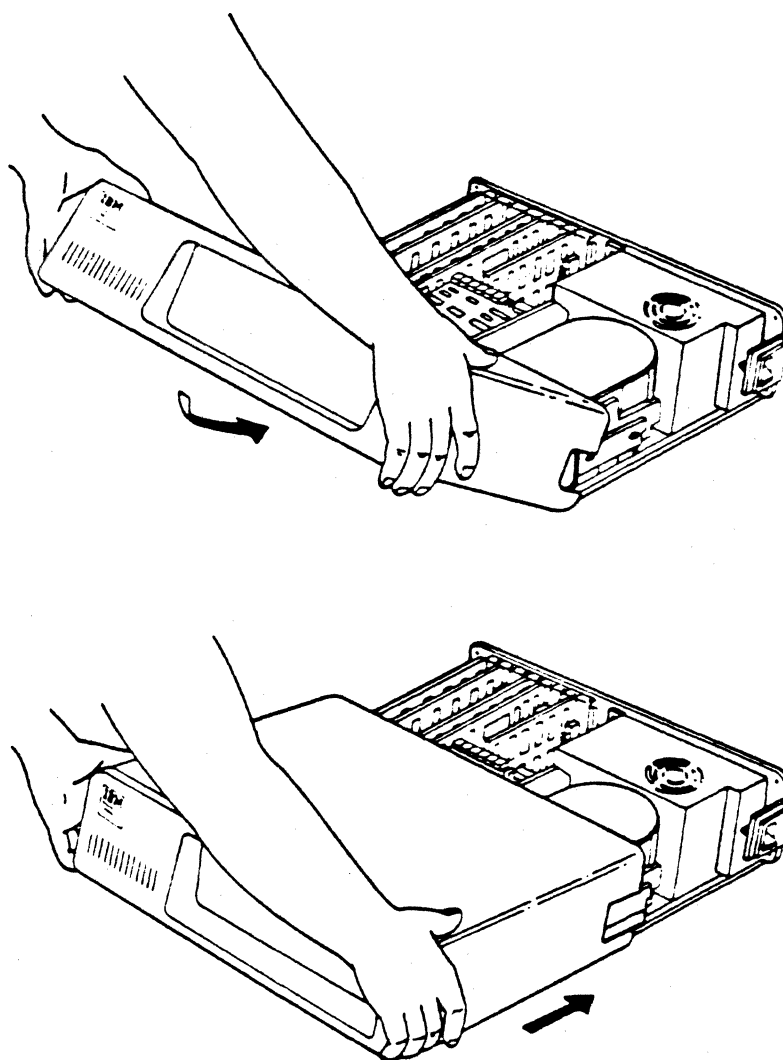


- 6a) Connect the DB-15 RGB video cable to the female DB-15 connector on the VX/PC card.

This cable has a male DB-15 connector on one end, and five connectors on the other end, carrying RGB video signals, and ending in BNC type connectors.

- 6b) Replace the system unit's cover by reversing the removal procedure.

Illustration 1.10
Replacing the System Unit Cover



1.2.4 Connecting The RGB Monitor

***** WARNING *****

AN IBM OR EQUIVALENT COLOR MONITOR MUST NOT BE USED.
DAMAGE TO THE MONITOR AND THE VX/PC CARD SET WILL RESULT.

AN IBM MONOCHROME MONITOR, IF USED, MUST BE CONNECTED
DIRECTLY TO THE IBM MONOCHROME ADAPTER CARD - IT CANNOT BE
CONNECTED TO THE VX/PC CARDS.

7a) Setting the VX/PC Monitor Switches.

Although all Vectrix monitor switches were properly set at the factory, the switch positions may have changed during shipping. On the right front of the monitor, two switches should be verified as follows:

UNDERSCAN / OVERSCAN should be set to UNDERSCAN.
VIDEO OFF / ON must be set to ON.

On the back of the monitor, near the top, are three switches that should be checked. The first switch (Termination) should be set to 75 Ohms, the up position. The switches under the legends VERT SYNC and HORIZ/COMB SYNC are set as follows:

VERT SYNC should be set in the down position.
HORZ/COMB SYNC should be set in the down position.

7b) Connecting the VX/PC Card Set and Monitor

The Vectrix RGB monitor cable has a male DB-15 connector on one end and five BNC connectors on the other.

The five ends of the cable are connected to the RGB monitor. These connectors are labeled RED, GREEN, BLUE, VSYNC AND HSYNC, corresponding to the labels on the monitor back. Connection is made by positioning the BNC connector over the monitor connector, then pushing and twisting the connector to lock it in place.

8) Reconnect any peripheral equipment removed in step 1a.

1.3 INSTALLING THE SOFTWARE DRIVER

After the VX/PC hardware is installed, software designed to make the IBM PC/XT recognize the VX/PC is installed. This software is called a driver, and is installed using standard driver installation procedures. Once the driver is installed, it will always be available when the system is turned on. RGB and Monochrome system installation follows the same procedures as an RGB system installation.

The driver program, MIDASDRV.COM, must be installed on the boot disk, in the root directory, in a file named CONFIG.SYS. For a PC, the boot disk will be DRIVE A. On an XT, the driver will generally be installed on the hard disk which is usually DRIVE C.

The steps necessary to install the device driver and test the system are listed next. Where (enter) appears, press the ENTER key. All parenthetical text is comments and should not be entered.

- 1) Turn the computer on. Insert the disk that came with this manual in Drive A.
- 2) Choose a disk to be the VX/PC system disk.

When using a XT with a hard disk, the system "disk" is usually the root directory on drive C, the hard disk.

In either case, these programs and files must be on the system disk, in the root directory:

COMMAND.COM	(copy from the PC DOS disk)
EDLIN.COM	(copy from the PC DOS disk)
ANSI.SYS	(copy from the PC DOS disk)

- 3) Copy all the files from the supplied VX/PC disk to the system disk. The command

```
COPY d1: *.* d2:
```

(where d1 is the disk - A, B, or C - to copy FROM, and d2 is the disk to copy TO) will do this.

This has the added benefit of creating a backup disk for the files on the VX/PC disk.

4) The CONFIG.SYS file can now be set to load and run MIDASDRV.COM each time the system is booted. The method detailed in the following paragraph will work whether or not there is an existing CONFIG.SYS file. This procedure sets up a file to be run whenever the system is turned on or booted.

4a) Type: EDLIN \CONFIG.SYS

The message "NEW FILE" will appear if no CONFIG.SYS file exists. If there is a CONFIG.SYS file, the message End Of Input File will appear.

In either case, at the asterisk type:

1I

This means Insert a line before line #1.

4b) At the "1:*" prompt, type:

DEVICE=MIDASDRV.COM

exactly as it appears here, then press the ENTER key. This line installs the device driver when this file is later executed.

4c) At the "2:*" prompt, type:

DEVICE=ANSI.SYS

exactly as it appears here, then press the ENTER key. This sequence will later allow redefinition of function #9 to issue the "SI" command, and function key #10 to issue the "SV" command.

4d) At the "3:*" prompt, type CONTROL C by holding down the "Ctrl" key and pressing the "C" key.

4e) At the asterisk, type:

E

This ends the editing session and saves the file CONFIG.SYS to disk.

The next step is testing both the hardware and software driver installation. This is explained in Section 1.4, "TESTING THE VX/PC CARD SET, SOFTWARE, AND MONITOR".

1.4 TESTING THE VX/PC CARD SET, SOFTWARE, AND MONITOR

This section describes using the test command file, TEST01.DAT to test both the RGB hardware and software driver installation.

1.4.1 Testing The Installation

The first step is making sure the MIDASDRV.COM driver is installed. Turn the system off, then on. The system must be rebooted to install the driver. This can also be done by keying the Ctrl-Alt-Del sequence. Now turn on the monitor.

At the DOS prompt, type:

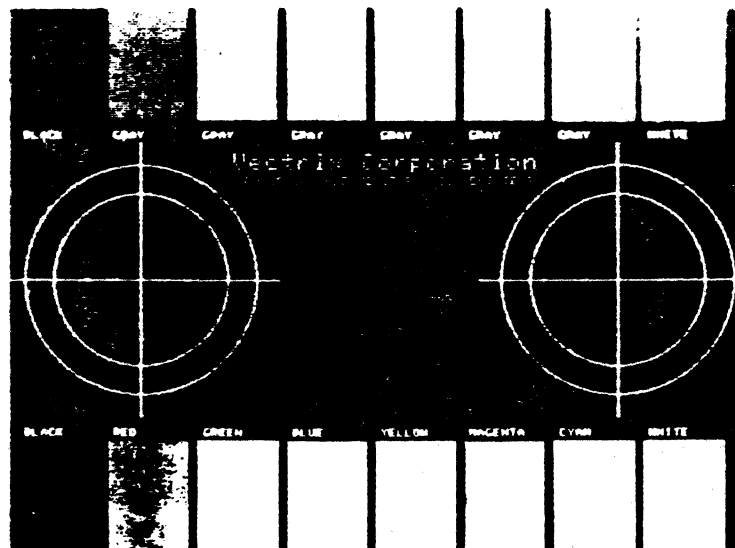
```
TYPE MODEKEYS (Sets Function key 9 to be "SI",
               and function key 10 to be "SV".)
```

When the DOS prompt returns, type:

```
COPY TEST01.DAT VECTRIX (Send command file to
                        the VX/PC)
```

The screen display should look like Illustration 1.11, although it will be in color.

Illustration 1.11
TEST01.DAT - The Color Bar and Monitor Alignment Test
Command File Output



Press F9 to switch to IBM mode when done. The image can be displayed again by pressing, at the DOS prompt, the F10 key, which issues a "SV" command, switching to Vectrix mode.

If the image does not display properly or if the circles are not circular, skip to "Problem Determination", Section 1.4.2.

Otherwise, the cards and monitor are correctly installed.

NOTE: This command series does not run the TEST01.BAS program. It COPYs the command file created earlier by the program. This command file contains the output of the program TEST01.BAS and is supplied on the VX/PC disk. More information on command files is presented in VX/PC User's Guide And Reference Manual, Chapter 3, Section 3.2.4, "Loading and Saving Images Faster".

1.4.2 Problem Determination

If No Image Appears

The first step in determining problems is checking all cable connections. If all cables are connected, and the system does not work, the card set may not be operative. Even though the card sets are tested for a 163 hour burn in period before shipment, it is possible that some of the chips may have worked loose.

The next step is removing the board set. The cards should be laid flat on a soft surface, component side up. Gently press each chip that is socketed. This reseats any loose chips. Insert the card set and connect the monitor. Repeat testing using the TEST01.DAT command file.

An Incorrect Image Appears

If the image appears, but the color names and the color of the rectangles do not agree, the cables from the VX/PC card to the monitor are probably misplaced. The individual cables are lettered, and must agree with the legends on the monitor. Change any that are incorrect and run the test sequence again.

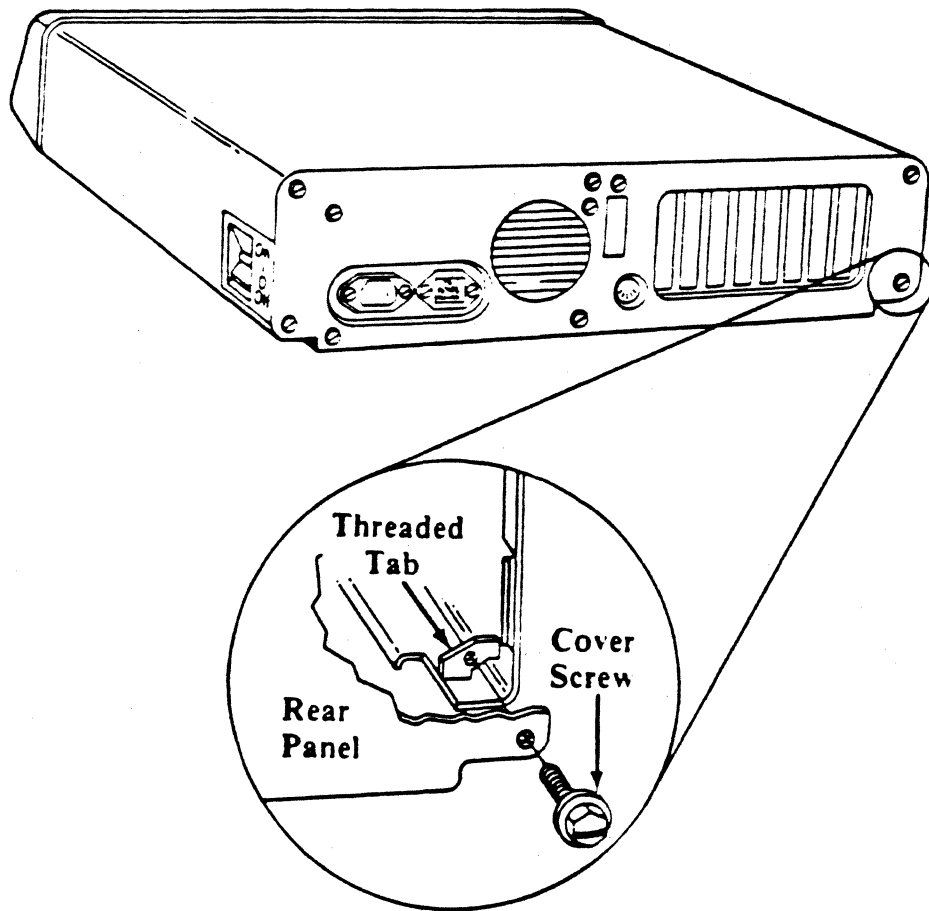
There are no adjustments on the VX/PC cards. If the image still appears incorrectly, there may be a problem in the board set. Call Vectrix Customer Service for assistance.

If the problem cannot be corrected over the phone, you can return the cards to Vectrix for repair. Be sure to ask for a Return Merchandise Authorization. Without this number, Vectrix will not accept the cards when returned. Repackage the card set and do not disconnect the cards.

The concentric circles in the image are provided to help in adjusting horizontal hold. If the circles are not perfectly round, the knob labeled HORIZONTAL on the front of the monitor can be turned until the circles appear correctly.

If all tests well, you are now ready to fasten the cover to the system unit. Press ENTER to exit the program. With the cover all the way to the rear, align the five screws with the threaded tabs and insert the screws.

Illustration 1.12
Fastening the System Unit Cover



1.4.3 SPHERE: A Demonstration Image Command File

To display the SPHERE program provided on the VX/PC disk, type, from the DOS prompt:

```
COPY SPHERE VECTRIX /B
```

The image will appear, moving down the screen. Short flashing lines will appear on the screen while the image is being loaded. When finished loading, the image will disappear. This happens because the command file contains the "SI" (Select IBM Mode) command.

The image can be made to reappear by following these instructions:

RGB Monitor System:

Press F10, which was redefined to be SV.
This will switch the system to Vectrix mode.

RGB and Monochrome Monitor System:

Type the command SV, switching to Vectrix mode.
or
Press F10, which was redefined to be SV.
This will switch the system to Vectrix mode.

1.5 INSTALLING THE COLOR PRINTER

The VX/PC prints color images on the Radio Shack CGP-220, the Canon A1210, the Quadram Quadjet, the ACT II, and the Canon PJ1080 printers.

1.5.1 Connecting the Color Printer

The five printers use a standard parallel interface connected directly to the IBM or equivalent parallel printer interface card, available from IBM, an authorized IBM dealer, and many third party sources. Connect the printer to parallel port LPT1 or LPT2.

1.5.2 Testing the Printer

The VX/PC card set can use the color printer in two ways. The first way is by invoking the external print programs, MPRINT.EXE, QPRINT.EXE, APRINT.EXE or CPRINT.EXE from DOS. MPRINT.EXE is used with the Radio Shack, QPRINT.EXE with the Quadram or Canon A1210, APRINT.EXE with the ACT II, and CPRINT.EXE with the Canon PJ1080 printer.

The second way to print images is using the VX/PC printing commands within a program. These commands are detailed in Chapter 2, Section 2.10.2. For testing purposes, use the MPRINT, QPRINT, CPRINT or APRINT program, depending on your printer. To reduce confusion, we will use the term MPRINT for any version of the program.

The first step in testing the printer is to bring up the card testing image as was done earlier. This is done with these commands, from the DOS prompt:

```
COPY TEST01.DAT VECTRIX
```

For more detailed instructions on this procedure, see Section 1.4 in this chapter.

When the image is finished drawing on the screen, press ENTER, returning the system to PC DOS.

From the DOS prompt, type:
MPRINT

if the printer is connected to LPT1.

If the printer is attached to LPT2, type

MPRINT 2

For a complete explanation of the MPRINT program, see Chapter 2, Section 2.10.1, "The Color Printing Programs".

The printer starts printing in about five seconds. The printed image should look like the screen image. If it does not, there is a problem in either the VX/PC, the printer, the interface card, or the cable.

If a problem exists, the printer and cable can be tested by attempting to print text to the printer in this manner.

At the DOS prompt, type:

PRINT TEST01.BAS (enter)

DOS will respond with the question: Name of list device [PRN]:

Type:

LPT1 (enter)

If LPT2 or LPT3 is being used, respond accordingly.

The file TEST01.BAS should start printing on the printer.

If text cannot be printed, the problem is in the printer or cable.

1.6 INSTALLING THE LIGHT PEN

1.6.1 Connecting the Light Pen

***** WARNING *****

TURN THE COMPUTER OFF BEFORE INSTALLING THE LIGHT PEN.

DO NOT PLUG THE LIGHT PEN CONNECTOR INTO THE IBM DISPLAY ADAPTER.
DAMAGE WILL RESULT. IT MUST BE PLUGGED INTO THE VX/PC.

The male DB-9 connector on the light pen connects directly to the female DB-9 connector on the VX/PC processor card. It can be connected in only one way.

1.6.2 Testing the Light Pen

TEST02.BAS is used to test the light pen. This series of commands will load and run the program from the DOS prompt:

```
BASICA TEST02.BAS
```

If the system is in BASIC already, type:

```
LOAD TEST02.BAS  
RUN
```

This program clears the screen to white with a full screen crosshair. When the tip of the light pen is placed near the screen, the crosshair will move to that position. To exit the program, press ENTER. If for any reason the crosshair does not appear, reload the program and try again. If the crosshair appears but the light pen does not move the crosshair, call Vectrix Customer Service for assistance.

1.7 FUNCTIONAL SPECIFICATIONS

1.7.1 The RGB Monitor

The Vectrix RGB monitor is a high quality, high resolution monitor well matched to the output capabilities of the VX/PC card set. If you have purchased an RGB monitor from Vectrix, skip to the next section. If a different monitor is used, it must meet these specifications:

The monitor must be RGB

Analog inputs are required (0.0 to 0.7 volts)

75 Ohm terminations to GROUND on the RGB channels

OPEN or NO termination on the H and V sync lines

25 Mhz video bandwidth

.31 mm pitch CRT

720 dot by 512 line resolution

22.3 khz horizontal scan rate

The VX/PC RGB output specifications are these:

10 microsecond horizontal blanking interval

50 to 75 Hz vertical frequency

Misconvergence of less than or equal to
.4 mm within diameter

1.7.2 Light Pen Connector Pin Designations

VX/PC PIN	SIGNAL	DESCRIPTION
#1	LPEN	Strobe low on pen detect (low)
#2	Reserved	Do not use.
#3	+5V	
#4	+5V	
#5	+5V	
#6	LPSW	Light pen tip switch active (low)
#7	GROUND	
#8	GROUND	
#9	GROUND	

Chapter 2

SYSTEM OVERVIEW

Contents	Page
2.1 HOW TO USE THIS CHAPTER.....	2-3
Introduction	
Definitions	
2.2 VECTRIX AND IBM MODES.....	2-4
2.3 ASCII vs. HEX MODE.....	2-5
2.4 GRAPHICS PRIMITIVES.....	2-7
2.5 TEXT PRIMITIVES.....	2-7
2.6 COLOR MANIPULATIONS.....	2-8
The 9 Bitplanes and The Color Table	
Color Drawing Modes	
2.7 3-D TRANSFORMATIONS.....	2-16
Clipping	
Rotating A Figure	
Rotating Around An External Axis	
Translation	
Rotating Around An Internal Axis	
Scaling	
Aspect Ratios	
2-D Use of 3-D Commands	
2.8 PIXEL AND RAM COMMANDS.....	2-28
Compressing Images With Run Length Encoding	
2.9 CROSS HAIR COMMANDS.....	2-29
2.10 COLOR PRINTING.....	2-30
The Color Printing Programs	
The Color Printing Commands: HNP, HNR, HP, and HR	
Dithered Colors	
2.11 EXPLORING THE SYSTEM.....	2-35
2-D Input Program; 3-D Input Program	

2.1 HOW TO USE THIS CHAPTER

2.1.1 Introduction

This section is intended to help both the graphics novice and the expert understand the concepts used with the VX/PC card set. For the newcomer to computer graphics there are explanations of many concepts unique to computer graphics. These include graphics and text primitives, 3-D concepts, color manipulation, and dithering. For those who are familiar with graphical concepts, the Vectrix implementation of these concepts - such as color table indexes, clipping, bitplane management, and loading and saving images - is explained.

For both groups, there are two programs at the end of this chapter to assist in exploring the 2-D and 3-D command set. The intent is for users to type in these short programs and experiment with the VX/PC command set. With these two programs, most of the 76 available commands can be used in interactive mode to demonstrate use of the VX/PC.

2.1.2 Definitions

There are several basic terms that are used consistently throughout this manual. Some common terms are defined here.

"Current Drawing Point" (CDP) is a concept used by many of the commands, including the graphics and text primitives, crosshair commands, and the pixel and RAM commands. Unless a command has a coordinate specified in its parameters, and therefore defines its own drawing point, the point at which drawing will start is the "current drawing point". This is usually the coordinate where the last command terminated. Some commands update the current drawing point; some leave it unchanged. The impact of each command on the drawing point is explained in the REMARKS sections in the REFERENCE GUIDE, Chapter 4.

"Current Drawing Color", (CDC) is simply the last color specified. Most commands (with the exception of some of the color manipulation commands), do not change the current drawing color even if a color parameter is required with the command. In these cases, the specified drawing color is used. For commands not having a color parameter, the current drawing color is used.

"Color Manipulation" refers not only to the use of the explicit color changing commands, but also to the management of the built in color index table and the management of the bitplanes.

"Transformations" are generally, but not always, performed in 3-D object space. Transformations, both 3-D and 2-D, refers to the spatial manipulation of an image on the screen. The VX/PC allows several types of transformations including rotation, scaling, and translation.

2.2 VECTRIX AND IBM MODES

When a VX/PC card set is installed in an IBM PC or XT, there are three ways the system can operate:

- 1) In IBM text mode;
- 2) In Vectrix color graphics mode;
- 3) In the IBM color graphics modes.

Each of these modes will work with either one monitor (high quality RGB only) or two monitors (high quality RGB and separate monochrome monitor and adapter card). However, there are minor differences in how the systems can be used.

When a single monitor is used, all output - text and graphics - will go to the available monitor. Because the system cannot be in two modes simultaneously, modes must be switched to move from the high resolution graphics environment to IBM text or IBM graphics modes.

For more information about typical system setups, see Section 3.1.1, "USING ONE MONITOR vs. TWO MONITORS, in Chapter 3, and Section 3.1.3, "SV.EXE, SI.EXE, and RESET.EXE", also in Chapter 3.

When dual monitors are used, the graphics for both the Vectrix and IBM modes will be performed on the RGB monitor. Text mode output will go to the monochrome monitor.

When using a dual monitor system, the two PC DOS command sequences, MODE CO80 and MODE MONO, can be used to switch between IBM color card and monochrome modes.

When a dual monitor system is powered on or RESET, the system will be in IBM mode and text mode. To change to IBM color graphics mode, the command MODE C080 (or MODE C040 if 40 column graphics mode is desired) should be given from DOS, optionally in a batch file. The MODE program, an external DOS file, can be found on the PC DOS system disk.

If Vectrix mode has previously been set with the SV command, the SI (Switch to IBM mode) command must be set before the MODE C080 command is given. If this is not done, the system will respond with "Invalid Parameter" and must be reset.

To send text output to the monochrome monitor, make sure the system is in IBM mode, then give the command MODE MONO. This is another PC DOS external command found on the IBM PC DOS system disk.

2.3 ASCII VS. HEX MODE

The VX/PC cards and the host IBM computer regularly pass data back and forth. These transmissions can be in either ASCII (decimal) form or in hexadecimal format. There are advantages and disadvantages to using each format. The prime advantage for ASCII is ease; the main reasons for hex mode are speed and smaller data files.

Using hexadecimal mode, as opposed to ASCII mode, can greatly reduce storage requirements and access time for images. Hex files created by a program do not have the overhead of spaces, carriage returns, and line feeds that ASCII files must have. Also, ASCII characters must be reduced by the VX/PC to their hex values before the processor can execute the commands and interpret data received.

Transmission modes can be set directly with the KD (Set Decimal Mode) and HX (Set Hex Mode) commands. There is also a second method for setting each mode. This can be done with the B (Bitplane Write Mask), C (Set Color), E (Erase Screen), and Q (Define Color Lookup Table) commands. The VX/PC changes the mode (ASCII or Hex) automatically, based on the format of the parameters with these commands. The format change then stays in effect until a direct mode change is given, or until the next B, C, E, or Q command is issued. A typical series of commands is presented next, using both modes for comparison.

PRINT #1, "M300 300 A100 0 3600" - requires 20 bytes
to be transferred.

PRINT #1, "M" + CHR\$(44) + CHR\$(1) + CHR\$(44) + CHR\$(1)
+ "A" + CHR\$(100) + CHR\$(0) + CHR\$(0) + CHR\$(0) +
CHR\$(16) + CHR\$(14) - requires 12 bytes to be transferred.

While the hex mode example requires longer commands when written in BASIC, the command strings created using hex mode are considerably shorter than those created with ASCII mode. Transmission time, rather than execution time and program size, is the more important consideration here.

When hex mode is used, any 16 bit data sent to or received from the VX/PC will be in the standard binary format: the Least Significant Byte (LSB) is sent first, the the Most Significant Byte (MSB) is sent. For example, the value 3600 is sent as CHR\$(16) + CHR\$(14).

The formulae for calculating the values to send based on a decimal value are:

$$\text{MSB} = \text{INT} (\text{Value} / 256) = \text{INT} (3600 / 256) = 14$$

$$\text{LSB} = \text{Value} - (\text{MSB} * 256) = 3600 - (14 * 256) = 16$$

Note the use of the CHR\$(x) format for hex data. Hex data cannot be sent in strings of hex characters as "0E10".

Using either ASCII or hex data transmission, the output of a program can be sent to a special type of file - a command file. Data captured in a command file can be sent to the VX/PC very quickly using the COPY command from DOS. Since the file contains the complete output of the program, data is passed to the VX/PC without any of the calculation time required by the program.

When COPYING a command file, the /B option is recommended because it is faster and does not filter any characters. The format is COPY xxxx.xxx/B, where xxxx.xxx is the name of the command file. The extension normally used is .DAT, which keeps command files identifiable.

Further information on ASCII and Hex operating modes can be found in Chapter 4, Section 4.2, "COMMAND ARGUMENTS".

2.4 GRAPHICS PRIMITIVES

The VX/PC graphics processor offers a wide variety of geometric forms that can be combined on the display to create a virtually infinite number of images. These basic forms are called graphics primitives. The Vectrix command set includes primitives to draw dots, lines, polygons, arcs, patterns, and several types of color fills and floods.

Most of the commands in this category work in both 2-D and 3-D modes: D (Dot), M (Move), L (Line), and P (Polygon), and F (Filled Polygon) commands work in either 2-D or 3-D space. Rectangle Fill and any command producing an arc - A (Arc), OA (Originate Arc), WA (Wedge Arc), or WB (Wedge Arc Boundary Filled) - work in both modes, but in 3-D mode use a 2-D coordinate implementation.

2.5 TEXT PRIMITIVES

The VX/PC card set has provisions for standard and custom character sets. The default set resides in PROM aboard the VX/PC - no downloading is needed to use the characters. Custom sets can be created using the JD (Design New Characters) command and will then reside in the VX/PC RAM.

The default set has 95 characters - the standard characters from ASCII 32 to ASCII 126. These include all upper and lower case alphabets, numbers, basic mathematical symbols, and many other special characters.

Characters can be mixed freely on the display with any graphic image; displayed at any screen location in a variety of sizes and rotations, with or without custom horizontal, vertical and intercharacter spacing.

Any of these characters can be re-defined using the JD command. This is often done for calligraphic type fonts and to create symbols not otherwise available, such as the copyright symbol. There are several rarely used ASCII characters that can be used when only one or a few characters are to be redefined. These are ASCII 91 through 96 and ASCII 123 through 126.

The current custom character set can be returned to the default set at any time with the JN, G, G0 commands and by executing the RESET.EXE program found on the VX/PC disk.

When the text commands are used in 3-D mode, they cannot be rotated, scaled or translated using the 3-D commands R (Rotate X, Y, or Z), S (Scale X, Y, or Z), or T (Translate X, Y, or Z).

2.6 COLOR MANIPULATIONS

The color manipulation commands allow precise color control for graphic and text primitives and the display background. Color effects can be mixed in several ways to create very subtle shading effects. Colors can also be used to hide images in the background until a new color definition puts them in the foreground, making possible animation and hidden drawing effects.

2.6.1 The 9 Bitplanes and The Color Table

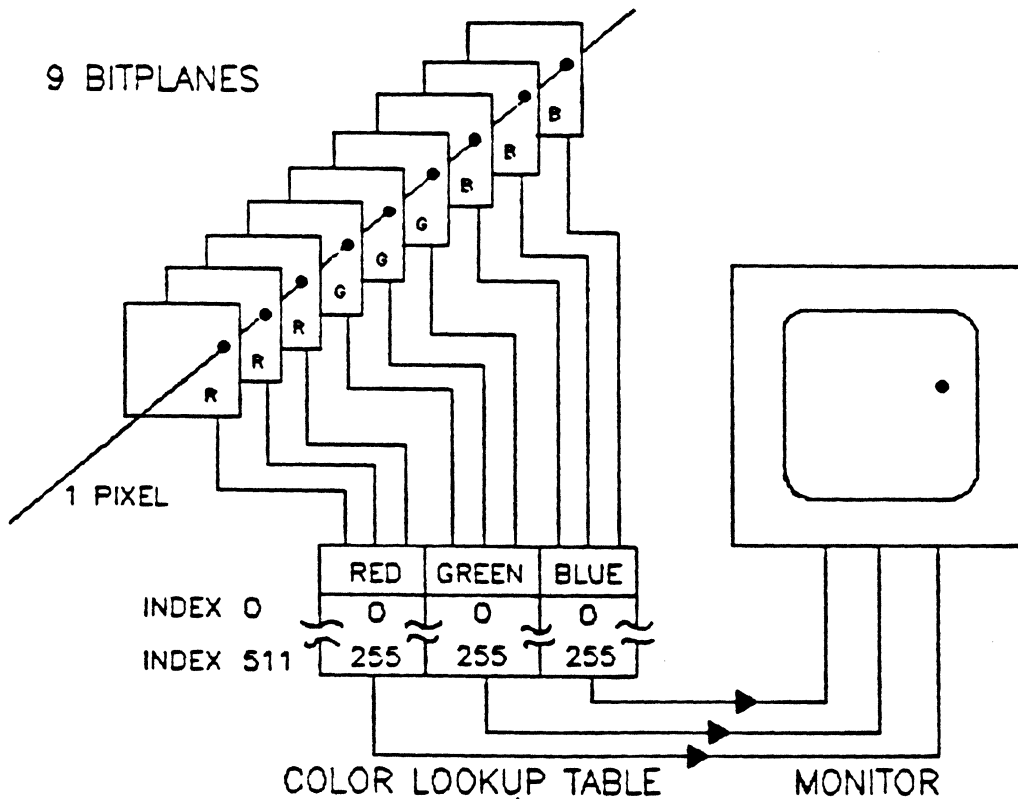
The graphics memory on the VX/PC is organized as nine bitplanes, and each pixel's color is determined by combinations of the nine bits. VX/PC uses the selected color number as a nine bit index to the color lookup table where the color values are stored. Since there are 2^9 combinations possible, 512 colors can be stored in the color table at any given time.

When a color table entry is referenced, three stored eight bit values are output to the RGB monitor. These three values are the amounts of red, green, and blue an entry will display. The three values control the intensities of each of the three color guns in the monitor. While each of the three color components stored in the color table has eight bits, the lower four bits are masked off and are not used in the standard VX/PC configuration. Since there are four active bits for each component, there are 2^{12} or 4096 possible colors.

Changing the contents of a color table entry changes the color that will be displayed when a color command references that color table entry. When a displayed image uses a particular color table entry, and that entry is changed, the display will immediately reflect the new contents. For example, a red color on the screen can be changed to grey by setting each of the entry's three components, red, green, and blue, to the same value.

A complete 16 entry grey table can be built by giving entries 0 through 15 equal incremental values for red, green, and blue. For example, entry #1 would contain a value of 16 for the red component, 16 for the green component, and 16 for the blue component. Entry #31 could contain 32, 32, 32 for the RGB components. This continues until the 16th table entry is changed. A sample grey table is given in Illustration 2.2B.

Illustration 2.1
The Nine Bitplanes



The default color table's 512 entries are arranged so that the lower three bits of the color number index represent the red components, the middle three bits represent the greens, the high bits represent the blues. As the value of each group of bits increases, the intensity of its respective color increases.

The algorithm used to determine the default color table values is this:

The last entry, #511, is set to R=255, G=255, B=255.

Red, green and blue values are decreased by 32 in nested loops until all three equal 32. Red is decremented first, then green, then blue.

The first entry, #0, is set to R=0, G=0, B=0.

NOTE: Even though each color table component can have values ranging from 0 through 255, the lower four bits are masked off in the standard configuration of the VX/PC. When the 16 million color option is added, all eight bits will be used.

This indicates that the value 255 is interpreted the same as 250 or 245. The value 15 in any color table component will be understood as a zero, since all significant bits are zero.

A partial default color table is given in Illustration 2.2A, and a partial grey table in 2.2B.

Illustration 2.2
The Default Color Table and A Custom Grey Table

DEFAULT LOOKUP TABLE

Index	Red	Green	Blue
0	0	0	0
1	64	32	32
2	96	32	32
3	128	32	32
~~~~~			
511	240	240	240

GREY LOOKUP TABLE

Index	Red	Green	Blue
0	0	0	0
1	16	16	16
2	32	32	32
3	48	48	48
~~~~~			
15	240	240	240

2.6.2 Color Drawing Modes

There are four modes controlling how colors are displayed when put on top of existing colors on the screen. Colors can be made to mix, replace, or complement each other using the OR, RA, RC, or RE commands.

Three additional factors affect color display:

1. Setting the Bit Plane Mask Register
2. Setting the Pattern Register
3. The original color and the overlaid color

The Bitplane Mask Register, set with the B command is the highest level of control. If any bit in this register is zero, no change will occur to bits in the corresponding bitplanes.

The next level of control is the Pattern Register, set with the N or the JR command. The displayed result of a bit in this register being on or off is determined by the color drawing mode.

A functional description of each mode is given next. The old color refers to an existing color on the screen. The new color is the color currently selected. The results indicate the new color table index used when the old and the new colors overlay each other on the screen.

NOTE: Color mixing refers to the mathematical manipulation of color table indexes; it does not directly refer to visual changes.

MODE	NAME	RESULT
OR	OR	Mix old color with new. Colors are logically ORed together.
RA	Replace All	Replace old color with all ones if pattern bit is one, or all zeroes if pattern bit is zero. Ignore drawing color.
RC	Replace Complement	Replace old color with its complement. Ignores new color.
RE	Replace	Replace old color with new. If pattern bit = 0, bit is unchanged.

Illustration 2.3 describes what happens to individual bits in graphics memory under all combinations of these four factors. All color bits must be looked at as a group to determine the actual color displayed.

Illustration 2.3
Interaction Of Bitplane Mask Register, Pattern Register, Color, and Mode

COLOR DRAWING MODES	Bitplane Mask Bit = 1		Bitplane Mask Bit = 0	
	Pattern Bit = 0	Pattern Bit = 1		
		Color Bit=0	Color Bit=1	
	OR	M	M	1
RA	0	0	1	M
RC	M	-M	-M	M
RE	M	0	1	M

LEGEND: 0 = Result is a binary Zero
 1 = Result is a binary One
 M = Memory Contents Unchanged
 -M = Result is Memory Complemented

The example in Illustration 2.4 demonstrates how the bits describing the contents of graphics memory will mix using each color drawing mode when colors are overlaid. It is assumed that the Bitplane Register and the Pattern Register contain all ones, the default setting, allowing update of all bitplanes.

Illustration 2.4
Color Drawing Mode Bit Examples

```

=====
Bit Number          1   2   3   4   5   6   7   8   9
Bit Value           1   2   4   8  16  32  64 128 256
=====

Original Color = 7   1   1   1   0   0   0   0   0   0
Overlaid Color = 57  1   0   0   1   1   1   0   0   0
-----

Resulting Colors by Mode:
OR                   1   1   1   1   1   1   0   0   0 = 63
RA                   1   1   1   1   1   1   1   1   1 = 511
RC                   0   0   0   1   1   1   1   1   1 = 504
RE                   1   0   0   1   1   1   0   0   0 = 57
=====

```

This program can help clarify which mode creates which effects. Type in and run the color drawing modes demonstration program.

Color Drawing Modes Demonstration Program

```
10 REM ***** COLOR DRAWING MODES *****
20 OPEN "VECTRIX" FOR OUTPUT AS #1
30 LET A$="RF 100 100"
40 PRINT #1, "SV G E7 C56"
50 PRINT #1, "OR M 100 300";A$
60 PRINT #1, "RA M 226 300";A$
70 PRINT #1, "RC M 352 300";A$
80 PRINT #1, "RE M 478 300";A$
90 PRINT #1, "JR 129 66 36 24 24 36 66 129"
100 PRINT #1, "OR M 100 150";A$
110 PRINT #1, "RA M 226 150";A$
120 PRINT #1, "RC M 352 150";A$
130 PRINT #1, "RE M 478 150";A$
140 PRINT #1, "JM3 M100 100"
150 PRINT #1, "OR$HELLO "
160 PRINT #1, "RA$HELLO "
170 PRINT #1, "RC$HELLO "
180 PRINT #1, "RE$HELLO "
190 INPUT A$
200 PRINT #1, "SI"
210 CLOSE
220 END
```

The program displays eight squares, four with pattern and four without, each of whose colors are completely different. The difference between the squares in each set is due to the color mixing modes with which they were drawn.

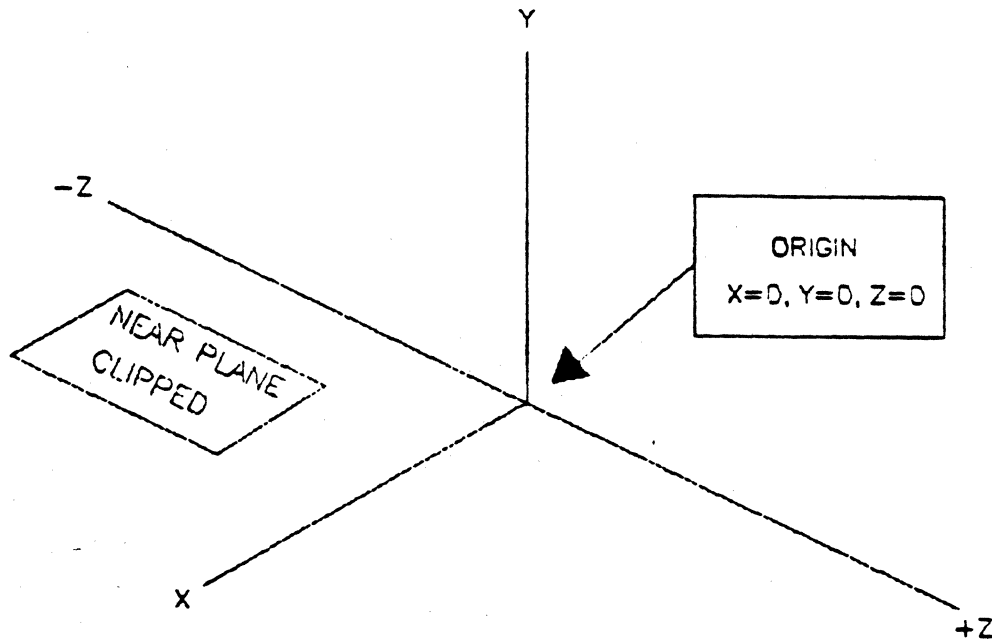
2.7 3-D TRANSFORMATIONS

The 3-D commands provide ways to manipulate an object's orientation, to move a figure to a specific part of the display, and to manipulate aspect ratios.

In 3-D mode, drawing coordinates sent to VX/PC do not represent physical screen coordinates; instead object space coordinates are represented. Object space coordinates can vary from -32767 to +32767 along all three axes. The processor automatically transforms 3-D images into the two dimensional screen coordinates needed to display a 3-D image on a physical device. This process uses 3-D vanishing perspective and clipping techniques.

When in 3-D mode, the eye position, or origin, where $X=0$, $Y=0$, and $Z=0$, is considered always to be at the center of the current viewport. When the viewport is full screen, the origin is the point where $X=336$ and $Y=240$ in 2-D mode.

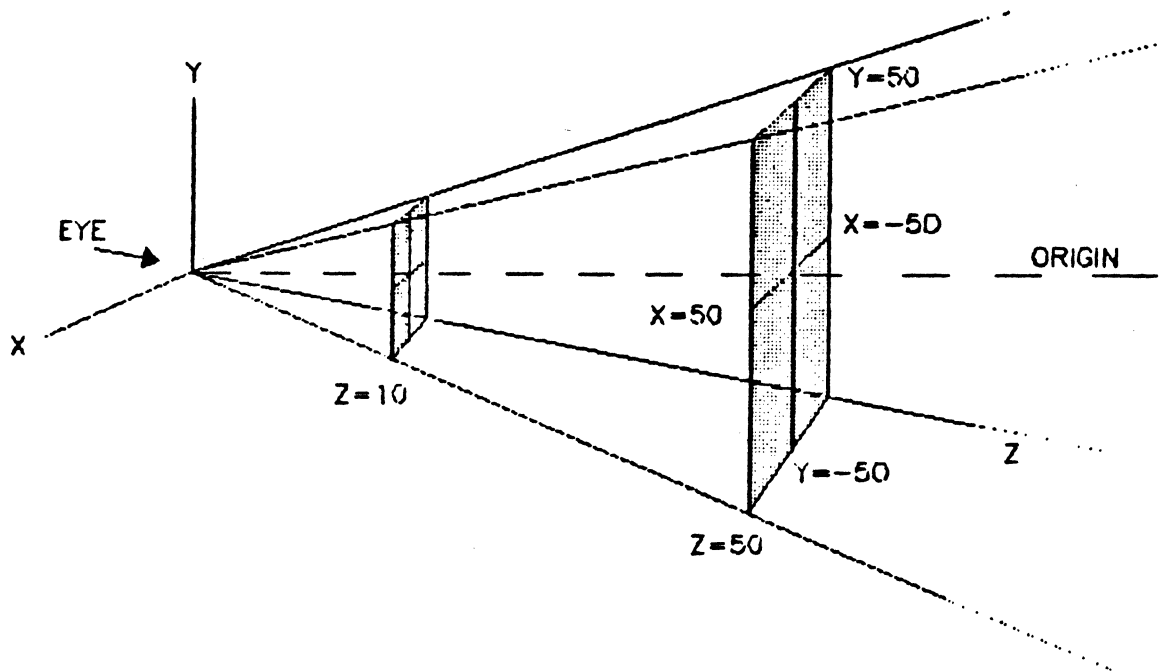
Illustration 2.5
3-D Object Space



2.7.1 Clipping

The viewing pyramid defines the area of 3-D coordinates that will be clipped.

Illustration 2.6
The Viewing Pyramid



Figures will be clipped if:

- 1) The Z coordinate is less than 0. This is known as near plane clipping.
- 2) The X or Y coordinates are greater than the Z coordinate.

Figures will not be clipped, and therefore visible, if:

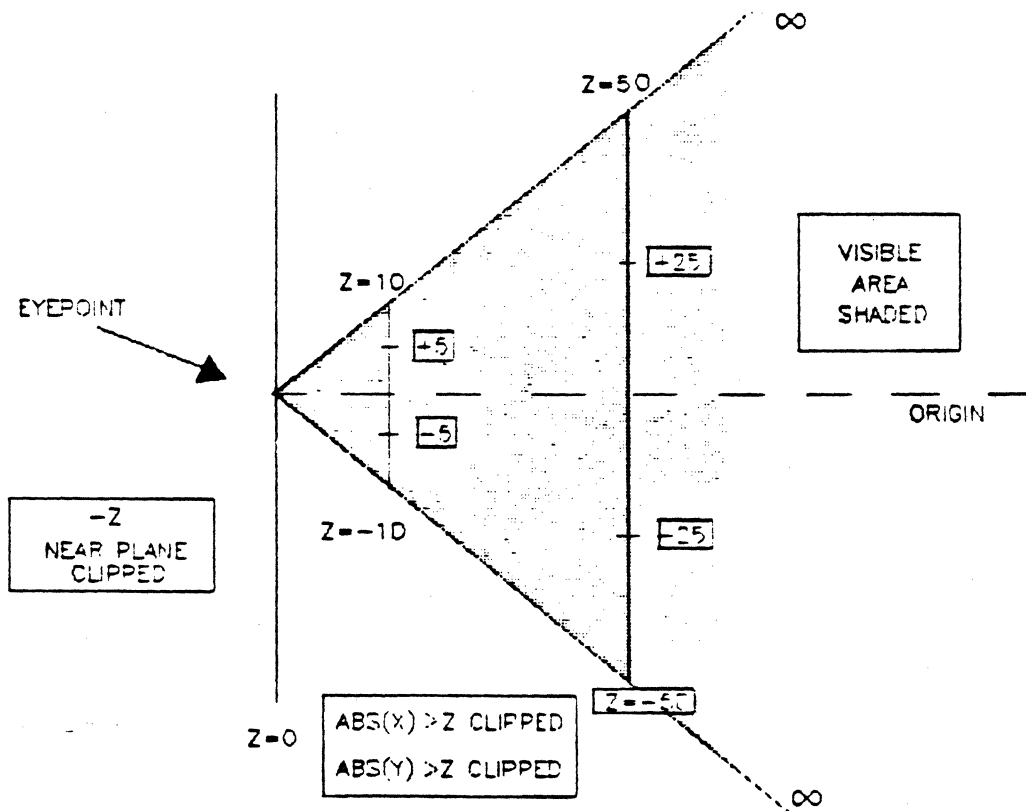
The Z coordinate is greater than 0, and the absolute value of X and Y are less than or equal to the Z coordinate.

For example, when the Z coordinate is 10, the visible X and Y values are from 10 to -10. When the Z value is 50, the visible X and Y values are 50 to -50. The center of the viewport is always considered to be $X=0$, $Y=0$, and $Z=0$. So when $Z=10$, an X value of +5 will be the point halfway from the center of the viewport to the viewport boundary along the X axis. If the viewport is the full screen, $X=10$ will be a point on the right edge of the screen and $X=-10$ will be a point on the left edge of the screen.

However, when $Z=50$, with a full screen viewport, the right edge of the screen represents $X=50$; $X=-50$ is the left edge.

Positive Z coordinates are only clipped if the absolute value of X or of Y is greater than Z.

Illustration 2.7
Another View of The Viewing Pyramid

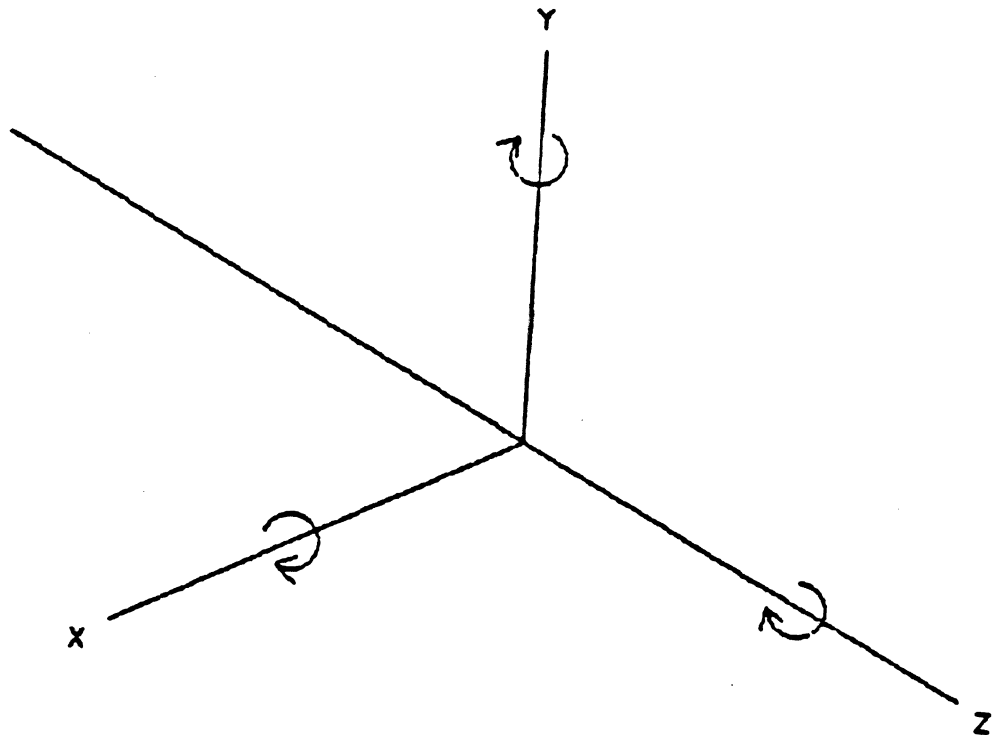


2.7.2 Rotating a Figure

Two types of rotation can be performed in 3-D mode, on any of the three axes, X, Y, or Z:

- 1) Rotating an object around an external axis.
- 2) Rotating an object around its own axis.

Illustration 2.8
Direction of Rotation Around X, Y, and Z Axes



NOTE: Rotation around an axis should not be confused with the direction a figure moves. For example, rotating around the X axis will actually move a figure in the Y-Z plane.

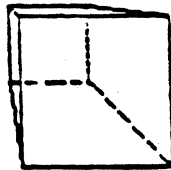
2.7.3 Rotating Around An External Axis

To rotate an object around an external axis:

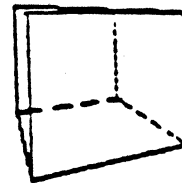
The origin is almost always external to the object if the object is visible. Use the R (Rotate) command followed by the axis to rotate on (X, Y, or Z), and the number of degrees to rotate it. No translation need be done. The object will rotate around the origin.

Illustration 2.9
Rotation Around an External Axis

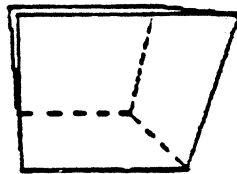
ROTATE THREE-DIMENSIONAL OBJECTS AROUND EXTERNAL AXES



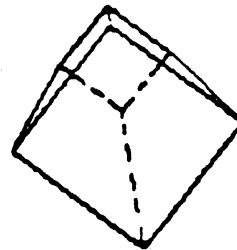
UNROTATED CUBE



30° AROUND Y AXIS



30° AROUND X AXIS



30° AROUND Z AXIS

A typical series of commands to rotate an object around an axis external to the object would be:

```
K3          - Select 3-D mode
I           - Initialize translation matrix
V           - Set viewport if desired

Draw image  - Optional

RX, RY, or RZ - Rotate object around X, Y, or Z

Draw image, with or without first clearing screen
```

2.7.4 Translation

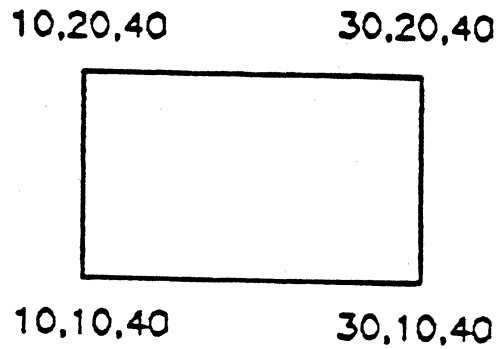
Translation is the process of moving a figure in object space. Figure replication is simplified using the translate command to select new coordinates at which to redraw a figure.

Translation can also be used to move a figure to the origin in preparation for later transformations, including rotating and scaling. This method allows a figure to spin, or rotate around its own axis.

The necessary matrix mathematics for translations is performed internally by the VX/PC processors, removing that burden from applications programs.

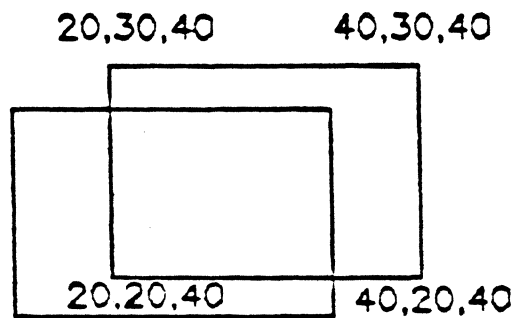
Illustration 2.10
Translating A Figure

A. Original Figure



ORIGINAL FIGURE

B. Translation



AFTER TRANSLATING
TX 10, TY 10

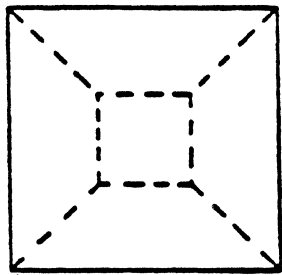
2.7.5 Rotating Around An Internal Axis

To rotate an object around its own (internal) axis, the correct procedure is:

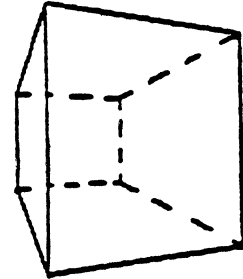
- 1) Translate the figure to the origin
- 2) Rotate it on one or more axis
- 3) Translate it back to the desired position

Illustration 2.11
Rotation Around an Internal Axis

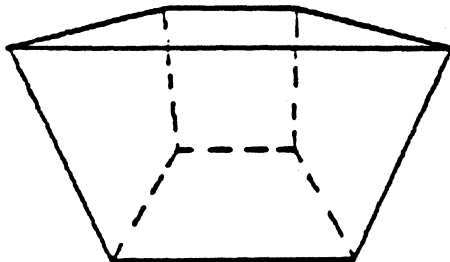
ROTATE THREE-DIMENSIONAL OBJECTS AROUND INTERNAL AXES



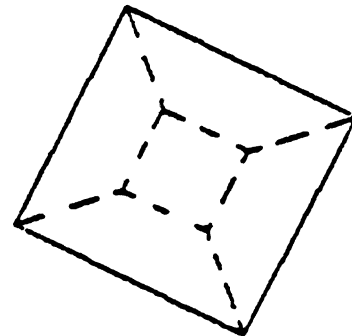
UNROTATED CUBE



30° AROUND Y AXIS



30° AROUND X AXIS



30° AROUND Z AXIS

The key is that rotation is always done around the origin. If this point is at the center of the object, it will spin on its own axis. If the object does not contain the origin, the entire figure will move around the origin.

The T (Translate) command can be used to move the object toward and away from the origin. The actual translation is not visible - nothing new will be displayed until the image is drawn or redrawn.

A typical series of commands to rotate or spin an object around its own axis would be:

```
K3 - Select 3-D mode
I  - Initialize translation matrix
V  - Set viewport if desired
```

Draw image (optional)

```
TX - Translate X to 0
TY - Translate Y to 0
TZ - Translate Z to 0

R  - Rotate on X, Y, or Z axis or any
    combination

TZ - Translate back to original Z coordinate
TY - Translate back to original Y coordinate
TX - Translate back to original X coordinate
```

Draw image, with or without first clearing screen.

To get a figure to rotate on its center, the figure should be translated so its center, and not an edge, is at the origin. For instance, if the Z coordinates used range from 10 to 50, the center of the Z axis is at Z=30. So translating to origin requires this command: TZ -30. The same applies to the X and Y axes.

2.7.6 Scaling

Scaling operations are similar to rotations because the figure can be scaled with respect to the axes of the origin or with respect to the axes of the figure's center. The latter requires translating the figure to the origin before scaling.

A method of scaling a figure around its own center would be:

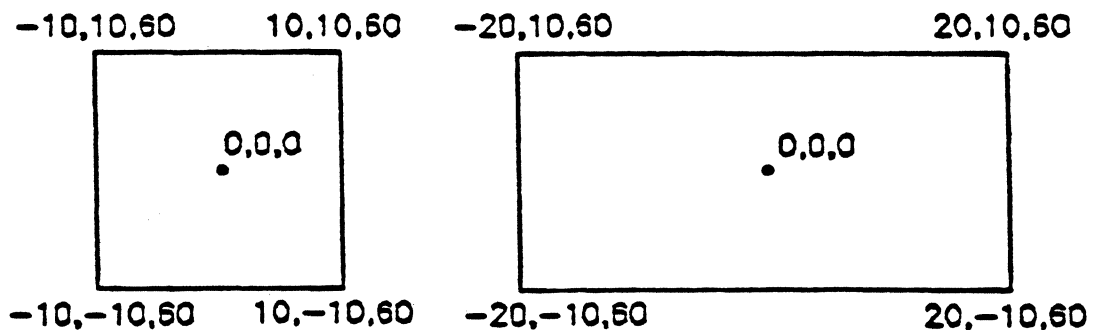
- 1) Draw figure (optional)
- 2) Translate to the origin of $X=0$, $Y=0$, and $Z=0$.
- 3) Perform scaling on the X , Y , or Z axis
- 4) Translate the figure to the desired position.

Scaling a figure around the origin requires that scaling be performed and the figure be drawn. However, the results are completely different than when using the translation method.

Illustrations 2.12A and 2.12B show the results of the two methods on a simple polygon.

Illustration 2.12A
Scaling Around a Figure's Center

ORIGIN INTERNAL TO FIGURE

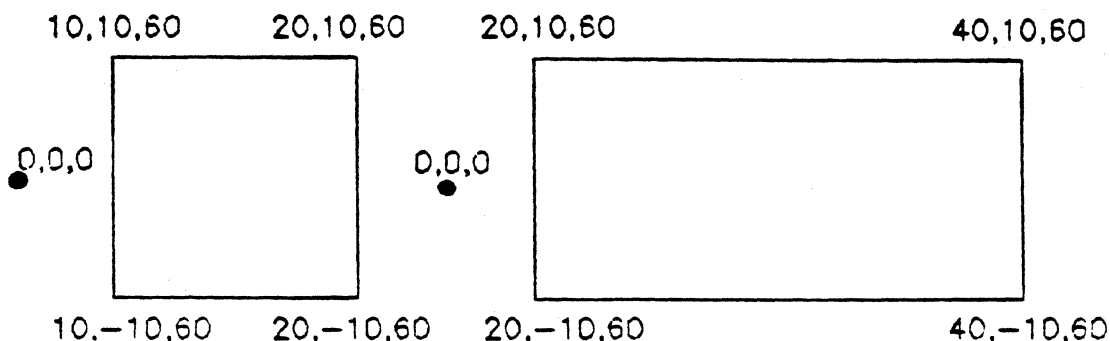


ORIGINAL FIGURE

AFTER SCALING SX 2,1

Illustration 2.12B
Scaling Around An Origin External To A Figure

ORIGIN EXTERNAL TO FIGURE



ORIGINAL FIGURE

AFTER SCALING SX 2,1

2.7.7 Aspect Ratios

Viewports can be used to manipulate aspect ratios in 3-D mode. The usual reason for changing ratios is to create square viewports. Occasionally, rectangular viewports are desirable for manipulating an object. The default viewport is the entire screen (Ø 671 Ø 479), a rectangle. It is important to note that a square drawn with the default viewport will not be truly square.

Three techniques for creating a square aspect ratio are:

- 1) By using a square viewport such as: V 96 575 Ø 479, which sets the largest square ratio available (see Viewport command).
- 2) By using a scale factor of 480 divided by 672 as the final step before drawing: SX 480,672.
- 3) By using the Set Perspective Scaling command (SP). (See the SP command in Chapter 4, "Reference Guide".)

2.7.8 2-D Use of 3-D Commands

Rotation, scaling, and translation also work in 2-D mode. The commands able to use 2-D transformations are: Dot, Move, Line, Polygon, and Filled Polygon. Not affected by 2-D transformations are the arc commands (A, OA, WA, and WB) and the Rectangle Fill command (RF).

2-D clipping is performed for these commands and for Rectangle Fill and the Arc commands. Next is an overview of 3-D commands and their effects in 2-D mode.

RZ - Rotates along Z axis. Positive rotation is clockwise. Negative rotation is counterclockwise.

SX - Scale along X axis.

SY - Scale along Y axis.

TX - Translate along X axis.

TY - Translate along Y axis.

RX, RY, SZ, and TZ - have no effect in 2-D mode.

V - Viewport is active in 2-D mode, however it serves only as a clipping boundary and will not change aspect ratios.

All 2-D rotations and scalings are performed in respect to the origin ($X=0, Y=0$), which is the lower left corner of the screen. In 3-D mode, the origin is at the center of the current viewport.

2.8 PIXEL AND RAM COMMANDS

The VX/PC cards have built in commands to transfer graphics RAM directly from the VX/PC to the host or from the host to the VX/PC. Memory can be read and written pixel by pixel or bitplane by bitplane. While reading or writing, the memory data containing the screen image can be compressed or maintained in its original form.

The Pixel commands RP (Read Pixels), RNP (Read Pixels Encoded), WP (Write Pixels), and WNP (Write Pixels Encoded), read and write pixel color values directly to and from graphics memory. The four commands operating on pixels read or write all bitplanes at once, giving a specific color, regardless of the setting of the Bitplane Mask Register.

The RAM commands RR (Read RAM), RNR (Read RAM Encoded), WR (Write RAM), and WNR (Write RAM Encoded) read and write graphics memory as binary words on one bitplane at a time. The use of these commands is usually faster than the Pixel commands.

Both types of commands, Pixel and Graphics RAM, start operations at the current drawing point and proceed left to right on the screen. Both can be used in ASCII or Hex modes, and Flash or Blank Video modes.

2.8.1 Compressing Images With Run Length Encoding

Both command types can read or write raw data or can encode the data using Run Length Encoding. The unencoded versions of the commands are used to manipulate small areas of the screen. The encoded versions are usually used when an entire screen or bitplane is to be saved, because the amount of data to be transferred can be greatly reduced.

The RNP and RNR commands use a process called Run Length Encoding to compress images, often achieving a five to one compression ratio. These two commands are similar to the unencoded versions but take the transfer several steps further.

The Read Encoded Pixel command (RNP), starting at the current drawing point, reads the color displayed and continues reading pixel colors until a different color is found. A counter keeps track of any color repetitions found. When the new color is found, the VX/PC processor sends a 16 bit word of data to the host which includes both the color value and the number of repetitions. It then proceeds on to the next pixel. This process continues until either the end of the screen is found, or the specified pixel count is met.

The Read Encoded RAM command operates in a similar manner, but reads only one bitplane at a time, and encodes only words of all zeroes (0000H) or all ones (FFFFH). Other word values are sent unchanged to the host as they are read. Reading and transferring a entire image requires 9 separate commands, one for each bitplane.

The commands WNP and WNR can be used to restore an image compressed with the RNP and RNR commands, but data types cannot be mixed. If an image is stored pixel by pixel, it cannot be properly restored using the WNR command.

In general, the amount of compression is determined by the characteristics of an image. Pictures with large contiguous areas of a particular color will compress better than pictures where there is little color repetition. Along with reducing the stored size of an image, these commands speed up loading and saving operations - less data is transferred between the IBM and VX/PC.

2.9 CROSSHAIR COMMANDS

There are five related commands used to control internal crosshairs. Each can be turned on and off, set to a specific size, set to a specific location, and the system can return the current location of the crosshair.

All crosshair values are accepted and returned in physical screen coordinates. These can be 0 to 671 along the X axis, and 0 to 479 along the Y axis.

Crosshair size is not affected by viewing transformations such as scaling, rotation, and translation. Size, horizontal and vertical orientation, and screen location remain the same throughout these transformations. The relationship between the physical screen coordinates and the information on the screen is the user's responsibility.

Crosshairs are drawn in the complement of the color covered. On a white screen, crosshairs will be black; on a black screen, crosshairs will be white. The colors are complemented based on their color table index value. In the default color table, white is the complement of black and vice versa. If the color table is changed, this may no longer be a valid assumption. When the crosshair intersects objects on the screen, the crosshair color will change to reflect the complement of the color currently being covered.

All crosshair commands can be drawn in both 2-D and 3-D modes, although always displayed in 2-D. In both cases, the crosshair is clipped at viewport boundaries.

2.10 COLOR PRINTING

The VX/PC command set provides access to printer drivers for printing any screen image to the Radio Shack CGP-220, Canon A1210, Canon PJ1080, Quadjet, or Advanced Color Technology ACT II color ink jet printer. To use the printing commands, a printer must be connected to one of the IBM parallel printer ports, LPT1, LPT2, or LPT3.

There are two basic types of printing the Vectrix command set can perform: non-dithered and dithered. Non-dithered printing allows for eight colors created from the four ink colors. When reproducing images from the monitor, the closest of the eight colors is printed. Non-dithered printing can be done with the commands HNP and HNR or with the MPRINT.EXE, CPRINT.EXE, QPRINT.EXE or APRINT.EXE programs.

The dithered printing commands work differently. After finding the first color, the color table entry for that color is scrutinized. The three bytes, one each for red, green, and blue, are read. If any one is less than 127, it is ignored. If the value is greater than 127, the color is printed. When a color table entry contains values greater than 127 for two or more of the three bytes, the commands to mix or overlay primary colors are passed to the printer. Then the next color is read from memory.

Knowing this pattern can come in useful if a certain color is not printing well. The color table values can be manipulated to make the values greater or less than 127 to emphasize or remove emphasis from any particular color. Dithered printing can be done with the dithered printing commands (HP or HR) or with the MPRINT.EXE, CPRINT.EXE, QPRINT.EXE, or APRINT.EXE programs.

2.10.1 The Color Printing Programs

Provided on the VX/PC disk are the executable programs MPRINT.EXE, CPRINT.EXE, APRINT.EXE and QPRINT.EXE. MPRINT is used to print to the Radio Shack printer, QPRINT prints to the Quadjet or Canon A1210 printers, CPRINT to the Canon PJ1080 printer and APRINT to the ACT II. To reduce confusion, we will use the term MPRINT to refer to the program for any printer. The three programs use similar parameters and function in the same manner.

MPRINT prints the contents of the screen when it is called from PC DOS. When operating, the program calls the dithered or non-dithered printing routines residing in PROM, on board the VX/PC.

There are four optional parameters used with this program. The parameters can be passed in any order, and can be left out when the default controls are desired.

- 1) The first parameter is for eight color (non-dithered) screen print or for 125 color (dithered) printing. An "8" indicates 8 color, and a "1" prints in 125 colors.
- 2) The second parameter is for reversed or normal printing. Normal printing prints the screen exactly as the screen appears. Reversed printing prints black where the screen is white and prints white (leaving the paper unchanged), where the screen displays black. "R" reverses black and white.
- 3) The third optional parameter controls which printer port will be used: LPT1, LPT2, or LPT3. The parameter is a "1", a "2", or a "3".
- 4) The fourth parameter can be used with only the Quadjet or ACT II printers. A "D" indicates double strike on the Quadjet, and double size on the ACT II.

The codes for the printing parameters are listed next.

#	CODE	MEANING
1	1 (or blank) 8	125 color dithered printing 8 color non-dithered printing
2	blank R	Normal printing Reversed printing
3	1 (or blank) 2 3	Use printer connected to LPT1: Use printer connected to LPT2: Use printer connected to LPT3:
4	D D	Double-strike on Quadjet Double size on ACT II. This doubles the size of the left 3/4 of the screen.

Two examples of MPRINT options:

MPRINT - prints the screen using up to 125 colors, in normal fashion, through LPT1. All parameters are default.

MPRINT 8R2 - prints in 8 color mode, reversed, and using LPT2.

NOTE: Due to limitations of the printer, the MPRINT.EXE program prints from pixel 16 to pixel 655 of the screen; 640 pixels are printed.

2.10.2 The Color Printing Commands: HNP, HNR, HP, and HR

The HNP, HNR, HP, and HR commands are available for direct use. These commands return data representing the screen image to the host computer; the commands do not also send the data to the printer. This is a separate operation.

Using assembly language and the supplied MPRINT source code, MPRINT.ASM on the VX/PC disk, the code to do this can be developed. Using some high level languages, including BASIC, a way must be devised to receive the data without having the language trap characters such as Ctrl Z and carriage returns. A sample BASIC program is provided next as a guide.


```

10 OPEN "VECTRIX" FOR OUTPUT AS #1 : PRINT #1,"SV HR"
20 CLOSE #1
30 WIDTH "LPT1:",255
40 ON KEY(1) GOSUB 90 : KEY(1) ON
50 OPEN "R",1,"VECTRIX",128: FIELD 1,128 AS C$
60 GET 1
70 X$ = C$ : LPRINT X$;
80 GOTO 60
90 CLOSE #1 : OPEN "VECTRIX" FOR OUTPUT AS #1
100 PRINT #1,"SI"
110 CLOSE : END

```

An alternative to writing a program can be used from IBM's BASICA. This method uses the SHELL command, from BASICA, to call a DOS routine (in this case MPRINT), execute the routine, and return to BASICA at the next statement. SHELL can also be used from within BASICA in the immediate mode. Two samples of the correct syntax for this method are given next.

```
SHELL "MPRINT"           (immediate mode, no parameters)
```

```
100 SHELL "MPRINT 8R2" (deferred mode, explicit parameters)
```

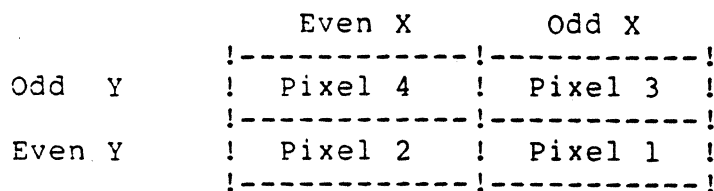
2.10.3 Dithered Colors

Dithering is the process of creating patterns of dots, simulating to the eye apparent colors and shades. The dithered printing commands (HP and HR) approximate the printing of up to 125 colors.

Dithering is performed by first reading the color table values for each color. Based on these values, the algorithm decides on the pattern to be used when printing the color. The algorithm is designed for situations where the color to be printed covers a two pixel by two pixel or larger area. If only one dot of a color is found, results will be different, based on whether the coordinate printed is even or odd in the X and Y axes.

The 2 x 2 matrix is set up as shown in Illustration 2.13.

Illustration 2.13
Dithering Patterns



These even and odd coordinates represent pixels on the screen. The color values determine the actual printing pattern as given next.

Color Table Value	Pattern	Comments
0 - 31	0 ! 0 --!-- 0 ! 0	Prints a blank area, with no color
32 - 63	0 ! 1 --!-- 0 ! 0	Prints 1 of 4 possible dots. If one dot, will print only if position on paper is odd in both X and Y directions.
64 - 127	0 ! 1 --!-- 1 ! 0	Prints 2 of 4 dots. In small areas of color, prints if dot X and Y position are both even or both odd.
128 - 191	0 ! 1 --!-- 1 ! 1	Prints 3 of 4 dots. Leaves paper blank only if both X and Y are even.
192 - 255	1 ! 1 --!-- 1 ! 1	Prints a solid area of the proper color.

NOTE: The dithering pattern is performed separately for each of the three primary colors. When two colors are mixed, the dots are overlaid by mixing the primary colors.

2.11 EXPLORING THE SYSTEM

This section provides two programs for passing commands from the computer to the VX/PC interactively. The first program is intended to aid in exploring the 2-D command set. The second can be used to interactively manipulate a figure with 3-D commands Rotate, Scale, and Translate.

2.11.1 2-D Input Program

2-D Input accepts any 2-D commands and passes them to the processor. To enter more commands, press ENTER. When finished, type END.

```
10 DIM Y$(80)
20 OPEN "VECTRIX" FOR OUTPUT AS #1
30 INPUT "ENTER 2-D COMMAND ";Y$
40 IF Y$= "END" THEN CLOSE :STOP
50 PRINT #1, "SV"; Y$
60 INPUT A$ : PRINT #1, "SI"
70 GOTO 30
```

Some typical commands are:

E	c	- Erase screen to color c.
C	c	- Select c as next color.
M	x y	- Move drawing point to x,y.
L	x y	- Draw line to x,y.
A	r al a2	- Draw arc of r radius, from angle al to angle a2.
RF	w h	- Draw filled rectangle of width w and height h.
N	n	- Set pattern register to n.
\$	string	- Prints 'string'.
JA	al a2	- Set character slant, al, and rotation, a2.
JM	n	- Set character magnification to n.
OR		- Set OR color drawing mode.
RE		- Set REPLACE color drawing mode.
RC		- Set REPLACE COMPLEMENT color drawing mode.
XHN		- Turn crosshair on.
XHF		- Turn crosshair off.
XHP	x y	- Set crosshair position to X,Y
XHS	w	- Set crosshair size to w width and height.
RV		- Print PROM version number on screen.
XF		- Fill complex bounded area.
V	x1,x2,y1,y2	- Set viewport bounded by x1,x2,y1,y2.

Commands can be entered one at a time or several to a line. Be sure to enter spaces between numeric parameters, not commas, because BASICA uses commas to delimit input arguments. Most available commands, excluding only some of the 3-D commands and those needing many parameters, can be entered with this small program.

Pressing the enter key when finished will clear the screen and the program will prompt for another command. To exit, type END and press the enter key.

2.11.2 3-D Input Program

3-D INPUT is a useful program for understanding how to use the 3-D command set. Since more than one command can be entered on a line, a series of commands, such as Translate, Rotate, and Translate again can be entered. This program is intended for use with the Rotate, Scale, Translate, and Set Perspective Scale commands. The 2-D commands should not be used.

When this program has been typed in and run, a cube will appear on the screen. The front face of the cube is shaded yellow to help identify the cube's orientation after rotation or translation. Pressing ENTER will clear the screen and ask for 3-D input. Commands can now be entered to manipulate the cube in object space. After each figure is drawn, press ENTER to enter more 3-D commands or type END when finished. The commands are not cumulative because of the I command in line 40. Type END and press return when finished.

```
5 OPEN "VECTRIX" FOR OUTPUT AS #1
6 GOTO 40
10 INPUT "ENTER 3-D COMMANDS ";Y$
20 IF Y$ = "END" THEN STOP
40 PRINT #1, "SV RE K3 V96 575 0 479"           'Replace Mode,
50 PRINT #1, "I";Y$                             'Initialize matrix,
60 PRINT #1, "E0 C7"                             'Clear screen.
70 PRINT #1, "F63 4 -25 -25 50 -25 25 50 25 25 50 25 -25 50"
80 PRINT #1, "P4 -25 -25 100 -25 25 100 25 25 100 25 -25 100"
100 PRINT #1, "M -25 -25 50 L -25 -25 100"
110 PRINT #1, "M -25 25 50 L -25 25 100"         'Draw connecting lines
120 PRINT #1, "M 25 25 50 L 25 25 100"
130 PRINT #1, "M 25 -25 50 L 25 -25 100"
135 INPUT A$
140 PRINT #1, "SI"
160 GOTO 10
```

Some interesting command series are:

- RX300 - Rotates cube 30 degrees around X axis.
- RY300 - Rotates cube 30 degrees around Y axis.
- RZ300 - Rotates cube 30 degrees around Z axis.

- TZ -75 RX 300 TZ 75 - Translates cube to Z origin, rotates on X axis, and translates back to original position.
- TZ -75 RY 300 TZ 75 - Translates cube to Z origin, rotates on Y axis, and translates back to original Z position.

- SX 2 1 - Scales 2:1 along X axis.
- SX 1 2 - Scales 1:2 along X axis.

- SY 2 1 - Scales 2:1 along Y axis.
- SY 1 2 - Scales 1:2 along Y axis.

- SZ 2 1 - Scales 2:1 along Z axis.
- SZ 1 2 - Scales 1:2 along Z axis.

- SP 2 1 - Same as SZ 1 2.
- SP 1 2 - Same as SZ 2 1.

These are only a few of the many possibilities. If you are interested in 3-D, you can combine commands and vary argument values to get more or less perspective on these commands. For instance, by changing line #30 to set the default viewport, the cube will become a rectangular figure.

Chapter 3

PROGRAMMING THE VX/PC CARD SET

Contents	Page
3.1 GENERAL PROGRAMMING OVERVIEW.....	3-3
Using One Monitor vs. Two Monitors	
The MIDASDRV.COM Device Driver	
SV.EXE, SI.EXE, and RESET.EXE	
3.2 HIGH LEVEL LANGUAGE PROGRAMMING	3-6
Programming Languages	
Bi-Directional Commands	
Saving The Color Lookup Table	
Loading and Saving Images Faster	
Read and Store Pixel and RAM Programs	
3.3 LOW LEVEL LANGUAGE PROGRAMMING.....	3-11
Ports Used	
Assembly Language Code Fragments: PUTCHR and GETCHR	
Color Printer Data Format	
Source Files On The VX/PC Disk	

3.1 GENERAL PROGRAMMING OVERVIEW

3.1.1 Using One Monitor vs. Two Monitors

Either one or two monitors may be used in a system with the VX/PC card set. The first will always be a high quality analog RGB monitor. A second monitor, if connected, can only be an IBM monochrome monitor or equivalent, connected by a monochrome adapter card. This adapter cannot be a multidisplay board, containing both a monochrome and color graphics adapter.

***** WARNING *****

AN IBM COLOR CARD CANNOT BE USED IN A SYSTEM WITH VX/PC CARDS
DAMAGE MAY RESULT TO EITHER OR BOTH CARDS

A color adapter board can be nowhere in the system because the VX/PC board set has a built in IBM equivalent color graphics card. If another graphics board is in the system, bus contention will result. The IBM will not know which system to use, therefore neither will work properly. The same contention would result if two IBM graphics boards were installed.

With a two monitor system, any errors generated by a program will display on the monochrome screen, in the normal fashion.

Using a one monitor system for programming in a high level language means a single monitor is used for the IBM text commands and messages as well as the VX/PC graphics output. If an error occurs while the VX/PC is in Vectrix graphics mode, the IBM will print its error message to the IBM screen, which is hidden by the VX/PC display. The IBM must issue the SI command to return the system to IBM color graphics mode so the message or condition is visible.

Details provided here and in the REFERENCE GUIDE, Chapter 4, under the commands SI and SV, will assist the user with a single monitor system. As long as the IBM can issue the SI command, the system should be able to return to IBM mode. This can be done by creating ON ERROR routines in all high level programs. Additionally, selected function keys can be set to give the SI command if needed.

Both methods are included in this BASICA program fragment, in lines 10 and 1000 and lines 20 and 1010.

```
5 OPEN "VECTRIX" FOR OUTPUT AS #1
10 ON KEY(1) GOSUB 1000 : KEY(1) ON
20 ON ERROR GOTO 1010
30 PRINT #1,"SV"
.
.
.
PROGRAM CODE HERE...
.
.
.
1000 PRINT #1,"SI" : CLOSE : STOP
1010 PRINT #1,"SI" : PRINT ERR : STOP
```

A function key can also be set in immediate mode by typing:

```
KEY 10, "PRINT #1,"+CHR$(34)+"SI"+CHR$(34)
```

3.1.2 The MIDASDRV.COM Device Driver

The MIDASDRV.COM file is the device driver telling the IBM PC DOS to recognize the VX/PC card set. This driver opens a channel to the card set through which all communications are sent. Data can be passed in both directions using this channel.

The device driver should be installed by the CONFIG.SYS program whenever the system is powered on. If this has not been done, refer to Section 1.6.4 in Chapter 1, "Testing the Card Set and Monitor". This procedure will install the driver whenever the system is turned on.

A program must open the channel made available by the driver before the IBM system will communicate with the VX/PC. The card set is made available to the host by opening a file called "VECTRIX", using standard file opening syntax:

```
5 OPEN "VECTRIX" FOR OUTPUT AS #1
  or
5 OPEN "VECTRIX" FOR INPUT AS #1
```

The VX/PC is addressed by the host computer as a file; sending output to this file is similar sending output to a printer. Any command may be sent to the VX/PC by a statement similar to this:

```
PRINT #1, "SV G E0 RF 100 100 ..."
```

NOTE: LPRINT statements cannot be used in place of the PRINT #1 statement.

The source code for the device driver is provided on the VX/PC disk, in the file MIDASDRV.ASM.

3.1.3 SV.EXE, SI.EXE, RESET.EXE

The first two of these executable files are utility programs used to switch, from DOS, in and out of the Vectrix and IBM modes. The third performs a hard RESET of the VX/PC.

If the system is in Vectrix mode and DOS is available, SI.EXE. will return the system to IBM mode.

SV.EXE, added to a preliminary batch file, is often useful for opening the VX/PC immediately upon power up, and for switching to high resolution mode before loading an image. RESET.EXE resets the VX/PC. After this reset, the system will be in IBM mode. Any image resident in the VX/PC graphics memory will remain there.

3.2 HIGH LEVEL LANGUAGE PROGRAMMING

This section is provided to answer questions often asked by VX/PC users. It is intended to be a section that can grow as large and as detailed as necessary.

3.2.1 Programming Languages

All of the sample programs provided in this manual are written in either BASICA (IBM PC BASIC) or assembly language. However, the VX/PC card set can be programmed using any language.

When using BASIC, most programs can be compiled to dramatically increase drawing speed. Interpreted BASIC can be slow but is a convenient language for development work.

3.2.2 Bi-Directional Commands

The VX/PC command set includes twelve bi-directional commands. Each of these commands return one or more values to the host computer. The twelve commands are listed next, in alphabetical order.

Command	Name	Data Returned
HNP	Hardcopy Non-Dithered Print	Screen image data
HNR	Hardcopy Non-Dithered Reverse	Screen image data
HP	Hardcopy Dithered Print	Screen image data
HR	Hardcopy Dithered Reverse Print	Screen image data
HD	Return Drawing Point	X,Y position
RL	Return Light Pen Status	Pen pressed, X,Y position
RNP	Read Pixels (Encoded)	Pixel values
RNR	Read Graphics RAM (Encoded)	RAM contents
RP	Read Pixels	Pixel values
RQ	Return Color Lookup Table Value	Red, Green, Blue values
RR	Read Graphics RAM	RAM contents
XNR	Return Cross Hair Position	X,Y position

When data is to be returned to the host computer, the procedure will be similar to this example:

```
10 OPEN "VECTRIX" FOR OUTPUT AS #1
20 PRINT #1,"RD" : CLOSE #1
30 OPEN "R",1,"VECTRIX",6: FIELD 1,5 AS C$
40 GET 1
50 X$ = C$
60 GET 1
70 Y$ = C$
80 PRINT X$,Y$
90 CLOSE
```

The channel must be opened, the command sent, the channel closed, and each incoming data byte read and placed in appropriate variables.

NOTE: The data reads are done by GET statements, not INPUT statements.

3.2.3 Saving The Color Lookup Table

Another bi-directional command is RQ (Return Color Lookup Table Values). With this command, the contents of the color lookup table can be saved to a command file. When saving an image with an altered color lookup table, the contents of the color lookup table can be included in the image command file. This is necessary to restore the screen image properly. A program to do this is presented next. The current color lookup table is saved in the command file LUT.DAT.

```
10 OPEN "LUT.DAT" FOR OUTPUT AS #2
20 OPEN "VECTRIX" FOR OUTPUT AS #1
30 PRINT #1,"SV KF RQ 0 512 "
40 CLOSE 1
50 PRINT #2,"SV G KF Q0 512 "
60 OPEN "R",1,"VECTRIX",4 : FIELD 1,3 AS C$
70 FOR I = 1 TO 1536
80 GET #1,3
90 PRINT #2,C$
100 NEXT I
```

3.2.4 Loading and Saving Images Faster

One way to speed up data loading and saving is through use of the Flash video mode. This mode allows graphics memory to be updated during all non-blanking periods and will draw a screen about four times faster than Blank mode. However, using Flash video mode will cause short horizontal lines to appear on the screen during image loading and saving. Once the image is loaded or saved, these "flash" lines disappear.

Another method is using hex mode. Hex mode reduces the amount of data needed to build an image and speeds up transfer. The fastest image transfer can be done in this manner:

- 1) Draw the image.
- 2) Set hex mode and Flash mode.
- 3) Use the RNP or RNR command with the arguments in the correct hex format to return the screen image to the host. Data will be returned in hex format.
- 4) Save the image to a disk file.
- 5) To restore the image, set Hex and Flash mode, and transfer the file with WNP or WNR.

This method also removes the time it takes the computer to generate the original image. The image is stored with maximum compression and requires no interpretation by the host computer.

The command file created can be transferred from disk to the VX/PC using the DOS command COPY /B or just COPY. To use these commands, a WNP or WNR command MUST be included in the command file. The actual transfer is done directly by the DMA (Direct Memory Access) facilities onboard the VX/PC.

Some languages will not be able to use this method easily because certain hex codes are trapped, usually Ctrl Z and carriage return. This problem exists in BASICA. If characters that are valid hex data are trapped and not passed to the file created, the image cannot be restored to the screen correctly. For more information on this problem, see Chapter 2, Section 2.10.2.

The encoded RAM command (RNR) will read and encode graphics RAM and send the data to the host. This results in faster data transmission times, as there is less data to move. Reading graphics RAM directly also allows reading some bitplanes and ignoring others.

The fastest possible data transfer will occur when Flash mode (KF) is set, hex mode (HX) is set, and encoded RAM (RNR) images are being transferred.

3.2.5 Read and Store Pixel and RAM Program Examples

The next two programs will demonstrate methods of using the Run Length Encoding process to read and save screen images.

```
1  REM ** SAVE PIXELS **
10 OPEN "PIXIMAGE.DAT" FOR OUTPUT AS #2
20 OPEN "VECTRIX" FOR OUTPUT AS #1
30 PRINT #1,"SV KF RQ 0 512 " : CLOSE 1
50 PRINT #2,"SV G E0 KF Q0 512 "
60 OPEN "R",1,"VECTRIX",4 : FIELD 1,3 AS C$
70 FOR I = 1 TO 1536
80 GET #1,3
90 PRINT #2,C$
100 NEXT I : CLOSE 1
120 OPEN "VECTRIX" FOR OUTPUT AS #1
130 PRINT #1,"RNP0 "
140 CLOSE 1
150 PRINT #2,"WNP0 "
160 OPEN "R",1,"VECTRIX",6 : FIELD 1,5 AS C$
170 GET #1,5
180 IF C$="00000" THEN PRINT "DONE" : GOTO 210
190 PRINT #2,C$
200 GOTO 170
210 INPUT A$ : PRINT #1,"SI" : CLOSE
```

NOTES: After execution, the PIXIMAGE.DAT command file will contain the generated commands. Lines 50, 90, 150, and 190 write data to this file.

In lines 60 and 160, the VX/PC is opened for random access.

In the OPEN commands, record length is 1 byte more than the field length. This provides for a carriage return.

Lines 50 through 100 read the current color lookup table, and save it in the command file.

Line 170 reads the screen data.

Line 180 checks for the end of screen data.

Line 190 saves the data read in line 170.

Line 210 is included for single monitor systems.

Gets are used instead of input statements.

To restore the image, type: COPY PIXIMAGE.DAT VECTRIX /B

```

1  REM ** SAVE GRAPHICS RAM **
10 OPEN "RAMIMAGE.DAT" FOR OUTPUT AS #2
20 OPEN "VECTRIX" FOR OUTPUT AS #1
30 PRINT #1,"SV KF RQ 0 512 "
40 CLOSE 1
50 PRINT #2,"SV G E0 KF Q0 512 "
60 OPEN "R",1,"VECTRIX",4 : FIELD 1,3 AS C$
70 FOR I = 1 TO 1536
80 GET #1,3
90 PRINT #2,C$
100 NEXT I
110 CLOSE 1
120 OPEN "VECTRIX" FOR OUTPUT AS #1
130 PRINT #1,"SV" : CLOSE 1
150 FOR I = 1 TO 9
160 S=0
170 OPEN "VECTRIX" FOR OUTPUT AS #1
180 PRINT #1,"RNR"+STR$(I);" 0"
190 CLOSE 1
200 PRINT #2,"WNR"+STR$(I);" 0"
210 OPEN "R",1,"VECTRIX",6 : FIELD 1,5 AS C$
220 GET #1,5
230 IF S=0 THEN 250 ELSE 240
240 IF C$="00000" THEN PRINT "DONE" : GOTO 280
250 IF C$="00000" THEN S=1 ELSE S=0
260 PRINT #2,C$
270 GOTO 220
280 CLOSE 1 : NEXT I
300 OPEN "VECTRIX" FOR OUTPUT AS #1
310 INPUT A$ : PRINT #1,"SI"

```

NOTES: After execution, the RAMIMAGE.DAT command file will contain the generated commands. Lines 50, 90, 180, and 260 write data to this file.

In lines 60 and 210, the VX/PC is opened for random access. In the OPEN commands, record length is 1 byte more than the field length. This provides for a carriage return.

Lines 50 through 110 read the current color lookup table and save it in the command file.

Line 220 reads the screen data.

Line 230 through 250 check for the end of screen data. This is signaled by two sets of zeroes.

Line 260 saves the data read in line 170.

Lines 300 and 310 are included for single monitor systems. A complex image may take longer than fifteen minutes to save using this program.

Gets are used instead of input statements.

To restore the image, type: COPY RAMIMAGE.DAT VECTRIX /B

3.3 LOW LEVEL LANGUAGE PROGRAMMING

System developers can use the hardware ports on the VX/PC, the assembly language drivers, and printer program for any need they may have. Vectrix provides these routines as a base from which system programmers can write routines. However, Vectrix cannot be responsible for the functionality of the routines if changed.

3.3.1 Ports Used

The three ports used are listed next.

PORT (in HEX)	USED FOR
03DD	Status from VX/PC
03DE	Write to VX/PC
03DF	Read from VX/PC (also RESET VX/PC cards)

NOTE: A write operation to the read port, 03DF, will reset the VX/PC. This is what RESET.EXE does.

The status port, location 3DD, uses these three bits for these flags:

- BIT 7 - 0 = VX/PC busy
1 = VX/PC ready

- BIT 6 - 0 = VX/PC data available
1 = No VX/PC data available

- BIT 5 - 0 = VX/PC in IBM mode
1 = VX/PC in Vectrix mode

3.3.2 Assembly Language Code Routines: PUTCHR and GETCHR

These two routines are the subroutines used to move data to and from the VX/PC. They are included here and on the VX/PC disk. A single character at a time is moved.

```
;*****  
;  
;          SUBROUTINE PUTCHR  
;  
;    Sends a character in AL to the VX/PC card.  
;  
;*****  
  
PUTCHR:  PUSH    DX  
         PUSH    AX  
         MOV     DX,03DDH      ;Status port  
PUTCH2:  IN      AL,DX  
         TEST    AL,80H      ;Check ready but  
         JZ      PUTCH2      ;JMP if busy  
  
         INC     DX          ;Point to write port  
         POP     AX  
         OUT    DX,AL        ;Send it  
  
         POP     DX  
         RET
```

```

;*****
;
;          SUBROUTINE GETCHR
;
;    Gets a character in AL from the VX/PC card.
;
;    DL is 0 if character in AL is valid
;    DL is 1 if timeout occurs
;*****

```

```

GETCHR:  MOV     CX,0FFFFH

        MOV     DX,03DDH      ;Status port
GETCH1:  IN      AL,DX
        TEST    AL,40H       ;IBF?
        JZ     GETCH2       ;JMP if buffer not full
        LOOP   GETCH1       ;Loop till CX=0
        MOV     DL,1        ;Set flag
        RET

GETCH2:  INC     DX          ;Point to input port
        INC     DX
        IN      AL,DX       ;Get the byte
        SUB     DL,DL       ;Set flag

        RET

```

3.3.3 Color Printer Data Format

When the HP, HR, HNP, or HNR commands are used, the VX/PC returns screen image data to the host computer. Unlike MPRINT and related programs, the data is not passed from the host to the color printer. The format of the data returned is described next. 116,654 bytes are transmitted.

```

1 set of:      0DH, 0AH (Carriage return, Line feed)
480 lines of:  1BH, 43H, 50H, 240 bytes of data (ESC C P)
12 bytes of:   0D      (Carriage return)

```

When using the dithered commands HP and HR, the data sent to the host will have passed through the dithering algorithm.

These data can be passed directly to the standard printer, the Radio Shack CGP-220 Inkjet Printer. When other printers are to be used, the required data format may be different.

The actual screen data is the 480 lines of 240 bytes each. This breaks down to 80 bytes (640 bits) per color per row of pixels. Each bit represents one pixel. With three colors (red, green, and blue) per pixel, 240 bytes are transferred for each of the 480 pixel rows.

Because the screen is actually 672 pixels wide, sixteen pixels on either edge of the screen are not transmitted.

3.3.4 Source Files On The VX/PC Disk

The source code for the system programs used with the VX/PC are provided on the VX/PC disk. The source files and their uses are listed next:

Source File	Purpose
CODEFRAG.ASM	GETCHR, PUTCHR Routines
MIDASDRV.ASM	The VX/PC Device Driver
MPRINT.ASM	One of the Printer Drivers
TODMA.ASM	Use DMA Transfers
FROMDMA.ASM	Use DMA Transfers
TESTTO.ASM	DMA's Command Files to VX/PC
TESTFROM.ASM	DMA's Image to Host

Chapter 4

REFERENCE GUIDE

Contents	Page
4.1 HOW TO USE THIS CHAPTER.....	4-3
4.2 COMMAND ARGUMENTS.....	4-8
ASCII Mode; Hex Mode	
4.3 COMMAND AND DATA DELIMITERS.....	4-11
4.4 COMMANDS:	
\$ Display Character String	4-12
A Arc or Circle	4-14
B Bitplane Write Mask	4-15
C Set Color	4-16
D Dot	4-18
E Erase Screen	4-19
F Filled Convex Polygon	4-20
G Go Warmstart	4-22
GØ Go Warmstart	4-23
HNP Hardcopy Non-Dithered Print	4-24
HNR Hardcopy Non-Dithered Reverse Print	4-25
HP Hardcopy Dithered Print	4-26
HR Hardcopy Dithered Reverse Print	4-27
HX Select Hex Mode	4-28
I Initialize Transformation Matrix	4-29
JA Adjust Character Angle	4-30
JD Design New Character	4-32
JM Change Character Magnification	4-34
JN Display Default Character Set	4-35
JR Set Rectangular Fill Pattern	4-36
JS Set Character Spacing	4-38
K2 Select 2-D Coordinates	4-40
K3 Select 3-D Coordinates	4-41
KA Select Absolute Coordinates	4-42
KB Select Blank Video Mode	4-43
KD Select ASCII Decimal Mode	4-44
KF Select Flash Video Mode	4-45
KR Select Relative Coordinates	4-46
L Line	4-47
M Move	4-48

4.4 COMMANDS: (continued)

N	Set Pattern Register	4-49
OA	Originate Arc	4-50
OF	Turn Video Off	4-51
ON	Turn Video On	4-52
OR	OR Mode	4-53
P	Polygon	4-54
PAN	PAN Video Image	4-55
Q	Define Color Lookup Table Value	4-56
R	Rotation	4-58
RA	Replace All Mode	4-60
RC	Replace Complement Mode	4-61
RD	Return Drawing Point	4-62
RE	Replace Mode	4-64
RF	Rectangle Fill	4-65
RL	Return Light Pen Position	4-67
RNP	Read Pixels (Encoded)	4-69
RNR	Read Graphics RAM (Encoded)	4-72
RP	Read Pixels	4-74
RQ	Return Color Lookup Table Value	4-77
RR	Read Graphics RAM	4-78
RV	Return Version Number	4-80
S	Scale	4-81
SI	Select IBM Mode	4-83
SP	Set Perspective Scale	4-85
SQ	Return Color Lookup Table to Default	4-89
SV	Select Vectrix Mode	4-90
T	Translation	4-91
TB	Transfer Block	4-94
U	Upload Code	4-96
V	Define Viewport	4-97
WA	Wedge Arc	4-99
WB	Wedge Arc, Boundary Filled	4-100
WF	Wait Frames	4-101
WNP	Write Pixels (Encoded)	4-102
WNR	Write Graphics RAM (Encoded)	4-105
WP	Write Pixels	4-107
WR	Write Graphics RAM	4-109
XB	Complex Boundary Fill	4-111
XF	Complex Flood Fill	4-112
XHC	Set Crosshair Position To CDP	4-113
XHF	Crosshair Off	4-114
XHN	Crosshair On	4-115
XHP	Set Crosshair Position	4-116
XHR	Return Crosshair Position	4-117
XHS	Set Crosshair Size	4-118
Z	Zoom Video Image	4-119

4.1 HOW TO USE THIS CHAPTER

This Reference Guide is intended to provide a quick method of finding detailed information on every command available with the VX/PC Card Set.

The first part of this section is an overview of the chapter where general principles relating to all commands are introduced. These principles include logical command groupings, reference section page format, command argument structure, byte and word argument values, and the use of delimiters.

In the second part, a detailed specification for each command is presented. The commands are presented in alphabetical order with each command starting on a new page. Each command contains these elements:

- 1) The heading at the top of the page;
- 2) A short description of the command's purpose;
- 3) The format of the command and use of its arguments;
- 4) Remarks describing uses and any limitations of the command;
- 5) Examples using the command;
- 6) Notes relating to specialized uses of the command, and any germane cautionary messages.

The heading at the top gives the command mnemonic, the descriptive name of the command, and the command group to which the command belongs. The twelve command groups are listed next with descriptions.

Operating Modes

- the available operating modes:

- Absolute Coordinates
- Relative Coordinates
- Blank Drawing
- Flash Drawing
- 2-D Coordinates
- 3-D Coordinates
- ASCII Transmission
- Hex Transmission
- Select Vectrix Mode
- Select IBM Mode

Graphics Primitives

- basic graphic commands:

- Arc
- Complex Boundary Fill
- Complex Flood Fill
- Dot
- Filled Polygon
- Line
- Move
- Originate Arc
- Polygon
- Rectangle Fill
- Return Drawing Point
- Return Light Pen Position
- Set Pattern Register
- Set Rectangle Fill Pattern
- Wedge Arc
- Wedge Arc Boundary Filled

Text Primitives

- basic text commands:

- Adjust Character Angle
- Change Character Magnification
- Design New Character
- Display Character String
- Display Default Character Set
- Set Character Spacing

Color Manipulations

- color and bitplane commands:

- Bitplane Write Mask
- Define Color Lookup Table Value
- Erase Screen
- Return Color Lookup Table Value
- Return Color Lookup Table To Default
- Set Color

- Color Drawing Modes - modes controlling color mixing when colors are overlaid:
- OR
 - Replace
 - Replace All
 - Replace Complement
- 3-D Commands - commands to manipulate images in 3-D and 2-D space:
- Initialize Transformation Matrix
 - Rotate
 - Scale
 - Set Perspective Scale
 - Translate
 - Viewport
- Pixel Commands - commands for image manipulation on the pixel level:
- Read Pixels
 - Read Pixels (Encoded)
 - Write Pixels
 - Write Pixels (Encoded)
- RAM Commands - commands for image manipulation on the graphics RAM level:
- Read Graphics RAM
 - Read Graphics RAM (Encoded)
 - Write Graphics RAM
 - Write Graphics RAM (Encoded)
- Crosshair Commands - commands for setting and changing the crosshair:
- Crosshair Off
 - Crosshair On
 - Return Crosshair Position
 - Set Crosshair Position
 - Set Crosshair Size

Video Commands

- commands to modify the monitor parameters and the video image:

- PAN Video Image
- Turn Video Off
- Turn Video On
- Zoom Video Image

Color Printing

- commands to return screen data to the host computer:

- Hardcopy Non-Dithered Print
- Hardcopy Non-Dithered Reverse Print
- Hardcopy Print
- Hardcopy Reverse

Miscellaneous

- special use commands:

- Return Version Number
- Upload Code
- Wait Frames

In addition, three DMA commands are described in Appendix B.

DMA Commands

- the Direct Memory Access commands:

- Set Non-DMA Mode
- Set DMA Read Mode
- Set DMA Write Mode

The remaining sections for each command are explained below.

PURPOSE: explains the most common use of the command.

FORMAT: shows the command with all arguments, though some commands need not use all arguments to be valid. An example is any command that can be used in both 2-D and 3-D mode. Though three arguments are used in 3-D mode (e.g. X, Y, and Z coordinates), only two (X and Y coordinates) are used in 2-D mode.

Below each format description is a section explaining the meaning and range of the format arguments. Included is the argument type (whether it is a byte or a word, signed or unsigned, etc.) , the maximum and minimum range of the argument, and the units in which the argument is measured.

REMARKS: details use of the command, and describes any special points germane to the command. The details needed to use the command in the real world is provided in this section.

Any sections of text that are blocked smaller than normal in this section refer to special points concerning the command.

EXAMPLE(S): gives at least one example for each command with arguments. Commands without arguments may or may not have examples. If any trouble using the command examples is experienced, issue the G (Go Warmstart) command and try the example again.

NOTE(S): gives any special cases that must be understood to fully use the command. Included are command limitations and where to find more information on the command.

4.2 COMMAND ARGUMENTS

This section describes the formats for numeric command arguments. A synopsis is presented in Illustration 4.1.

Command arguments are sent to VX/PC as either ASCII or hex values, and as byte or word values. Byte values are 8 bits, word values are 16 bits (2 bytes).

Word values have two subgroups: one for screen coordinates and another for counts, as in the number of vertices in a polygon. Coordinates can range from -32767 to +32767. Count values range from 0 to 65535.

Acceptable values can be greater than what may actually appear on the screen. For example, in 2-D mode, the line drawing command (L x, y) can accept values from -32767 to +32767 for the X or Y arguments. This does not mean that all these points will be reproduced on the screen. The resolution of the VX/PC system is 672 (X) by 480 (Y). Any points not falling within these boundaries will be clipped. If a line is specified from X=0, Y=0 to X=10000, Y=10000, only the portion that falls within the boundaries of X=0, Y=0 to X=671, Y=479 are displayed.

If an argument's possible range is larger than 256, it is a word value. If the argument has a range of exactly 256 values (0-255), it is a single byte value.

The command arguments shown give the ASCII format for each argument.

4.2.1 ASCII Mode

In ASCII mode, byte values are unsigned numbers from 0 to 255. Coordinate values are given as ASCII decimal numbers from -32767 to +32767, though the plus sign can be dropped. Count values are specified as ASCII values from 0 to 65535.

When byte values are returned to the host computer with commands like RQ (Return Color Lookup Table Value) in ASCII mode, they will be a fixed length string of three ASCII digits, from "000" to "255", followed by a carriage return with no linefeed.

When word values are returned to the host computer with commands like RP (Read Pixels) in ASCII mode, they will be a fixed length string of five ASCII digits, from "00000" to "65535", followed by a carriage return but no linefeed.

However, one exception to this rule is the RD command (Return Drawing Point). Positive coordinates returned by this command are represented as 0 through 32767 and negative coordinates are represented as the returned value minus 65536. A returned value of 65535 is equal to 65535 minus 65536 or -1. Thus the possible range is +32767 to -32768.

4.2.2 Hex Mode

Since there are no convenient representations of byte values other than binary format (strings of 0's and 1's) or hex format (00-FF), the term "byte value" is used here to mean the actual value to be transmitted. Hexadecimal representations of the byte values are used to quantify the values in a convenient manner. However, values are not passed or returned as hex digits.

In hex mode, byte values are represented by 00H to FFH. Coordinate values can be from 0000 to 7FFF (0 to 32767) or FFFF to 8000 (-1 to -32768). Counts and other non-coordinate data can have unsigned values from 0000 to FFFF. The low byte (LSB) of each numeric argument should be sent first, then the high byte (MSB). To send a value represented by 7FFF, first send FF, then 7F.

When byte values are returned to the host computer with commands like RQ while in hex mode, the returned byte values are represented by hex numbers 00 to FF.

When word values are returned to the host computer with commands like RP while in hex mode, the returned binary values are represented by hex numbers from 0000 to FFFF. The value 0F00 returns with the low byte (LSB) first, as 000F. No carriage return or linefeed is returned.

Illustration 4.1 How To Send BYTE And WORD Values To The VX/PC

If in ...	and Value is a ...	Send ...
ASCII Mode	Byte	"0" to "255"
	Coordinate	"-32768" to "+32767"
	Count	"0" to "65535"

Hex Mode	Byte	00 to FF (0...255)
	Coordinate	00 00 to 7F FF (0...32767) or FF FF to 80 00 (-1...-32768)
	Count	00 00 to FF FF (0...65535)
=====		

Illustration 4.2 How The VX/PC Returns Values To The Host Computer

If in ...	and Value is a ...	Returned Number is ...
ASCII Mode	Byte	"000" to "255"
	Word	"00000" to "65535"

Hex Mode	Byte	00 to FF (0...255)
	Word	00 00 to FF FF (0...65535)

* The RD command is an exception.		
ASCII mode:	Word	"0" to "32767" or "32768" to "65535" (-1...-32768)
Hex mode:	Word	00 00 to 7F FF (0...+32767) or FF FF to 80 00 (-1...-32768)
=====		

4.3 COMMAND AND DATA DELIMITERS

Delimiters are separators between numeric values. Valid delimiters are commas, spaces, or carriage returns with or without linefeeds. When a command is entered in ASCII mode, one or more delimiters must be put between numeric arguments. An example using the L (Line) command is given next.

```

                                ASCII Mode
-----
L x,y,z   or   L 50,100,50  works.
L x y z   or   L 50 100 50  works.
Lx y,z    or   L50 100, 50  works.
Lxyz      or   L5010050     Does not work.
-----
```

In hex mode, delimiters cannot be used. In hex mode, every byte counts and unpredictable results will occur if byte counts are in any way incorrect.

In either mode, multiple commands can be put on a line line to code space. However, lines should not exceed the maximum line length the operating system or language allows. If this happens, carriage returns and possibly linefeeds may be inserted when data is sent to the VX/PC. This can produce unwanted results.

\$
DISPLAY CHARACTER STRING
TEXT PRIMITIVE

PURPOSE: Used for displaying text on the monitor.

FORMAT: \$ character-string (carriage return)

character-string = any group of ASCII characters

REMARKS: This command allows the addition of text and labels to images. The "character string" is displayed in the current drawing color. The upper left hand corner of the first character is placed at the current drawing point.

The printable ASCII characters are ASCII 32 (space) through ASCII 126. This includes upper and lower case letters, numbers 0 through 9, punctuation and special characters.

Non-ASCII characters and non-printing ASCII characters embedded in the character string will be displayed as spaces.

Most high level languages allowing strings also produce an implied carriage return to end them. In these languages, no terminating carriage return need be added by the user.

However, when assembly language is used, care must be taken to use a carriage return (decimal 13 or hex 0D), letting the VX/PC know the character string is finished.

To cause an actual carriage return on the screen, send two carriage returns, with or without linefeeds, in place of the terminating carriage return.

When a carriage return is found, the current drawing point is moved to the upper left hand corner of the next position into which a character would be drawn. This position is on the current character line if the string is terminated by one carriage return, and on the next character line if the text is terminated by two carriage returns.

\$ (continued)

Characters are not clipped at screen or viewport boundaries. If a character is printed beyond the edge of the screen, it wraps around and prints one scan line down on the opposite screen edge.

EXAMPLES: PRINT #1, "\$This is the text to be displayed"

```
SS = "This is the text to be displayed"  
PRINT #1, "$";SS
```

NOTES: Since there is no command to end a string, text to be printed should be the last command on the line. Otherwise, any commands following will be assumed to be text.

Characters need not start on particular boundaries. They can be printed at any X,Y location on the screen.

PURPOSE: Used to draw arcs and circles centered on the current drawing point.

FORMAT: A radius, start angle, arc angle

radius = unsigned word 0...65535 pixels
 start angle = signed word +-3600 degree tenths
 arc angle = signed word +-3600 degree tenths

REMARKS: The A command draws arcs and circles around the current drawing point, using the current drawing color. The radius is in pixels. The arc starts at "start angle", sweeping through "arc angle" degrees. Both angles are specified in tenths of a degree: 90 degrees would be given as 900 for either angle. Positive values indicate a counterclockwise drawing direction; negative angles draw clockwise. The current drawing point remains at the center of the arc upon completion. The "start angle" is measured from the current X and Y position.

EXAMPLES: A 100 900 900 draws a ninety degree arc of 100 pixel radius. The CDP is the arc's center point.

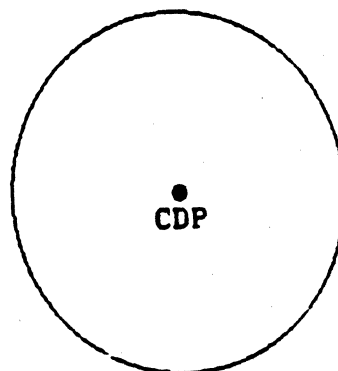
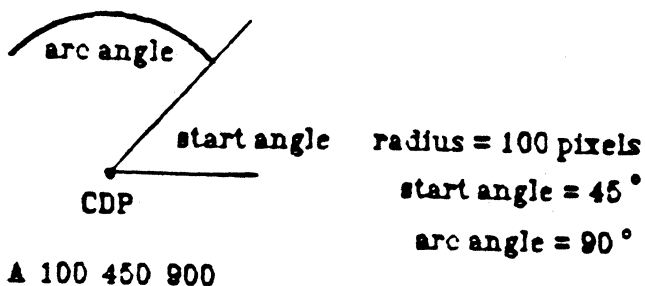
A 100 0 3600 draws a circle with 100 pixel radius.

NOTES: To draw a circle, set "arc angle" to 3600.

Arcs and circles cannot be rotated, scaled, or translated in 3-D mode.

In RC (Replace Complement) color drawing mode, gaps may be produced in the arc.

Illustration 4.3 Using The ARC Command



A 100 0 3600
 radius = 100 pixels
 start angle = 0°
 arc angle = 360°

B
BITPLANE WRITE MASK
COLOR MANIPULATION

PURPOSE: To set bitplanes to accept or not accept update.

FORMAT: B bitplane mask

bitplane mask = unsigned word 0...511

REMARKS: Use of this command provides access to individual or grouped bitplanes, making animation and other special effects possible.

Bitplanes are groups of bits organized as one bit for each of the 672 x 480 pixels on the screen. The VX/PC has nine bitplanes, which store color and image data for each pixel, allowing $2^9=512$ simultaneous colors on the screen.

Animation effects can be created by storing several images, each in different planes (or groups of planes) using the B command - and switching between them by changing the color lookup table with the Q command (Define Color Lookup Table Value). Bitplanes can be used to store text to be displayed on the screen independently of graphics images.

The default value for the bitplane mask is 511, all available bits are set to one, enabling all planes to be updated. When the RESET program is run or when the command G or G0 is given, the bitplane mask returns to the default value.

The VX/PC determines if transmissions are in ASCII or HEX mode from the format of the arguments in the B, C, E, and Q commands. Command format is changed accordingly.

EXAMPLES: B1 Selects only bitplane 1

B3 Selects bitplanes 1 and 2

B56 E0 Selects bitplanes 4, 5, and 6, then sets all bits in planes 4,5, and 6 to zeroes.

C
SET COLOR
COLOR MANIPULATION

PURPOSE: Select a color for the next drawing operation.

FORMAT: C color

color = unsigned word 0...511

REMARKS: The C command sets the current drawing color. All drawing will use this color until the next C command is used to change the color.

If a color value larger than 511 is specified, a modulus operation will be performed to determine the color. For example, "C512" will actually select color table index entry #0.

Some commonly used color numbers are:

Color	Default Index Entry
Black	0
Red	7
Green	56
Yellow	63
Blue	448
Magenta	455
Cyan	504
White	511

It is important to note that the colors refer to entries in the 512 entry color lookup table. Each entry has a default value that is a specific mix of red, green, and blue. These values can be changed at will, so a color number can refer to any of 4096 possible mixtures of the three basic colors.

The VX/PC determines if transmissions are in ASCII or HEX mode from the format of the arguments in the B, C, E, and Q commands. Command format is changed accordingly.

C (continued)

EXAMPLES: C511 Select color table entry #511, white in
the default table.

C7 Select color table entry #7, which in the
default table is red.

NOTE: For information on changing the default entries,
see the Q (Define Color Lookup Table Value)
command in this section.

PURPOSE: Draws a dot, at the current drawing point, using the current drawing color.

FORMAT: D x, y, z

x = signed word +- 32767 pixels
y = signed word +- 32767 pixels
z = signed word +- 32767 pixels

REMARKS: The D command draws a dot at screen location X,Y or if in 3-D mode at location X,Y,Z, and resets the current drawing point to these coordinates.

This command is often useful when many lines must begin at the same point. A series of commands to radiate lines from a single point is given below.

```
D 100 100  
L 200 200  
D 100 100  
L 200 220  
D 100 100  
L 200 240
```

The M command (Move) will also achieve this effect.

EXAMPLES: D 100 100 draws one dot (pixel) in the current drawing color at X = 100 and Y = 100 in 2-D mode.

D 50 75 1000 draws one dot in the current drawing color at X = 50, Y = 75, and Z = 1000 in 3-D mode.

E
ERASE SCREEN
COLOR MANIPULATION

PURPOSE: Clears the entire screen to a particular color.

FORMAT: E color

color = unsigned word 0...511

This command:

- 1) Erases the screen. Any image is overlaid.
- 2) Sets the entire screen's color to the color number following the E.
- 3) Sets the current drawing point to X=0, Y=0 in 2-D mode or X=0, Y=0, Z=0 in 3-D mode.

The VX/PC determines if transmissions are in ASCII or HEX mode from the format of the arguments in the B, C, E, and Q commands. Command format is changed accordingly.

EXAMPLES: E 448 - Changes entire screen to the color indicated in color table entry #448, blue in the default table.

E 0 - Clears the screen to the color in color table index #0, which is black in the default color lookup table.

NOTES: The ERASE command performs in FLASH mode when the bitplane mask register is all 1's. Otherwise ERASE is performed in the current video mode, FLASH or BLANK.

E clears the entire screen to the specified color regardless of any viewport setting.

Any bitplanes not enabled by the B command (Bitplane Write Mask) command will not be erased.

F
FILLED POLYGON
GRAPHIC PRIMITIVE

PURPOSE: Draws an "n" sided polygon and fills it with the specified fill color.

FORMAT: F fill color, count, x1, y1, z1, x2, y2, z2, ...

fill color	= unsigned word	0...511
count	= unsigned word	3...256 vertices
x1,..	= signed word	+ - 32767 pixels
y1,..	= signed word	+ - 32767 pixels
z1,..	= signed word	+ - 32767 pixels

REMARKS: The F command draws a polygon with "count" number of vertices at the specified X,Y coordinates, or if in 3-D mode, at the X,Y,Z coordinates. Vertices may be specified either in clockwise or counterclockwise order. The VX/PC will automatically fill the interior with the color specified by "fill color".

The border of the polygon is drawn first, using the current drawing color. Then the interior of the polygon is filled using the "fill color". The current drawing color is not changed.

The contents of the pattern register affects the interior and the perimeter of the shape. See the N command (Set Pattern Register) for more details.

The final side of the polygon need not be specified because the VX/PC assumes the first vertex is also the last. After this command executes, the current drawing point becomes the first vertex. This applies in both absolute and relative mode.

If there are gaps in the borders of the polygon when it is filled, the fill color will leak into the surrounding area. These gaps can occur when drawing over a background of two or more colors while in the RC color drawing mode. If leakage happens, it can be stopped by sending a Ctrl-E from the host computer to the VX/PC. From a BASIC program, the same can be done using CHR\$(5).

F (continued)

EXAMPLES: C 511 KA F 5 3 100 100 150 150 200 100

This series draws and fills a triangle with vertices at the coordinate pairs 100 100, 150 150, and 200 100. The perimeter is white (C 511) and the fill color is 5.

C 511 KR F 5 3 100 100 50 50 50 -50

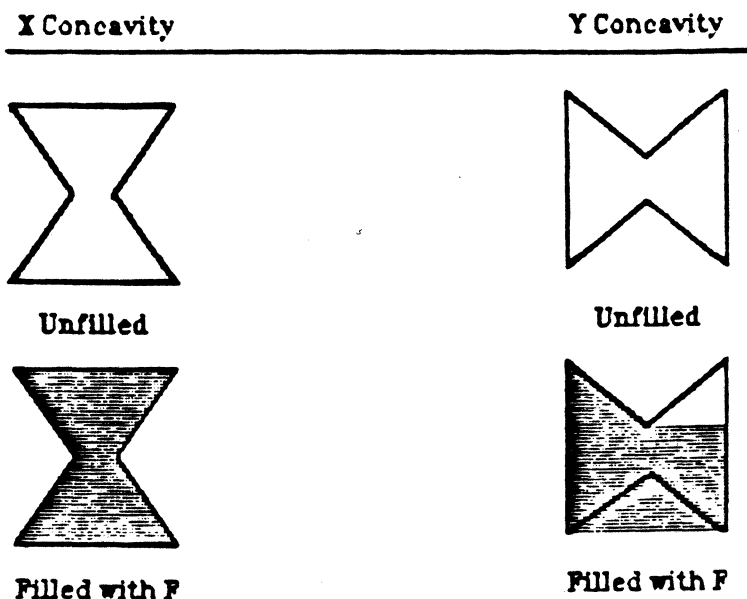
This draws the same triangle at the same location in Relative mode.

KA K3 F 5 3 100 100 100 150 150 200 200 100 100

This command series draws and fills a triangle in 3-D absolute coordinates.

NOTE: The VX/PC polygon fill algorithm is a "Y scan line convex polygon fill" and does not fill polygons with concave (squeezed in) areas on the top or bottom of the polygon. The algorithm will fill concavities in the X direction. When concavities along the Y axis need to be filled, use the P command to draw the polygon, then fill the polygon using the XF command (Complex Boundary Fill) or XB command (Complex Flood Fill).

Illustration 4.4
Results Of Y Scan Line Convex Polygon Fill Algorithm



PURPOSE: Resets most operating parameters.

FORMAT: G (no arguments)

REMARKS: This command provides a way to reset all modes and many other conditions to a default state. This process is called a warmstart initialization. The conditions reset are:

- Absolute coordinates
- 2-D coordinates
- Blank video mode
- ASCII decimal transmission
- Current drawing point to 0,0
- OR color drawing mode
- Color Lookup Table to the default
- Default character set
- Character magnification to 1
- Character angle to 0
- Character spacing to 7 (horizontal),
0 (vertical), and 12 (line)
- Rectangular Fill Pattern to all ones
- Viewport to full screen
- 3-D transformation matrix to identity
- Image pan to zero
- Bit Plane Mask Register to all ones
- Pattern Register to all ones (a solid line)

Other than resetting the color lookup table, the G command will not affect the screen display.

GØ
GO WARMSTART

PURPOSE: Resets parameters similar to the G command.

FORMAT: GØ (no arguments)

REMARKS: GØ performs a warmstart identical to the G command except the color look up table is not reset to the default table.

HNP
HARDCOPY NON-DITHERED PRINT
COLOR PRINTING COMMAND

PURPOSE: Used to transfer the screen image to the host computer.

FORMAT: HNP (no arguments)

REMARKS: The HNP command returns the current contents of the VX/PC's graphic RAM, representing the screen image, to the host computer. These data can then be sent to a color printer in a separate operation.

Data is returned in the format required by the Radio Shack CGP-220 Color Inkjet Printer, the Canon Inkjet, and the Quadram Quadjet printer. Other printers may require different printer control codes or different data formatting.

The data is transferred to the host in the eight color, non-dithered format. Screen colors are matched to the closest counterpart color that can be produced from the red, green, and blue color components.

For information on data format and the use of the HNP command, see Chapter 3, Section 3.3.3, "Color Printer Data Format".

For more information on color printing, refer to Chapter 2, Section 2.10, "Color Printing".

NOTE: The MPRINT and related programs provide the functions of returning data to the host and routing the data to the printer. These programs are provided on the VX/PC disk. See also Chapter 2, Section 2.10.1, "The Color Printing Programs".

HNR
HARDCOPY NON-DITHERED REVERSE PRINT
COLOR PRINTING COMMAND

PURPOSE: Used to transfer the screen image to the host computer.

FORMAT: HNR (no arguments)

REMARKS: The HNP command returns the current contents of the VX/PC's graphic RAM, representing the screen image, to the host computer. These data can then be sent to a color printer in a separate operation.

Before transfer, some of the image data is reversed - white areas are sent as black, and black areas are sent as white. Others colors are not affected.

Data is returned in the format required by the Radio Shack CGP-220 Color Inkjet Printer, the Canon Inkjet, and the Quadram Quadjet printer. Others printers may require different printer control codes or different data formatting.

The data is transferred to the host in the eight color, non-dithered format. Screen colors are matched to the closest counterpart color that can be produced from the red, green, and blue color components.

For information on data format and the use of the HNR command, see Chapter 3, Section 3.3.3, "Color Printer Data Format".

For more information on color printing, refer to Chapter 2, Section 2.10, "Color Printing".

NOTE: The MPRINT and related programs provide the functions of returning data to the host and routing the data to the printer. These programs are provided on the VX/PC disk. See also Chapter 2, Section 2.10.1, "The Color Printing Programs".

HP
HARDCOPY DITHERED PRINT
COLOR PRINTING COMMAND

PURPOSE: Used to transfer the screen image to the host computer.

FORMAT: HP (no arguments)

REMARKS: The HP command returns the current contents of the VX/PC's graphic RAM, representing the screen image, to the host computer. These data can then be sent to a color printer in a separate operation.

Data is returned in the format required by the Radio Shack CGP-220 Color Inkjet Printer, the Canon Inkjet, and the Quadram Quadjet printer. Others printers may require different printer control codes or different data formatting.

The data is transferred to the host in the dithered, 125 color format. This format uses color dot mixing to achieve apparent shadings of colors.

For information on the data format and the use of the HP command, see Chapter 3, Section 3.3.3, "Color Printer Data Format".

For more information on color printing and dithering, refer to Chapter 2, Section 2.10, "Color Printing".

NOTE: The MPRINT and related programs provide the functions of returning data to the host and routing the data to the printer. These programs are provided on the VX/PC disk. See also Chapter 2, Section 2.10.1, "The Color Printing Programs".

HR
HARDCOPY DITHERED REVERSE PRINT
COLOR PRINTING COMMAND

PURPOSE: Used to transfer the screen image to the host computer.

FORMAT: HR (no arguments)

REMARKS: The HR command returns the current contents of the VX/PC's graphic RAM, representing the screen image, to the host computer. These data can then be sent to a color printer in a separate operation.

Before transfer, some of the image data is reversed - white areas are sent as black, and black areas are sent as white. Others colors are not affected.

Data is returned in the format required by the Radio Shack CGP-220 Color Inkjet Printer, the Canon Inkjet, and the Quadram Quadjet printer. Others printers may require different printer control codes or different data formatting.

The data is transferred to the host in the dithered, 125 color format. This format uses color dot mixing to achieve apparent shadings of colors.

For information on data format and the use of the HP command, see Chapter 3, Section 3.3.3, "Color Printer Data Format".

For more information on color printing and dithering, refer to Chapter 2, Section 2.10, "Color Printing".

NOTE: The MPRINT and related programs provide both the functions of returning data to the host and routing the data to the printer. These programs are provided on the VX/PC disk. See also Chapter 2, Section 2.10.1, "The Color Printing Programs".

HX
SELECT HEX MODE
OPERATING MODE

PURPOSE: Used to increased data transmission speed and when programming with assembly language.

FORMAT: HX (no arguments)

REMARKS: HX sets the current data transmission mode to hexadecimal. Data will be expected to be in hex format until one of these conditions is met:

- 1) The KD command is sent, telling the VX/PC to go into ASCII transmission mode.
- 2) One of the B, C, E, or Q commands is sent with the required argument(s) given in ASCII format.

Drawing is usually performed faster in Hex mode because the VX/PC does not have to perform translation between ASCII and binary, and because there are no carriage returns, line feeds, or other delimiters to strip from the incoming data.

***** CAUTION *****

Care must be taken to insure error-free data in hex mode. Single isolated errors are compounded, destroying subsequent information, often causing the VX/PC to become locked up.

I
INITIALIZE TRANSFORMATION MATRIX
3-D COMMAND

PURPOSE: Resets the transformation matrix in preparation for further transformations.

FORMAT: I (no arguments)

REMARKS: The I command resets the current drawing point to (0,0) in 2-D mode and (0,0,0) in 3-D mode and initializes the image transformation matrix to identity, cancelling previous rotation, scaling, and translation commands.

All rotate, scale, translate, commands are cumulative.

This command is automatically performed when the VX/PC is first turned on, when a RESET is performed, or when a G or G0 command is transmitted.

JA
ADJUST CHARACTER ANGLE
TEXT PRIMITIVE

PURPOSE: Changes character orientation in two ways.

FORMAT: JA slant angle, rotation angle

slant angle = byte 0,1, or 2
rotation angle = byte 0...7 units of 45 degrees

REMARKS: This command adjusts the slant angle and rotation angle for characters. There are two kinds of character angles that can be varied with the JA command:

1) Slant angle of a character is defined as follows:

- 0 = Normal upright
- 1 = Leaning forward 45 degrees
- 2 = Leaning backward 45 degrees

2) Rotation angle (0 to 7) of the character. Rotation starts at 0 (the conventional upright vertical position) and proceeds counterclockwise in steps of 45 degrees.

Characters are drawn starting with the upper left hand corner of the original character when rotation equals 0.

The default value for both slant and rotation is 0.

EXAMPLES: JA 1 0 creates upright characters slanting forward - as in italics.

JA 0 4 displays subsequently printed characters upside down.

JA (continued)

JA 1 4 characters will be printed upside
down and slanting to the right.

NOTES: When using this command, you may want to adjust
the horizontal or vertical spacing for various
angle and rotations, using the JS command.

The JA command has no effect on user-defined
characters created with the JD command.

PURPOSE: Allows custom character sets.

FORMAT: JD character, row 1, row 2, ... row 8

character = any ASCII character from 32 to 126
row 1..row 8 = byte values 0...255

REMARKS: This command is used to design custom characters and typefaces using an 8 x 8 cell. First, a standard ASCII character is redefined. The redefined character can then be displayed by using the standard ASCII character in a string.

The VX/PC default character set uses a 8 x 8 matrix. However, each default character uses only the five leftmost columns of the eight available dots, leaving three pixel columns blank in each cell.

***** CAUTION *****

There must not be any spaces between the JD and the character that is being redefined. This character is always specified in its ASCII format. When in hex mode, the ASCII character is still used.

To design new characters or fonts:

- 1) Draw the character in 8 x 8 character cell using graph paper. Row 1 is the top of the pattern, and the high order bit is the left border of the pattern.
- 2) Write down the binary numbers for the eight rows of the matrix (on = 1, off = 0).
- 3) Convert each binary number to its decimal or hex equivalent "row" values and specify the character name, which can be any valid ASCII character.

EXAMPLE: Redefine the "#" character (ASCII 35) as a solid box or cursor:

Binary	Hex	ASCII
0000 0000	00	0
0111 1110	7E	126
0111 1110	7E	126
0111 1110	7E	126
0111 1110	7E	126
0111 1110	7E	126
0111 1110	7E	126
0000 0000	00	0

The command to redefine this character is:

```
JD# 0 126 126 126 126 126 126 0
```

The "#" will now display as this solid cursor character. Any of the 95 ASCII characters can be redefined in the same way. Note that the "#" immediately follows the "JD" part of the command.

NOTES: Be careful of placing characters too close to each other. If the entire 8 x 8 cell is used, you may want to change horizontal character spacing, using the JS command, to prevent character overlap.

Custom character sets must be redefined after the system has been RESET or the G or G0 or JN commands have been given.

JM
CHANGE CHARACTER MAGNIFICATION
TEXT PRIMITIVE

PURPOSE: Changes character size.

FORMAT: JM factor

factor = byte 1...16

REMARKS: Characters displayed after this command is sent can be magnified up to 16 times with the JM command. The default magnification is 1. The table below lists the possibilities.

Factor	Pixels
-----	-----
1	5 x 9
2	10 x 18
3	15 x 27
4	20 x 36
5	25 x 45
6	30 x 54
...	
16	80 x 144

The horizontal character spacing is magnified along with the character.

When one uses the JM command, the JD (Design Character) command is still limited to an 8 x 8 cell. The JM command simply magnifies each pixel in this 8 x 8 cell.

EXAMPLE: JM 16 All text displayed after this command will be magnified 16 times. For each of the original 64 pixels, 256 (2^{16}) pixels will be displayed, for a total of 16384 pixels.

NOTE: Since each pixel is magnified, diagonal edges will not appear as smooth as when magnification is 1.

JN
DISPLAY DEFAULT CHARACTER SET
TEXT PRIMITIVE

PURPOSE: Returns any redefined character set to the default ASCII character set.

FORMAT: JN (no arguments)

REMARKS: This command cancels the effect of all previous JD commands, changing any redefined characters back to the default character set of 95 ASCII characters, from ASCII 32 (space) to ASCII 126.

The default character set is also activated when the host is first turned on, when the G or G0 command is given, or when the RESET.EXE program is run.

NOTE: Custom character sets must be redefined after the system has been RESET or the G or G0 or JN commands have been given.

JR
SET RECTANGLE FILL PATTERN
TEXT PRIMITIVE

PURPOSE: Creates a two dimensional fill pattern.

FORMAT: JR row 1, row 2, ... row 8
row 1...row 8 = bytes 0...255 for each row

REMARKS: JR is used to define two dimensional patterns for display when the RF (Rectangular Fill) command is used. Other fills and characters are not affected by this command. The pattern is designed using an 8 x 8 grid. Row 1 is the bottom of the pattern, and the low order bit is the left border of the pattern. The pattern repeats from the lower left corner in the X and Y directions as the pattern is drawn.

This command can be used to crosshatch or otherwise differentiate areas of the same color.

EXAMPLE:

Binary	Hex	ASCII
-----	---	-----
1000 0001	81	129
0100 0010	42	66
0010 0100	24	36
0001 1000	18	24
0001 1000	18	24
0010 0100	24	36
0100 0010	42	66
1000 0001	81	129

The command to perform this, while in ASCII mode, is:

JR 129 66 36 24 24 36 66 129

When the rectangular fill is used, it will now display an "X" pattern as above instead of a solid color.

NOTES: The G and G0 commands and resetting the VX/PC cards will reset the rectangular fill pattern to all binary ones.

NOTES: The color drawing mode selected and the background color will determine what colors are displayed in the zero bits and the one bits. For more information, see the COLOR DRAWING MODE (OR, RA, RC, and RE) commands in this section, and Chapter 2, Section 2.6.2, "Color Drawing Modes".

One dimensional patterns should be set with the N command (Set Pattern Register).

JS
SET CHARACTER SPACING
TEXT PRIMITIVE

PURPOSE: Change intercharacter and line spacing.

FORMAT: JS horizontal spacing, vertical spacing, line spacing

horizontal spacing	=	signed word	+/-32767 pixels
vertical spacing	=	signed word	+/-32767 pixels
line spacing	=	signed word	+/-32767 pixels

REMARKS: This command adjusts horizontal, vertical, and line spacing between subsequently displayed characters. Adjustments in the default values are often useful when characters are redefined with the JD command or when characters are slanted with the JA command.

Horizontal spacing is measured from the first column of a character to the first column of the next character. This can be positive or negative. When positive, characters will read in the normal fashion from left to right. If negative, characters display right to left. The default value is seven pixels between the beginning of one character and the beginning of the next.

Vertical spacing is measured from the top row of one character to the top row of the next. If this value is not 0, subsequently displayed characters will be offset vertically. The offset can be positive or negative. Positive spacing makes the characters space down the screen. Negative vertical spacing will print upwards. The default value is 0; each character will print on the same line as the last character.

Line spacing, also positive or negative, is the number of pixels a new line will move up or down after a carriage return. The default is twelve pixels, which makes lines four pixels apart - twelve pixels from the top of one line to the top of the next minus an eight pixel character height.

Horizontal, vertical, and line spacing are adjusted according to the magnification factor set by the JM command.

JS (continued)

EXAMPLES: Positive vertical spacing makes later printed characters move down, like this:

s
l
a
n
t

This spacing can be done with these commands:

JS 7 8 12 \$slant

Negative vertical spacing makes later printed characters move up, like this:

e
p
o
l
s

This spacing can be done with these commands:

JS -7 -8 12 \$slope

K2
SELECT 2-D COORDINATES
OPERATING MODE

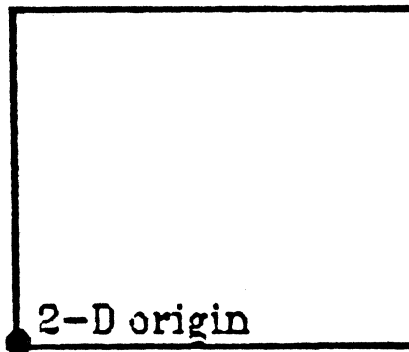
PURPOSE: Set 2-D coordinate mode.

FORMAT: K2 (no arguments)

REMARKS: When 2-D mode is set, the screen location $X=0$, $Y=0$ is the leftmost bottom pixel on the screen, and only the X and Y coordinates of any command are used.

Besides setting 2-D mode, this command initializes the transformation matrix and resets the current drawing point to $X=0$, $Y=0$. The command also sets the viewport to the default, full screen.

Illustration 4.5
The 2-D Origin



K3
SELECT 3-D COORDINATES
OPERATING MODE

PURPOSE: Set 3-D mode.

FORMAT: K3 (no arguments)

REMARKS: The DOT, LINE, MOVE, POLYGON, and FILLED POLYGON and other commands can use the 3-D X,Y,Z coordinate system. These coordinates are referenced to the object-space origin, $X=0$, $Y=0$, and $Z=0$, which is the center of the viewport rather than the lower left screen corner as in 2-D.

The Z-coordinate represents the third dimension of depth, the apparent distance of the object from the observer. Z-coordinates can range from -32767 to +32767. Values of less than zero are clipped unless the Z axis has been translated. For more information on clipping, see Chapter 2, Section 2.7.1, "Clipping".

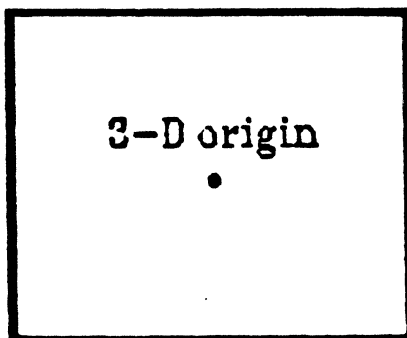
K3 performs an I command, initializing the transformation matrix and resetting the current drawing point to $X=0$, $Y=0$, and $Z=0$. The viewport is also returned to full screen.

Points outside the viewport in 3-D mode are clipped.

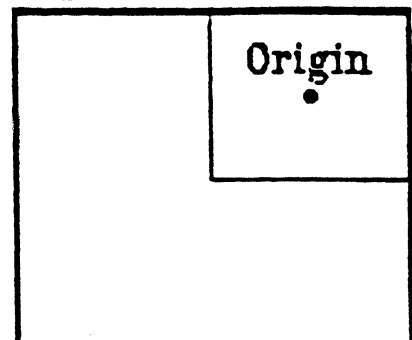
2-D and 3-D modes can be mixed in the same image.

Illustration 4.6
The 3-D Origin

Full Screen Viewport



1/4 Screen Viewport



KA
SELECT ABSOLUTE COORDINATES
OPERATING MODE

PURPOSE: Set Absolute coordinates.

FORMAT: KA (no arguments)

REMARKS: In 2-D mode, the X and Y coordinates are calculated from the origin ($x=0, y=0$) which is at the bottom left corner of the screen.

For example, in 2-D mode, the coordinates 100, 100 are always at the same physical location on the screen when in absolute mode.

In 3-D mode, coordinates X, Y, and Z are referenced to $(0,0,0)$, the center of the viewport.

KB
SELECT BLANK VIDEO MODE
OPERATING MODE

PURPOSE: Set blank video mode.

FORMAT: KB (no arguments)

REMARKS: Blank mode avoids streaks in a changing image - a phenomenon that often happens in FLASH mode. This is possible because the display is only updated during retrace periods, a time when the image will not be disturbed. Update speed is slower in BLANK mode than it is in FLASH mode.

KD
SELECT ASCII DECIMAL MODE
OPERATING MODE

PURPOSE: Set ASCII transmission mode.

FORMAT: KD (no arguments)

REMARKS: KD returns the current data transmission mode to ASCII. Data will be expected to be in ASCII format until one of these conditions is met:

- 1) The HX command is sent, telling the VX/PC to go into hexadecimal transmission mode.
- 2) One of the B, C, E, or Q commands is sent with the required argument(s) given in hexadecimal notation.

KF
SELECT FLASH VIDEO MODE
OPERATING MODE

PURPOSE: Sets Flash mode for greater transmission speed.

FORMAT: KF (no arguments)

REMARKS: In FLASH mode, the display can be updated continually rather than only during the vertical retrace period - hence update is faster. However, the screen may flash short horizontal streaks as the image is changed.

KR
SELECT RELATIVE COORDINATES
OPERATING MODE

PURPOSE: Sets relative coordinate mode.

FORMAT: KR (no arguments)

REMARKS: In relative mode, all subsequent X,Y or X,Y,Z values are relative to the current drawing point rather than the origin, as in absolute mode.

For example, if the current drawing point is X=0, Y=0 in 2-D mode, a command to move 100 units the X axis and 100 units the Y axis would be "M 100 100". The current drawing point would then be at X=100, Y=100. Were another move command to be given to move 100 more units in both directions, the current drawing point will be X=200, Y=200.

Relative mode is useful for expressing coordinates as lengths (delta-x, delta-y and delta-z) rather than calculating a new absolute position. This procedure can be used when a figure is to be duplicated at a new location.

L
LINE
GRAPHIC PRIMITIVE

PURPOSE: Draws a line from the current drawing point to the coordinates specified.

FORMAT: L x, y, z

x = signed word +-32767 units
y = signed word +-32767 units
z = signed word +-32767 units

REMARKS: In 2-D mode, L draws a line from the current drawing point to the x,y position on the screen.

In 3-D mode, this command draws a line to X,Y,Z coordinate values in object space.

The color of the line will be the current drawing color. After execution of this command, the current drawing point will be the endpoint specified in the command.

This command is affected by the pattern register. To draw a solid line, the register must contain all ones.

EXAMPLES: KA L 100 100 Draws a line from the current drawing point to X = 100, Y=100 with absolute coordinates in 2-D space.

KR L 100 -100 Draws a line from the current drawing point to a point 100 units to the right of and 100 units lower than the current drawing point, when in relative coordinates and 2-D mode.

KA K3 L -20 30 100 Draws a line in 3-D space, to X=-20, Y=30, and Z=100.

PURPOSE: Moves the current drawing point to a new location.

FORMAT: M x, y, z

x = signed word +-32767 units
y = signed word +-32767 units
z = signed word +-32767 units

REMARKS: Similar to lifting a pen from the paper surface and moving it to another point, this command moves the starting point for the next command.

MOVE is useful for specifying beginning points of lines, the seed points of complex fills, and the starting points of text.

The current drawing point moves to the new X,Y position on the screen without drawing any lines on the display.

EXAMPLES: KA M 100 100 Moves the current drawing point to X = 100, Y = 100 in absolute mode.

KR M 100 -100 Moves the current drawing point to the point 100 units to the right and 100 units lower on the screen, when in relative mode.

KA M 100 -100 In absolute mode, the current drawing point will be moved off the screen to Y=-100.

KA K3 M 100 100 200 Moves the current drawing point to X = 100, Y = 100, and Z = 200 in 3-D mode.

N
SET PATTERN REGISTER
GRAPHIC PRIMITIVE

PURPOSE: Sets up a one dimensional dot or dash pattern.

FORMAT: N pattern

pattern = unsigned word 0...65535

REMARKS: The N command permits custom design of dot and dash type one dimensional patterns. "Pattern" is a number whose 16-bit binary equivalent represents the dot-dash pattern (1 = dot on, 0 = dot off). The least significant bit of the pattern is always drawn first. The pattern will repeat once for each group of 16 pixels drawn.

This command may be used when drawing lines, arcs, polygons, and filled polygons.

Patterns are designed in three steps:

- 1) Draw the pattern on paper as a series of 16 cells, each on or off.
- 2) Write down the binary equivalent of the pattern.
3. Convert this binary number into its ASCII or HEX equivalent and use it as the "pattern".

EXAMPLES: N 65535 - Sets all bits on, so a solid line is produced, which is the default.

N 1 - Draws a dotted line (. . . .) with one dot every sixteen pixels.

N 52428 - (CCCC hex) - produces dot pattern with 2 bits on, 2 bits off.

N 4080 - (0FF8 hex) - produces dash pattern.

NOTES: If all bits are set off (0), nothing will be drawn using the Line, Arc, Polygon, or Filled Polygon commands.

This commands sets a one dimensional pattern. If a two dimensional pattern is required for a rectangular fill, use the JR (Set Rectangle Fill Pattern) command.

PURPOSE: Draws arcs starting at the current drawing point.

FORMAT: OA radius, start angle, arc angle

radius = unsigned word 0...65535 pixels
start angle = signed word +-3600 degree tenths
arc angle = signed word +-3600 degree tenths

REMARKS: OA allows arcs to be connected as sectionals to lines and polygons. The current drawing point is the original starting point for the arc.

After completion of the command, the current drawing point is moved to the final ending point of the arc.

EXAMPLES: OA 100 900 900 draws a ninety degree arc of 100 pixel radius starting at the CDP.

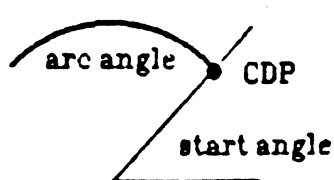
OA 100 0 3600 draws a complete circle with 100 pixel radius, starting at the CDP.

NOTES: To draw a circle, set "arc angle" to 3600.

Arcs and circles cannot be rotated, scaled, or translated in either 2-D or 3-D modes.

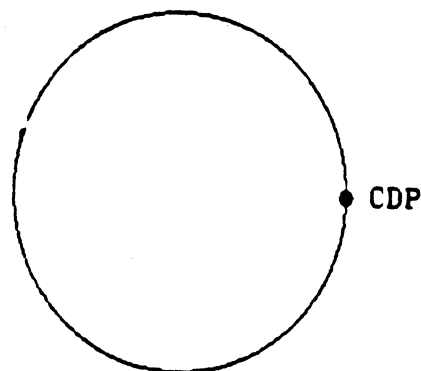
When in RC (Replace Complement) color drawing mode, gaps in the arc may be produced.

Illustration 4.7
Using The Originate Arc Command



OA 100 450 900

radius = 100 pixels
start angle = 45°
arc angle = 90°



OA 100 0 3600
radius = 100 pixels
start angle = 0°
arc angle = 360°

OF
TURN VIDEO OFF
VIDEO COMMAND

PURPOSE: Turns the video signal off.

FORMAT: OF (no arguments)

REMARKS: OF turns the video display of the VX/PC display off without changing the screen image.

This feature can be used when the VX/PC is used for long periods of time with no need for the monitor to be on.

ON
TURN VIDEO ON
VIDEO COMMAND

PURPOSE: Turns the video signal on.

FORMAT: ON (no arguments)

REMARKS: ON turns the video display of the VX/PC display on without changing the original image.

OR
OR MODE
COLOR DRAWING MODE

PURPOSE: Allows overlaid colors to mix.

FORMAT: OR (no arguments)

REMARKS: When in OR mode, any colors to be drawn will be logically ORed with any colors already residing in the affected pixels. In any areas where an existing color and a new color overlap, the color actually displayed will be a result of both color numbers being logically ORed together. The new color number can be determined by looking at the binary equivalents of the color numbers for the affected pixels. If either bit is a 1, the result is a 1; if both bits are 0, the result is 0.

EXAMPLES: Using the default color table and mixing pure red (color 7) and pure green (color 56), the bitplanes appear as follows:

	RED bits ----	GREEN bits ----	BLUE bits ----	
Red = 7 =	111	000	000	
Green = 56 =	000	111	000	
OR Result:	111	111	000	= 63 = Yellow

NOTE: For more information, see Chapter 2, Section 2.6.2, "Color Drawing Modes".

PURPOSE: Draws unfilled polygons.

FORMAT: P count, x1, y1, z1, x2, y2, z2,...

count = unsigned word	3...65535 vertices
x1,.. = signed word	+ - 32767 units
y1,.. = signed word	+ - 32767 units
z1,.. = signed word	+ - 32767 units

REMARKS: The P command draws an unfilled polygon with vertices x1, y1, x2, y2... or if in 3-D mode at x1, y1, z1, x2, y2, z2..., and the polygon is automatically closed. The final coordinate closing the figure need not be entered because it is also the first. The polygon is not filled and the perimeter color is the current drawing color.

The first vertex specified is where the polygon will start to be drawn, regardless of the current drawing point.

Vertices may be specified in either clockwise or counterclockwise order.

In both absolute and relative mode, the current drawing point becomes the first vertex specified (which is also the last) after execution of the command.

EXAMPLES: KA P3 100 100 150 150 200 100

Draws a triangle starting at X = 100, Y = 100, The last line, from X = 200, Y = 100 to X = 100, Y = 100, is automatically drawn.

KR P3 100 100 50 50 50 -50

Draws the same triangle at the same location, but in relative mode.

KA K3 P3 5 5 10 -5 0 10 5 -5 10

Draws a triangle in 3-D absolute coordinates.

PAN
PAN VIDEO IMAGE
VIDEO COMMAND

PURPOSE: Allows simulation of image motion and positioning when using the Z (Zoom) command.

FORMAT: PAN x, y

x = unsigned word 0...671

y = unsigned word 0...479

REMARKS: PAN moves the image on the display so that location X,Y in the original image appears at the lower left corner of the screen.

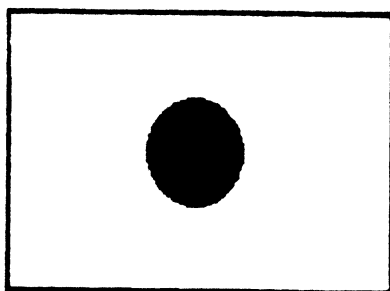
Both X and Y are set to 0 at power up, upon RESET, and when the G or G0 commands are given.

The value X is truncated to the next lower word boundary if the number given is not a multiple of 16.

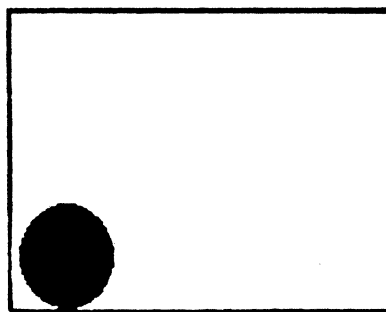
The value Y is truncated to the next lower even number if the number specified is not a multiple of 2.

EXAMPLE: PAN 160 100 Moves the image so that the current position X=160, Y=100 is at the lower left corner of the screen. Points that were drawn off the original screen are not displayed, but the screen contents will wrap around.

Illustration 4.8 Using The PAN Command



PAN 0 0



PAN 300 200

NOTE: PAN can be used separately or in conjunction with Zoom.

Q
DEFINE COLOR LOOKUP TABLE VALUE
COLOR MANIPULATION

PURPOSE: Used to redefine the color lookup table to create custom colors or to reorganize the table.

FORMAT: Q rows, count, r1, g1, b1, r2, g2, b2, ...

row	=	unsigned word	0...511
count	=	unsigned word	1...512
r1, r2, ...	=	byte value	0...255
g1, g2, ...	=	byte value	0...255
b1, b2, ...	=	byte value	0...255

REMARKS: This command is used to create a custom color lookup table. The color table defines red, green, and blue saturation levels for each of the 512 simultaneously displayable colors. Any or all of the table entries can be changed to create new shades and mixtures of colors. This command is used for animation and other special effects, such as grey scales.

Each three byte entry is considered a "row". Starting with "row" and continuing for "count" rows, each row (which is also a color number) in the table has three entries which can be changed to any value between 0 and 255.

For example, the first entry in the default table contains 0 for red, 0 for green, and 0 for blue, which produces black, the lack of any color. Using the Q command, each of these zeroes can be changed. If 255 is entered for red and 0 for green and blue, this entry will contain 255 0 0. When this entry is referenced, pure red with no green or blue will be displayed.

Greys are produced by making the red, green, and blue values in an entry equal, such as 15 15 15.

Q (continued)

The VX/PC determines if transmissions are in ASCII or HEX mode from the format of the arguments in the B, C, E, and Q commands. Command format is changed accordingly.

EXAMPLES: Q 0 1 255 0 0

Changes entry #0 to contain pure red.

Q 10 2 120 120 120 240 240 240

Changes entry #10 to a medium grey,
and entry #11 to a lighter grey.

PURPOSE: Used to re-orient figures in both 2-D and 3-D mode.

FORMAT: 3-D - RX angle 2-D - RZ angle
 RY angle
 RZ angle

angle = signed word +-3600 degree tenths

REMARKS: Figures can be rotated around the X, Y, or Z axes with this command. The figures drawn after a Rotate command will be rotated around the specified axis, by the specified angle.

To rotate on a particular axis, the axis (X, Y, or Z) should be linked to the R command and followed by the number of TENTHS of a degree of rotation, like this:

RX 100	Rotates 10 degrees around X axis
RY 450	Rotates 45 degrees around Y axis
RZ 900	Rotates 90 degrees around Z axis

Rotation is performed around the object space origin in 3-D mode, which is the center of the current viewport. In 2-D mode, rotations are performed around the screen origin which is at the lower left corner of the screen, where X=0 and Y=0.

A figure can be rotated around an external origin or can be made to rotate around a point internal to the figure (spinning the figure).

Rotating a figure around an external axis requires only that the R command be given with the number of degrees, which is specified in tenths of a degree.

To rotate an object around its own center, translate the center to the origin, rotate it, and translate it back. Translating an object to the origin is described under the T (Translate) command.

R (continued)

Illustration 4.9 3-D Direction of Rotation

Axis	If Arguments Are:	
	Positive	Negative
X	Up	Down
Y	Left	Right
Z	Clockwise	Counter-clockwise

EXAMPLES: Draw figure...RX 450...Redraw figure

Rotates the figure 45 degrees around the X axis, in an upward direction. This does not rotate in the X direction, but around the X axis. Perspective changes with the rotation.

Draw figure...RY -300...Redraw figure

Rotate the figure 30 degrees around the Y axis, moving it to the right. A positive rotation would move it towards the left.

NOTES: Rectangles produced with the RF command, any command producing an arc, and text cannot be rotated using the R command.

For more information, see Chapter 2, Section 2.7.2, "Rotating A Figure".

Rotate X (RX) and Rotate Y (RY) are not valid in 2-D mode. For information on how to use rotation in 2-D mode, see Chapter 2, Section 2.7.8, "2-D Use of 3-D Commands".

RA
REPLACE ALL MODE
COLOR DRAWING MODE

PURPOSE: Used when complete replacement of an area is required, as when drawing new characters over old.

FORMAT: RA (no arguments)

REMARKS: The RA command is similar to replace mode (RE command), except that not only are the 1's in the pattern register drawn, but also the zeros.

If all bitplanes are enabled for update and the pattern register contains all ones, any color reference will be to table entry #511 while in RA mode.

There are two ways to use RA and not get white as the drawing color. The first is to turn off certain bitplanes. The second method is to change the contents of color lookup table entry #511.

This command is useful for replacing one character with another, such as a cursor. It draws the background zeroes of the character description in order to completely erase the previously drawn character. This is done automatically when printing text.

EXAMPLE:	RED bits ----	GREEN bits -----	BLUE bits ----
Red = 7 =	111	000	000
Green = 56 =	000	111	000
RA Result:	111	111	111 = 511 = White

NOTES: This result depends on the setting of the pattern register, which is changed with the N or JR commands. If for any bit the pattern bit is a 0, the results will also be a 0.

For more information, see Chapter 2, Section 2.6.2, "Color Drawing Modes".

RC
REPLACE COMPLEMENT MODE
COLOR DRAWING MODE

PURPOSE: Used for non-destructive replacement of an area.

FORMAT: RC (no arguments)

REMARKS: This command complements or reverses 0's and 1's of an existing color. It is useful for displaying cursors or crosshairs: when performed twice, the affected pixels return to their original color.

Each pixel drawn after the RC mode is set will replace the pixel's binary value (0 or 1) with its complement (if 0, it becomes 1, and vice versa). All bitplanes that have been enabled for update by the B command are complemented, regardless of the current drawing color.

EXAMPLE: When mixing pure red (color 7) and pure green (color 56), the bitplanes appear as follows:

	RED bits ----	GREEN bits ----	BLUE bits ----
Red = 7 =	111	000	000
Green = 56 =	does not care		
RC Result:	000	111	000 = 504 = cyan

If another write to the same locations occurs in RC mode, the color will complement itself and become red again.

NOTE: For more information, see Chapter 2, Section 2.6.2, "Color Drawing Modes".

RD
RETURN DRAWING POINT
GRAPHIC PRIMITIVE

PURPOSE: Used to find the location of the current drawing point.

FORMAT: RD (no arguments)

Returns: x = signed word +-32767
y = signed word +-32767

REMARKS: The RD command returns to the host computer the X and Y coordinates of the current drawing point. The values returned can be either positive or negative.

In decimal mode, positive screen coordinates are returned as five digit numbers between "00000" and "32767", padded on the left as necessary. Negative values are returned as positive numbers between "32768" and "65535". The actual negative value can be found by subtracting 65,536 from the returned value.

In hexadecimal mode, values are returned as signed 16 bit word values. Negative values are returned as two's complement numbers.

In 3-D mode, the X and Y values returned are the screen location of the current drawing point when transformed through any current transformation matrix. They may be positive or negative, as above. Z axis coordinates are not returned.

EXAMPLES: M 335 240 RD Returns to the host these values:
335 240.

M -335 -240 RD Returns the values 65201 65296.
Since these are negative values,
the actual values are 65201-65536
=-335, and 65296-65536=-240.

RD (continued)

NOTE: This command is used for figuring correct placement of text, and figures that operate only in 2-D mode, in a 3-D image. For instance, to place text or arcs or filled rectangles on a 3-D image, use this series of commands:

- 1) RD to return to the host the current point.
- 2) K2 to set 2-D mode.
- 3) M to move to the X,Y coordinates returned by RD.
- 4) Draw text or figure.

PURPOSE: Used to replace an area with a new color, rather than mixing the new and overlaid color.

FORMAT: RE (no arguments)

REMARKS: The REPLACE mode permits you to write over an existing image with a new one. It replaces an old image with a new one, without mixing the current and previous colors. This mode is slower than OR mode, because all enabled bitplanes must be written in order to replace the color.

EXAMPLE: When mixing pure red (color 7) and pure green (color #56), the bitplanes appear as follows:

	RED bits ----	GREEN bits ----	BLUE bits ----	
Red = 7 =	111	000	000	
Green = 56 =	000	111	000	
RE Result:	000	111	000	= 56 = Green

NOTE: For more information, see Chapter 2, Section 2.6.2, "Color Drawing Modes".

RF
RECTANGLE FILL
GRAPHIC PRIMITIVE

PURPOSE: Used to draw and fill rectangles and squares.

FORMAT: RF width, height

width = signed word +-32767 pixels
height = signed word +-32767 pixels

REMARKS: RF draws a filled rectangle starting at the current drawing point. It is faster than using the F command, but can only be used for rectangles and squares. The boundary and fill color will be the current drawing color.

Positive values for width will draw to the right, negative values to the left. Positive values for height draw upward, negative values draw downward. The current drawing point is not changed.

Notice that figure size is specified in width and height rather than in screen locations as with the F (Filled Polygon) command. Placement is not affected by whether absolute or relative coordinates are in effect. If the current drawing point is X=100, Y=100, the command "RF 10 10" will draw a square from X=100, Y=100 to X=109, Y=109.

The next three commands series produce the same figure:

```
C7 M 100 100 RF 10 10
KA F 7 4 100 100 109 100 109 109 100 100
KR F 7 4 100 100 9 0 0 9 -9 0
```

The RF command can be used in either 2-D or 3-D mode. However, figures drawn cannot be scaled or rotated. To create a figure that can be rotated, scaled, and transformed, use the F command.

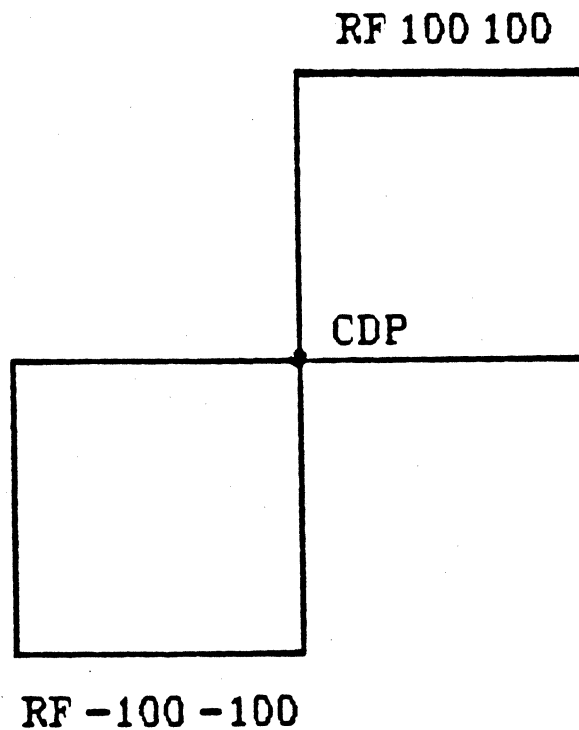
Figures drawn with the RF command are clipped at viewport boundaries in both 2-D and 3-D modes.

The figure is filled using the 8 x 8 pattern defined by the JR command.

EXAMPLES: RF 100 100 Draws a 100 pixel by 100 pixel square at the current drawing point and fills it with the current drawing color.

RF -100 -100 Draws the same square but it is drawn down and to the left of the current drawing point.

Illustration 4.10
Using The RF Command



RL
RETURN LIGHT PEN POSITION
GRAPHIC PRIMITIVE

PURPOSE: Returns to the host the current position of the light pen if it is over a non-black pixel.

FORMAT: RL status

status = byte value 0 or 1

Returns: Pen depressed = byte value 0 or 1
X coordinate = unsigned word 0...671
Y coordinate = unsigned word 0...479

REMARKS: RL enables the use of the light pen with the VX/PC. Status and coordinate data are returned to the host when the light pen is near the screen surface and a non-black color is detected.

"Status", which must be supplied with this command, determines whether to wait for the light pen tip to be depressed or not before returning coordinates to the host computer.

This command returns three values to the host, one byte value and then two word values. The byte value is zero if the light pen tip is not depressed upon light pen detect and a one if the tip is depressed. The X and Y coordinates then detected are returned as word values.

If "status" is zero, then a value of zero or one may be returned for the pen depressed condition.

If "status" is one, detected coordinates will only be returned if the pen tip is depressed.

In ASCII mode, the status is returned as a three digit number, padded on the left with zeroes as necessary. Next is the X coordinate as a five byte field, then the Y coordinate is returned, also as a five byte field. The fields returned are also padded if necessary. Carriage returns are returned after each of the three fields.

EXAMPLES: RL 0

Returns the value "000" in S\$, "00336" in X\$, and "00240" in Y\$ if the pixel detected is at screen center. If the pen tip was depressed when the dot was detected, S\$ will contain "001".

RL 1

Returns the same values as the above example, but only when the tip is depressed.

NOTES: For more information on sending data from the VX/PC to the host computer, see Chapter 3, Section 3.2.2., "Bi-Directional Commands".

An inherent limitation of the Graphics Display Controller chip is that the X coordinate returned may be as much as a half word (8 bits) off in either direction.

RNP or @RNP
READ PIXELS (encoded)
PIXEL COMMAND

PURPOSE: Used to compress images and transfer compressed data to the host computer.

FORMAT: RNP count

count = unsigned word 0...65535 pixels

Returns: encoded color values = unsigned words 0...65535

REMARKS: RNP transfers full or partial images from the display to the host computer in an encoded format. Starting at the current drawing point, "count" pixels are read and passed to the host.

If the pixel "count" exceeds the right edge of the screen, reading will continue on the next lower pixel row, starting at the left.

A "count" of 0 moves the current drawing position to the upper left corner and transfers the entire screen image to the host.

When the pixel values are passed to the host, a process known as Run Length Encoding is performed, compressing the data. This often reduces the data to one fifth its original length, making total picture storage practical on floppy disks. Images with large contiguous areas of the same color will compress the most. For more information on Run Length Encoding, see the System Overview section of this manual.

Each pixel color is encoded in an unsigned 16 bit word value as follows: the nine lowest bits are the color value and the seven high bits represent the pixel repetition count. All 512 screen colors can be represented by the nine bits given to the color. The repetition count, using seven bits, can represent up to 127 consecutive occurrences of the same color.

The formula for calculating the actual value returned to the host is:

Returned Value = 512 x Count + Color.

Transmission to the host computer by the RNP command is terminated by one word of zero ("00000").

The VX/PC has nine bitplanes of color information for each pixel, organized as one pixel per plane. The Bitplane Write Mask Register, controlled by the B command, allows use of selected bitplanes. There are two versions of the RP command - one ignores the Bitplane Write Mask, the second uses the mask.

The first version is simply RNP. All bits for a pixel are read at one time; the Bitplane Write Masked is ignored.

@RNP, the second version of this command, reads only the bitplanes enabled by the B command. If a B 511 command (all bitplanes enabled) has been given, the effect will be the same as using RNP without the '@'.

Values are transmitted in the current transmission mode, ASCII or hex. When ASCII mode is active, the color value of each pixel is returned in a five position string, terminated by a carriage return. Values with less than five digits are padded with zeroes on the left. When hex mode is active, the color values are returned as 16 bits with no carriage return between successive values.

EXAMPLES: M 0 479 RNP 10

Returns the values for the first ten positions of the top row of pixels, from X=0, Y=479 to X=9, Y=479. If all these pixels are color number #7, the value 5127 ($512 \times 10 + 7 = 5127$) is sent, then the transmission is ended with one word of "00000".

E 7 RNP 0

Sets the entire screen to color #7, and returns in ASCII, a series of 5 digit bytes, most of which are 65031 ($512 \times 127 + 7$). Transmission will be terminated with one word of "00000".

RNP (continued)

B 255 RNP 1

Enables bitplanes 1-8 (disabling bitplane 9 - the binary pattern is 011 111 111), and reads the pixel at the current drawing point. If the bit pattern of this pixel is 111 111 111 (511 in decimal, which is white in the default color table), the VX/PC will return these values: 1023 00000 (512 x 1 + 511 and the terminating set of binary zeroes).

B 255 @RNP 1

Enables bitplanes 1-8 and reads the pixel at the current drawing point. If the bit pattern of this pixel is 111 111 111 (511 in decimal) the VX/PC will return these values: 767 00000 (512 x 1 + 255 and the terminating set of binary zeroes. The bitplane mask allows only the first 8 bitplanes to be read when the symbol '@' is appended to the RNP command. Any bits not read are set to zero.

C 7 D 100 100 B 258 @RNP 1

Sets the pixel at coordinate X=100, Y=100 to color #7, red in the default color table. The Bitplane Write Mask Register is set to 258 (pattern: 100 000 010). One pixel is then read (pattern: 000 000 111, decimal 7), encoded, and returned to the host computer. The values returned are: 514 00000 (512 x 1 + 2).

NOTES: Images transfer slightly faster in Flash mode.

Images transferred to the host with this command may be restored with the WNP command.

RNP and @RNP do not use DMA mode.

For more information on specialized uses of this command, see Chapter 2, Section 2.8, "Pixel and RAM Commands".

RNR
READ GRAPHICS RAM (encoded)
RAM COMMAND

PURPOSE: Used to compress images and transfer compressed data to the host computer.

FORMAT: RNR bitplane, count

bitplane = byte value 1...9
count = unsigned word 0...65535 words

Returns: RAM contents encoded = unsigned words 0...65535

REMARKS: The RNR command transfers full or partial images from graphics RAM to the host computer in an encoded format, from the specified bitplane. Starting at the current drawing point, "count" unencoded 16 bit words of graphics memory are read, encoded, and passed to the host.

The encoding process works as follows:

- 1) Only words of "0000" (all bits off) and "FFFF" (all bits on) are encoded, as they are the most likely to occur. Other values are not encoded and are passed to the host in their original form.
- 2) When the system finds a word to encode in graphics memory, it determines the repetition count by counting words until it finds a different value.
- 3) If it finds a word of "0000" or "FFFF", it passes the word to the host. Then it passes the repetition count, also an unsigned word.
- 4) This process repeats until either "count" words have been analyzed, or the end of the bitplane is found.
- 5) To signal the end of data, the VX/PC processor sends two words of all zeroes.

A "count" of 0 moves the current drawing position to the upper left corner and transfers an entire bitplane.

Values are transmitted in the current transmission mode, ASCII or hex. When ASCII mode is active, the returned values are in a five position string, terminated by a carriage return. Values with less than five digits are padded with zeroes on the left. When hex mode is active, the values are returned as 16 bits with no carriage return between successive values.

If the "count" exceeds the right edge of the screen, reading will continue on the next lower row, starting at the left.

Reading starts at the beginning of the 16 pixel word in which the current drawing point is located.

EXAMPLES: E0 M 0 479 RNR 1 42

Reads and attempts to encode all values on the top row in bitplane 1. Since E0 has zeroed all graphics memory, the values will returned are "00000 00042 00000 00000".

B1 E0 RNR 1 0

Clears bitplane 1 (B1 E0), and reads the entire bit plane, finding only zeroes. The values returned to the host will be "00000 20160 00000 00000". The first set of zeroes indicates the word repeating is "00000"; 20160 is the repeat count (42 words per line x 480 lines); the last two sets of zeroes terminate the transmission. These four words are the maximum possible compression.

NOTE: Images transfer slightly faster in Flash mode.

Images transferred to the host with this command may be restored with the WNR command.

For more information on specialized uses of this command, see Chapter 2, Section 2.8, "Pixel and RAM Commands".

PURPOSE: Used to transfer pixel values to the host computer in their original, not compressed, format.

FORMAT: RP count

count = unsigned word 0...65535 pixels

Returns: color values = unsigned words 0...511

REMARKS: RP transfers full or partial images from the display to the host computer, one pixel at a time. Starting at the current drawing point, "count" pixels are read and passed to the host.

If the pixel "count" exceeds the right edge of the screen, reading will continue on the next lower pixel row, starting at the left.

A "count" of 0 moves the current drawing position to the upper left corner and transfers the entire screen image to the host. This is 322,560 pixels or 645,120 bytes. If an entire screen is to be transferred, the encoded version of this command (RNP) would execute more quickly and the data require less disk space.

Values are transmitted in the current transmission mode, ASCII or hex. When ASCII mode is active, the color value of each pixel is returned in a five position string terminated by a carriage return. Values with less than five digits are padded with zeroes on the left. When hex mode is active, the color values are returned as 16 bits with no carriage return between successive values.

The VX/PC has nine bitplanes of color information for each pixel, organized as one pixel per plane. The Bitplane Write Mask Register, controlled by the B command, allows use of selected bitplanes. There are two versions of the RP command - one ignores the Bitplane Write Mask, the second uses the mask.

RP (continued)

The first version is simply RNP. All bits for a pixel are read at one time; the Bitplane Write Masked is ignored.

@RNP, the second version of this command, reads only the bitplanes enabled by the B command. If a B 511 command (all bitplanes enabled) has been given, the effect will be the same as using RNP without the '@'.

EXAMPLE: RP 3

Returns the color of three pixels starting at the current drawing point. If all 3 pixels are color #7 and ASCII mode is set, the values returned will be "00007 00007 00007".

B 255 RNP 1

Enables bitplanes 1-8 (disabling bitplane 9 - the binary pattern is 011 111 111), and reads the pixel at the current drawing point. If the bit pattern of this pixel is 111 111 111 (511 in decimal, which is white in the default color table), the VX/PC will return this value: 00511. The Bitplane Mask Register is ignored.

B 255 @RNP 1

Enables bitplanes 1-8 and reads the pixel at the current drawing point. If the bit pattern of this pixel is 111 111 111 (511 in decimal) the VX/PC will return 00255. The bitplane mask allows only the first 8 bitplanes to be read when the symbol '@' is appended to the RNP command. Any bits not read are set to zero.

C 7 D 100 100 B 258 @RNP 1

Sets the pixel at coordinate X=100, Y=100 to color #7, red in the default color table. The Bitplane Write Mask Register is set to 258 (this pattern is: 100 000 010). One pixel is then read (pattern: 000 000 111, decimal 7) and returned to the host computer. The value returned is: 00002 (the binary pattern of two, using nine bits, is 000 000 010).

NOTES: Images transfer slightly faster in Flash mode (KF command).

For more information on specialized uses of this command, see Chapter 2, Section 2.8, "Pixel and RAM Commands".

RQ
RETURN COLOR LOOKUP TABLE VALUE
COLOR MANIPULATION

PURPOSE: Used to transfer to the host computer the contents of the color lookup table.

FORMAT: RQ rows, count

rows = unsigned word 0...511
count = unsigned word 1...512

Returns: red = byte value 0...255
green = byte value 0...255
blue = byte value 0...255

REMARKS: RQ transmits the red, green, and blue values in the color lookup table to the host computer, beginning with "row" and continuing for "count" rows. Three byte values are returned for each table entry. The first byte returned indicates the amount of red, the second the amount of green, and the third the amount of blue.

In ASCII mode, the values are returned as three fixed length strings of three ASCII digits (000 to 255) each with leading zeros, and followed by a carriage return with no linefeed.

In Hex mode, each of the three bytes are returned as byte values from 00 to FF.

This command always performs in FLASH mode, even if the system is currently using Blank mode.

EXAMPLES: RQ 7 1 Returns the values in entry #7 to the host computer. In the default table, these will be 240 for red, 16 for green, and 16 for blue, or the hex equivalents.

RQ 0 512 Returns the values in every entry of the table.

PURPOSE: Used to transfer graphics RAM contents to the host computer in its original, not compressed, format.

FORMAT: RR bitplane, count

bitplane = byte value 1...9
count = unsigned word 0...65535 words

Returns: RAM contents = unsigned words 0...65535

REMARKS: RR transfers full or partial images directly from graphics RAM to the host computer, sixteen bits at a time, from the specified bitplane. Starting at the current drawing point, "count" words are read and passed to the host.

A "count" of 0 moves the current drawing position to the upper left corner and transfers an entire bitplane, which is 20,160 words or 40,320 bytes. For long transfers the encoded version of this command (RNR), can be used.

Values are transmitted in the current transmission mode, ASCII or hex. When ASCII mode is active, the returned values are in a five position string, terminated by a carriage return. Values with less than five digits are padded with zeroes on the left. When hex mode is active, the color values are returned as 16 bits with no carriage return between successive values.

If the "count" exceeds the right edge of the screen, reading will continue on the next lower row, starting at the left.

Reading starts at the boundary of the 16 pixel word in which the current drawing point is located.

RR (continued)

The RR command can transfer images in either DMA or in normal (non-DMA) mode. DMA mode transfers data considerably faster than non-DMA mode. Combining VX/PC DMA mode, IBM DMA, and hex mode is the fastest method of transfer. An entire image can be transferred in as little as 1.5 seconds.

When not using DMA mode to transfer images, the RNR command (the Run Length Encoded version of RR), transfers data faster.

EXAMPLES: RR 1 10 Reads ten words (160 pixels) from bitplane 1 and sends them to the host in the current transmission mode.

M 0 240 RR 3 42 Moves current drawing point to X=0, Y=240 and returns RAM contents for the entire row in bitplane 3, 42 words.

NOTE: Images transfer faster in Flash Mode (KF).

For more information on specialized uses of this command, see Chapter 2, Section 2.8, "Pixel and RAM Commands".

For more information on setting up IBM DMA transfers, see Appendix B, "Using Direct Memory Access".

PURPOSE: Used to display the version number of the PROMS on the VX/PC processor board.

FORMAT: RV (no arguments)

REMARKS: RV displays the version number of the PROMs installed in the VX/PC. It first erases the screen, then prints the version number at screen center in white.

Any time it is necessary to call Vectrix Customer Service, this coding sequence should be known and communicated to the Vectrix representative. This may immediately indicate a problem.

A typical version number and its explanation is shown next.

V2.2R=I

Position	Codes	Meaning
2,3,4	-----	Revision level Position 3: ":" External Sync "." Internal Sync
5	R C A T blank	Radio Shack Inkjet Printer Canon or Quadram Inkjet Printer ACT II Printer Transtar Color Printer PRISM 80 or 132 Color Printer
6	"=" " "- "	8K RAM (Static) 4K RAM (Static)
7	I	IBM version (will always appear)

S
SCALE
3-D COMMAND

PURPOSE: Used to change the scale of a figure on any of three axes.

FORMAT: 3-D - SX multiply-factor, divide-factor
 SY multiply-factor, divide-factor
 SZ multiply-factor, divide-factor

2-D - SX multiply-factor, divide-factor
 SY multiply-factor, divide-factor

multiply factor = signed word +-32767
divide factor = signed word +-32767

REMARKS: The scaling command enables enlarging or reducing a displayed image. It increases or decreases the scales of subsequently drawn lines in X, Y, and Z axes with reference to the object-space origin (0,0,0). The size change is based on the signed multiply and divide factors.

Two signed integers are used to provide a full range of factors. A negative scale factor will provide a mirror image around an axis.

Note that only one of the factors should be negative to specify mirror image scaling; a negative number divided by a negative is positive.

The positive Z axis is defined as going "into" the screen, so a Z scaling ratio greater than one moves an object farther away from the viewer, appearing smaller on the screen.

Like the rotate command, the position of the object in relation to the origin makes a difference in how Scale performs. For a look at how this works, see "Scaling" in the System Overview.

EXAMPLE: Draw figure...S 2 1...Redraw figure

Doubles the length of the figure. The actual position the figure occupies on the screen depends on the relation of the original position and the origin.

NOTES: The VX/PC does not allow a "divide by zero" error. If zero is given as the divide factor, a value of one is used.

Rectangles produced with the RF command, any command producing an arc, and text cannot be scaled with the S command.

SI
SELECT IBM MODE
OPERATING MODE

PURPOSE: Used to switch to IBM mode.

FORMAT: SI (no arguments)

REMARKS: SI is used to switch the VX/PC into IBM mode, and is the complement to the SV command. After this command is given, the VX/PC card set will act just like the IBM color graphics card. IBM text and color graphics commands can then be used in their normal manner. This is possible because the VX/PC card set has the functional equivalent of an IBM color graphics card built in. The only IBM function not available with the VX/PC board set in IBM graphics mode is border colors.

Using a single monitor system, the SI command should be included in any high level language program that uses the SV command to set Vectrix graphics mode. When the program finishes, this command should be issued by the program to switch back into IBM mode.

The SI command can also be used in an ON ERROR type subroutine to catch errors occurring while in Vectrix mode. If a program error occurs while in Vectrix mode and is not trapped, the entire system may become inoperative.

EXAMPLE: 5 OPEN "VECTRIX" FOR OUTPUT AS #1
10 ON ERROR GOTO 1000
20 ON KEY(1) GOSUB 1010 : KEY(1) ON
30 PRINT #1,"SV"
.
.
.PROGRAM CODE HERE...
.
.
1000 PRINT #1,"SI" : PRINT ERR : STOP
1010 INPUT A\$: PRINT #1,"SI" : CLOSE : STOP

NOTE: It is important to understand that the IBM computer and the VX/PC are really two completely separate processors communicating with each other. The SI command does not tell the IBM to recognize the VX/PC cards. Rather, when the VX/PC has been selected to receive output from the IBM, SI turns on IBM graphics mode. For more information on the use of SI and SV, refer to Chapter 3, Section 3.1, "General Programming Overview".

SP
SET PERSPECTIVE SCALE
3-D COMMAND

PURPOSE: Used to change perspective relationships.

FORMAT: SP multiply factor, divide factor

multiply factor = signed word +-32767 pixels
divide factor = signed word +-32767 pixels

REMARKS: The SP command performs several transformation functions with a single command. These functions are:

- 1) SP automatically creates a square aspect ratio regardless of the setting of the 3-D viewport. However, after the command executes, the original viewport is left unchanged, but it must be set before any perspective scaling can be performed.
- 2) It then performs a simultaneous scaling of both the X and Y positions.

Illustration 4.12 The Viewing Pyramid

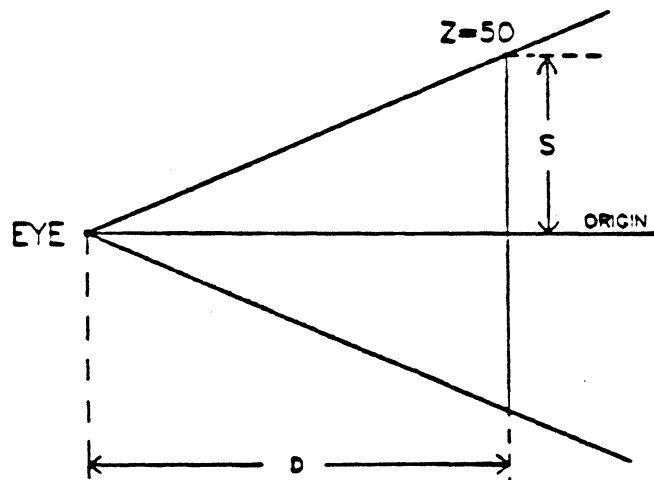


Illustration 4.12 represents the viewing pyramid. "D" (the multiply factor, and also the Z depth) is the depth from the eye (where X,Y, and Z=0), to an arbitrary viewing window along the Z axis. "S" (the divide factor) represents the height of the clipping boundary from the line of the origin.

Through the transformation matrix, different perspective effects can be created by modifying the arguments "D" and "S" in relation to the final Z coordinate established for an object.

If the ratio of D/S is large, the perspective will be similar to an image viewed through a telephoto lens and the object will appear larger.

If the ratio of D/S is small, the perspective will be similar to an image viewed through a wide angle lens and the image will appear to be smaller.

The correct command order to perform perspective scaling is this:

```
K3      - Select 3-D mode
I       - Initialize Transformation Matrix
V       - Set Desired Viewport
```

Draw figure - Optional

```
R,S,T   - Perform Rotate, Scale, or Translate
SP      - Set Perspective Scale
```

Draw Figure

Viewport must always be set before the SP command is used. The SP command should always be given after any rotate, scale or transform commands.

EXAMPLES:

Illustration 4.13A

Telephoto Effect: figure appears closer; perspective is fore-shortened.

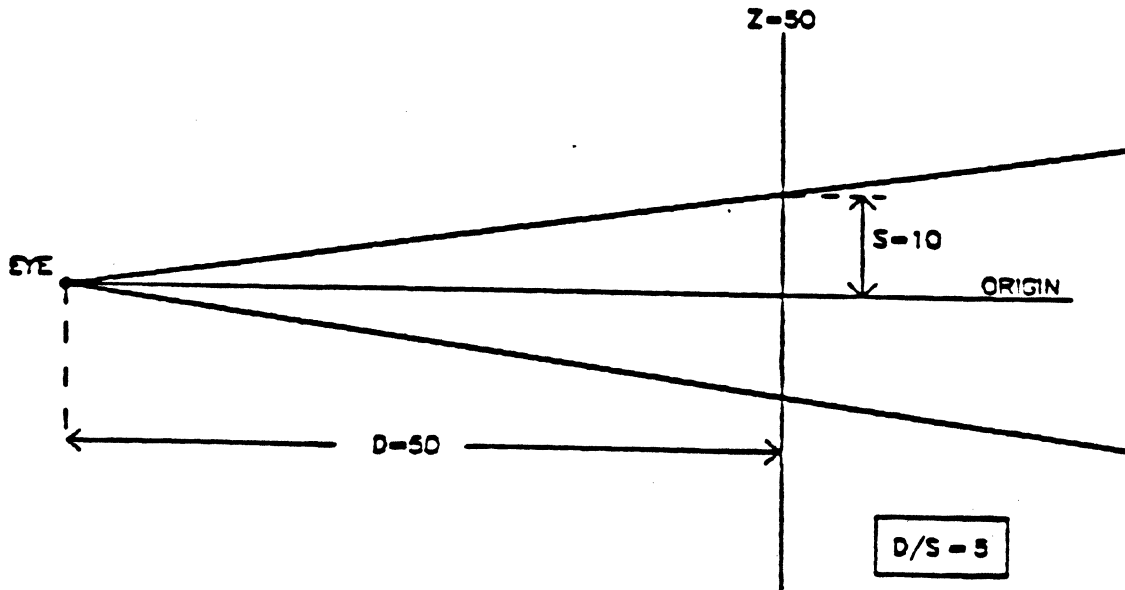


Illustration 4.13B

Normal Effect: figure appears with original perspective.

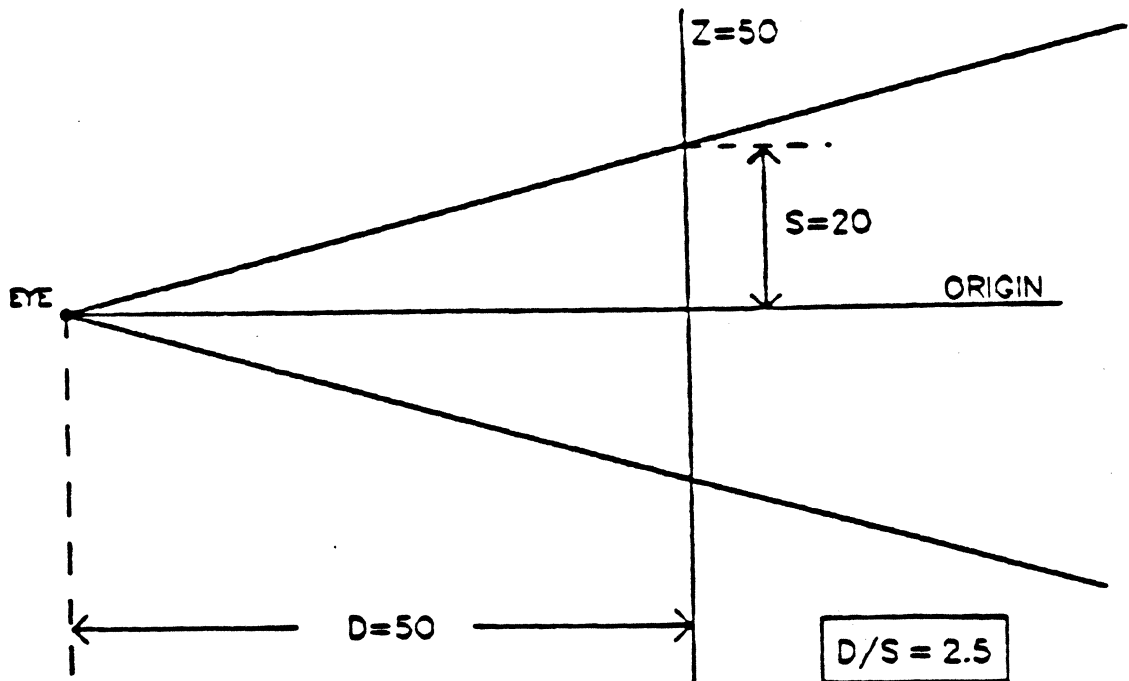
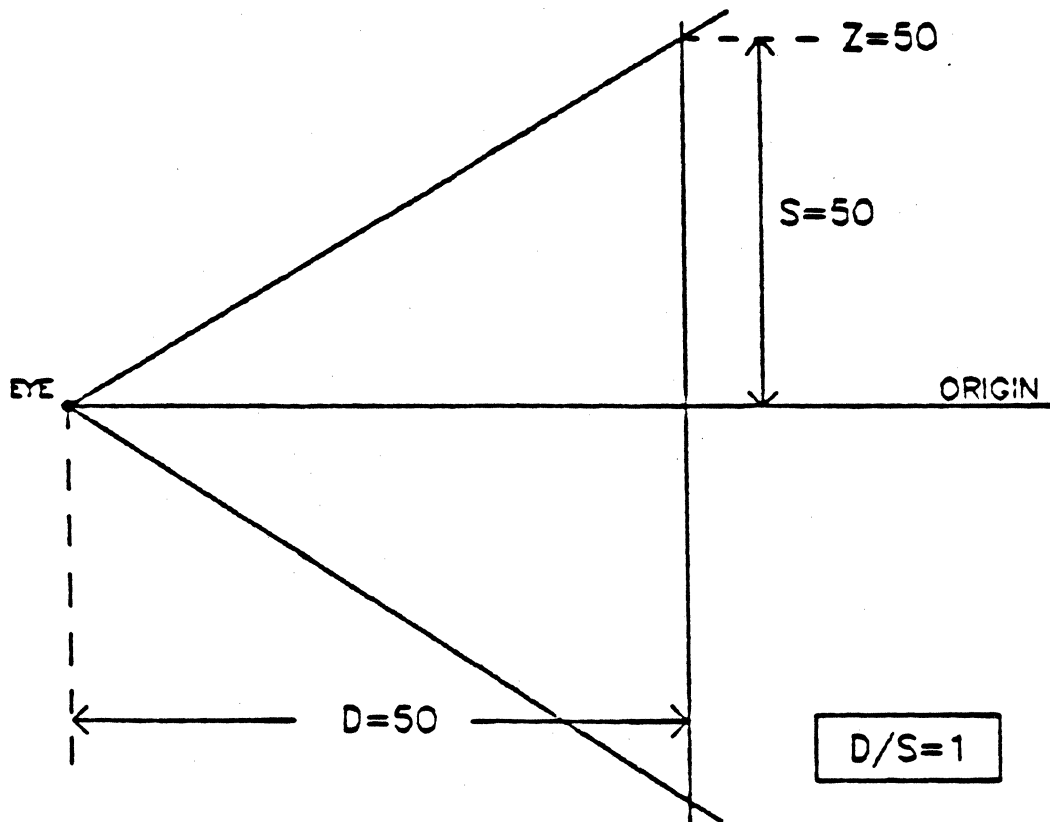


Illustration 4.13C

Wide Angle Effect: figure appears farther away, increasing the apparent perspective.



SQ
RETURN COLOR LOOKUP TABLE TO DEFAULT
COLOR MANIPULATION

PURPOSE: Sets the color lookup table contents to the original, default values.

FORMAT: SQ (no arguments)

REMARKS: This command returns the color lookup table to its default values. The G command does this, but changes other modes and parameters.

SV
SELECT VECTRIX MODE
OPERATING MODE

PURPOSE: Used to switch to Vectrix mode.

FORMAT: SV (no arguments)

REMARKS: The SV command is used to select VX/PC high resolution graphics mode after the VX/PC has been enabled. It is the complement to the SI command.

When using the SV command in a high level program, and using a single monitor system, provisions should be made to exit the program using the SI command. This can be done in two ways.

When an error occurs, a method to exit through the SI command should be provided. A good way to do this in BASIC is the ON ERROR sequence.

Secondly, by setting one of the function keys to call another exit routine, errors in program logic will not cause the system to hang.

EXAMPLE: 5 OPEN "VECTRIX" FOR OUTPUT AS #1
10 ON ERROR GOTO 1000
20 ON KEY(1) GOSUB 1010 : KEY(1) ON
30 PRINT #1,"SV"
.
.
.PROGRAM CODE HERE...
.
1000 PRINT #1,"SI" : PRINT ERR : STOP
1010 INPUT A\$: PRINT #1,"SI" : CLOSE : STOP

The "SI" command in line 1000 returns both a single monitor system and dual monitor system to IBM mode.

NOTE: It is important to understand that the IBM computer and the VX/PC are really two completely separate processors communicating with each other. The SV command does not tell the IBM to recognize the VX/PC. Rather, when the VX/PC has been selected to receive output from the IBM, SV turns on Vectrix high resolution graphics mode.

T
TRANSLATION
3-D COMMAND

PURPOSE: Used to change the orientation of a figure along any of the three axes.

FORMAT: 3-D - TX x 2-D - TX x
 TY y TY y
 TZ z

 x = signed word +-32767 units
 y = signed word +-32767 units
 z = signed word +-32767 units

REMARKS: Translate is used to move objects in either 2-D or 3-D mode. In some cases, a figure will need to be translated to the origin to perform subsequent operations, such as rotation or scaling. In other cases the Translate command is used to move a figure in the X, Y, or Z directions.

When rotating an object, if the object is to spin on its own axis, the object must first be translated to the viewport origin, where X=0, Y=0, and Z=0. It can then be rotated, and translated back to its original position. The amount of translation needed is determined by the distance the point of rotation is from the origin.

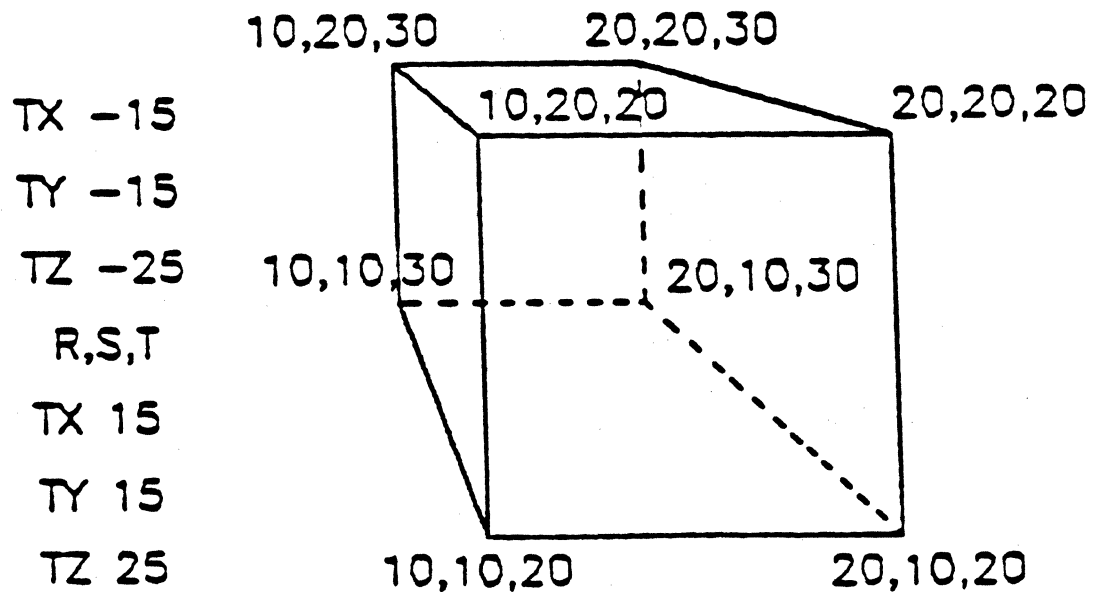
Given a cube of 100 pixels width, starting at X=100, and continuing through to X=200, the command required to rotate the cube around its own center and around the X axis, will be TX -150. The command to translate it back to its original position will be TX 150. Actual movement of the figure is not visible, and the program must redraw the figure after translation.

Another use for translation is to move an object around on the screen. The command "TX 100" will position the drawing point 100 pixels from its current location, in the X direction. If another command is then given to redraw the object, a duplicate of the object will be drawn at the current drawing point. This will duplicate a figure without calculating all the vertices of the object in its new position.

EXAMPLES: This cube can be rotated or scaled on any axis using this series of commands:

Draw cube	- Optional
I	- Initialize Transformation Matrix
TX -15	- Translate X to origin
TY -15	- Translate Y to origin
TZ -25	- Translate Z to origin
Rotate or Scale	- Perform Transformation
TZ 25	- Translate Z to original position
TY 15	- Translate Y to original position
TX 15	- Translate X to original position
Draw cube	- Cube will be transformed

Illustration 4.14 A 3-D Cube



T (continued)

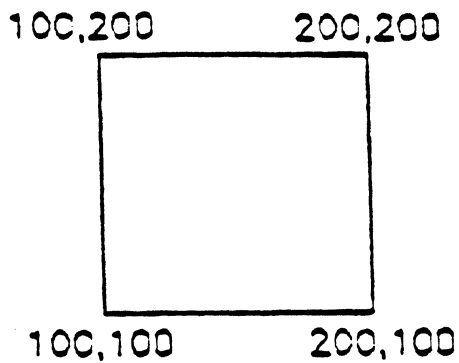
The next illustration is an example of using the Translate command to duplicate a 2-D square at a new set of X and Y coordinates.

The commands used are: Draw Figure

TX 70
TY 50

Draw Figure

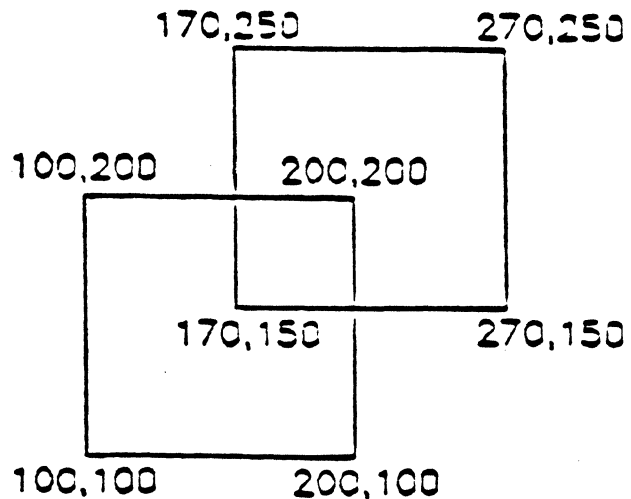
Illustration 4.15 A 2-D Square Translated



TRANSLATION TO
DUPLICATE A FIGURE
AT NEW LOCATION

TX 70

TY 50



TB
TRANSFER BLOCK
PIXEL COMMAND

PURPOSE: Used to copy a rectangular area of an image.

FORMAT: TB xl, xr, yb, yt

xl = x left = unsigned word 0...671 pixels
xr = x right = unsigned word 0...671 pixels
yb = y bottom = unsigned word 0...479 pixels
yt = y top = unsigned word 0...479 pixels

where: xr greater than xl and
yt greater than yb

REMARKS: TB copies a rectangular area of the screen to the location of the current drawing point. The area moved is bounded by the four coordinates given as arguments. The lower left corner of the block (indicated by the screen coordinate xl,yb) is placed at the current drawing point.

The screen coordinates 0...671 on the X axis and 0...479 on the Y axis are the only valid ranges for block origins.

When the block is moved, any part of the moved image falling outside the screen coordinates will be clipped.

When in 3-D mode, the TB command performs a 2-D to 3-D transformation and then moves the block. Other aspects of the third dimension are ignored.

The Bitplane Write Mask and the current color drawing mode will affect the copy of the image. However, the pattern register is ignored.

EXAMPLES: M 300 300 TB 100 200 100 200

Copies the part of an existing image bounded by the coordinates 100,100 and 100,200 and 200,200 and 200,100. The lower left corner of the block is moved to the position of the current drawing point. The copy will occupy the area bounded by the coordinates 300,300 and 300,400 and 400,400 and 400,300.

EXAMPLES: M 300 300 B 511 TB 100 200 100 200

Enables all 9 bitplanes for update. When the block transfer is performed, the colors in the copy remain the same as in the original block.

M 300 300 B 7 TB 100 200 100 200

Bitplanes 1, 2, and 3 are enabled for update and the block is copied. The colors in the copy will reflect the color indicated in the color table index pointed to by the value of the bits in bitplanes 1, 2, and 3 only.

RE TB 100 200 100 200

Selects REPLACE color drawing mode and copies the block. The colors in the original block will overlay (replace) the colors in the new area.

RA TB 100 200 100 200

Selects REPLACE ALL color drawing mode and copies the block. The results are the same as using the RE color drawing mode.

RC TB 100 200 100 200

Selects REPLACE COMPLEMENT color drawing mode and copies the block. The source and destination pixels values are exclusive OR'ed together.

OR TB 100 200 100 200

Selects OR color drawing mode and copies the block. The colors in the original block will mix with the established colors in the new area.

NOTE: The arguments XL, XR, YB, and YT must appear in the correct order.

U
UPLOAD CODE
MISCELLANEOUS COMMAND

PURPOSE: Used to move machine language code into the VX/PC for execution.

FORMAT: U count, number of bytes

count = unsigned word 0...16384
program bytes = bytes

REMARKS: This command provides access to the assembly language command sets of the onboard INTEL 80188 and NEC 7220 processors, and the contents of the VX/PC PROMs.

The U command uploads "count" program bytes of assembly language from the host computer to the VX/PC RAM and executes the program.

The program will be loaded at location 100H, in the 80188 memory space. The maximum length of the assembly routine is 700H. If user defined characters are used, the maximum length is 400H.

Specifying a count of zero without loading any program bytes causes a previously loaded program to be executed again.

This command allows one to write specialized functions. The subroutines used in the VX/PC are available in the jump table in the PROMs.

EXAMPLE: PRINT #1, "U 7 184 255 1 232 54 191 195"

Will upload seven bytes of code, in ASCII format, and clear the screen to white. The ASCII characters are derived from this code:

```
                ORG    100H    ;Program always begins at 100H.
CLEAR           EQU    0C03CH  ;Address of clear routine from
                                ;Advanced Programmers Manual.
0100 B8FF01  START:  MOV    AX,511 ;Color white.
0103 E836BF          CALL  CLEAR  ;clear screen routine.
0106 C3                RET     ;Return to handlers.
                END    START
```

NOTE: Refer to the VX/PC Advanced Programming Manual for PROM entry points.

V
DEFINE VIEWPORT
3-D COMMAND

PURPOSE: Used to change the viewport settings.

FORMAT: V xl, xr, yb, yt

xl = x left = unsigned word 0...671 pixels
xr = x right = unsigned word 0...671 pixels
yb = y bottom = unsigned word 0...479 pixels
yt = y top = unsigned word 0...479 pixels

where: xr greater than xl and
yt greater than yb

REMARKS: V sets the active region of the screen for drawing, defined in screen coordinates. Attempting to draw lines outside the viewport will result in the line being clipped to display only its visible segments.

Viewports are useful for positioning objects in a specific area and not destroying images elsewhere on the screen by overwriting. Several figures may be created and transformed in separate viewports, one at a time, without interfering with other objects displayed in other viewports.

Viewports are available in both 3-D and 2-D modes, but act differently in each. In 2-D mode, viewports act solely as clipping boundaries. However, in 3-D mode, viewports also will change the aspect ratio of objects drawn within them. To set a 1:1 aspect ratio in 3-D mode, a square viewport or the SP (Set Perspective Scaling) command must be used. The default aspect ratio is 4:3.

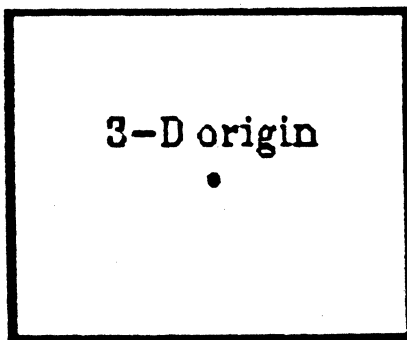
The screen coordinates 0...671 on the X axis and 0...479 on the Y axis are the only valid ranges for viewports. Specifying negative or greater values causes displays crossing the viewport boundaries to wrap around into successive bitplanes, giving unpredictable results.

The default viewport is the full screen, V 0 671 0 479.

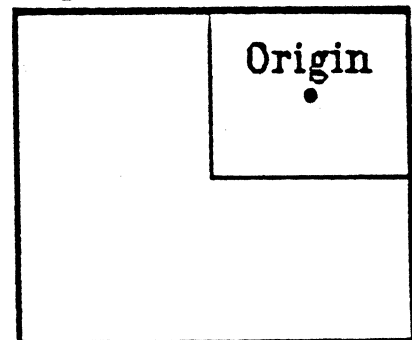
The origin or reference point is also different for each mode. In 2-D, the origin is considered to be the lower left corner, and is assigned the values $X=0$, $Y=0$. In 3-D mode, the origin ($X=0$, $Y=0$, $Z=0$) is the center of the current viewport, which defaults to the center of the screen. As 3-D viewports change, the origin changes.

Illustration 4.16
2-D And 3-D Viewports

Full Screen Viewport



1/4 Screen Viewport



EXAMPLES: V 96 575 0 479
Sets the largest square viewport available.

V 335 670 0 480
Sets the viewport to the right half of the screen.

WA
WEDGE ARC
GRAPHIC PRIMITIVE

PURPOSE: Used to draw an unfilled arc whose endpoints are connected to the arc's point of radius.

FORMAT: WA radius, start angle, arc angle

radius = unsigned word 0...65535 pixels
start angle = signed word +-3600 degree tenths
arc angle = signed word +-3600 degree tenths

REMARKS: The WA command draws an unfilled wedge arc in the current drawing color. The center of the arc wedge is at the CDP. Degrees are specified in tenths of a degree. The CDP is unchanged by this command.

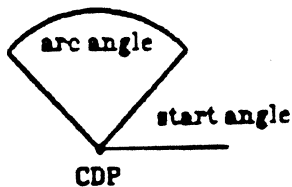
The drawing will proceed from the CDP "radius" pixels along the "start angle", then draw an arc of "arc angle" degrees. The figure will automatically be closed by a line from the end of the arc to the center point.

The perimeter will be drawn in the current drawing color and will be affected by the contents of the pattern register.

EXAMPLE: WA 100 0 900 Draws an unfilled quarter circle of 100 pixel radius, using the CDC.

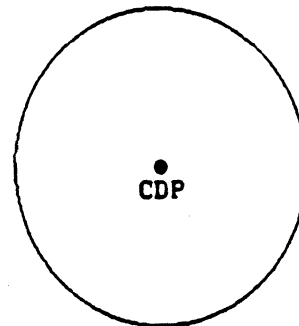
NOTE: Arcs produced in RC color drawing mode may have gaps.

Illustration 4.17 Using The WA Command



WA 100 450 900

radius = 100 pixels
start angle = 45°
arc angle = 90°



WA 100 0 3600
radius = 100 pixels
start angle = 0°
arc angle = 360°

WB
WEDGE ARC, BOUNDARY FILLED
GRAPHIC PRIMITIVE

PURPOSE: Used to draw a wedge arc filled with the current drawing color.

FORMAT: WB fill color, radius, start angle, arc angle

fill color = unsigned word 0...511
radius = signed word +-32767 pixels
start angle = signed word +-3600 degree tenths
arc angle = signed word +-3600 degree tenths

REMARKS: This command is similar to the Wedge Arc command, except that the boundary is drawn in the current drawing color and the wedge is filled with the fill color.

The boundary color must not be the same as the background color or the fill color will ignore the boundary and fill outside the wedge.

If the pattern is not set to all ones, the default, the fill color will leak into the surrounding area.

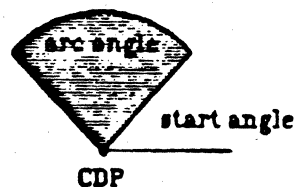
Using an "arc angle" equal to 3600 or -3600 will fill a circle of "radius" pixels centered around the CDP.

EXAMPLES: WB 3 100 0 900 Draws a quarter circle of 100

Draws a quarter circle of 100 pixel radius. The perimeter's color will be the current drawing color, and the figure will be filled with color 3.

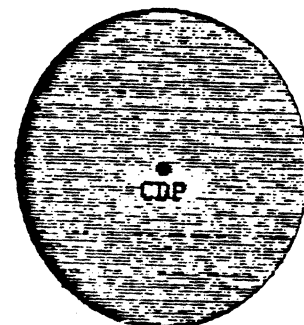
NOTE: Arcs made in RC color drawing mode may have gaps.

Illustration 4.18 Using The WB Command



WB 100 450 900

radius = 100 pixels
start angle = 45°
arc angle = 90°



WB 100 0 3600
radius = 100 pixels
start angle = 0°
arc angle = 360°

WF
WAIT FRAMES
MISCELLANEOUS COMMAND

PURPOSE: Used to create a pause between commands.

FORMAT: WF count

count = unsigned word 0...65535

REMARKS: The WF command causes the VX/PC to pause for "count" video frames. The default setting is 1 (84 frames per second), so there is no waiting.

This command is useful for creating timed effects such as animation. Up to about 12 minutes of delay is possible.

EXAMPLE: WF 10 Causes the frame display to pause 10 frames, or about 1/8 of second between updates.

WNP or @WNP
WRITE PIXELS (encoded)
PIXEL COMMAND

PURPOSE: Used to write encoded pixels to the graphics display memory.

FORMAT: WNP count, color 1, color 2,

count = unsigned word 0...65535 pixels
word 1, ... = unsigned word 0...65535

REMARKS: WNP transfers full or partial images from the host to the display in an encoded format. Starting at the current drawing point, "count" pixels are sent to the display.

Encoded values are sent and the VX/PC decodes these to find the pixel colors to display. "Count" is the number of unencoded pixels, after they are displayed on the screen. For more information on Run Length Encoding, see the System Overview section of this manual.

Each pixel color is decoded from an unsigned 16 bit word value as follows: the nine lowest bits are the color value and the seven high bits represent the pixel repetition count.

The formula for calculating the actual value passed to the VX/PC is:

$$\text{Passed Value} = 512 \times \text{Count} + \text{Color}.$$

A "count" of 0 moves the current drawing position to the upper left corner and the VX/PC will expect an entire screen of pixels to be passed.

Values are transmitted in the current transmission mode, ASCII or hex. When ASCII mode is active, values should be terminated by a carriage return or other valid delimiter. When hex mode is active, the color values are passed as 16 bits with no carriage return or other delimiters between successive values.

The VX/PC has nine bits of color information for each pixel, organized as one pixel per plane. The Bitplane Write Mask Register (controlled by the B command), allows selective use of bitplanes. There are two versions of the WNP command - one ignores the Bitplane Write Mask, the second utilizes it.

The first version is simply WNP. All bits for a pixel are transferred at one time; the bitplane mask register is ignored.

@WNP, the second version of this command, writes only the bitplanes enabled by the B command. If a B511 command has been given, enabling all bitplanes, the effect will be the same as using the WNP command without the '@'.

EXAMPLES: M 0 240 WNP 10 2050 3076

Starting at X=0, Y=240, writes 4 pixels with color #2 and then six pixels with color #4. Color #2 is indicated by 2050 ($512 \times 4 + 2$). Color #4 is indicated by 3078 ($512 \times 6 + 4$).

B 255 WNP 1 1023

Enables bitplanes 1-8 (disabling bitplane 9 - the pattern is 011 111 111), decodes the word argument as 1 byte of color #511 ($1023 - (512 \times 1) = 511$), and writes the pixel at the current drawing point. The color table index of the pixel is #511.

B 255 @WNP 1 1023 (or B 255 @WNP 1 767)

Enables bitplanes 1-8 (disabling bitplane 9 - the pattern is 011 111 111), decodes the word argument as 1 byte of color #511 ($1023 - (512 \times 1) = 511$ or $767 - (512 \times 1) = 511$), and writes the pixel at the current drawing point. The color table index value of the pixel will be 255 because the bitplane mask register is used and the bit #9 cannot be written.

WNP or @WNP (continued)

M 100 100 B258 @WNP 1 514

Moves the current drawing point to coordinate X=100, Y=100. Then the bitplane mask register is set to 258 (pattern: 100 000 010). One pixel is now written (pattern: 000 000 111). The bits zeroed in the bitplane mask register are ignored, making possible layered images.

NOTES: Images transfer slightly faster in Flash mode.

This command is usually given to restore a screen previously saved with the RNP command.

For more information on specialized uses of this command, see Chapter 2, Section 2.8, "Pixel and RAM Commands".

WNR
WRITE GRAPHICS RAM (encoded)
RAM COMMAND

PURPOSE: Used to restore graphics RAM previously encoded with the RNR command.

FORMAT: WNR bitplane, count, word 1, word 2,...

bitplane = byte value 1...9
count = unsigned word 0...20160 words
word 1, ... = unsigned word 0...65535 words

REMARKS: WNR transfers full or partial images from the host computer to the display in an encoded format, one bitplane at a time. Starting at the current drawing point, "count" words are sent to the display. This command is usually used to restore an image after it has been encoded and transferred to the host with the command RNR.

Encoded values are sent and the VX/PC decodes these to find the values to write to graphics memory. "Count" is the number of unencoded words after they are displayed on the screen. For more information on Run Length Encoding, see Chapter 2, Section 2.8, "Pixel And RAM Commands".

Only words of "0000" and "FFFF" are decoded. Any other word is moved as is into the graphics memory.

A "count" of 0 moves the current drawing position to the upper left corner and the VX/PC will expect an entire bitplane of pixels to be passed.

Values are transmitted in the current transmission mode, ASCII or hex. When ASCII mode is active, values should be terminated by a carriage return or other valid delimiter. When hex mode is active, the color values are passed as 16 bits with no carriage return or other delimiter between successive values.

Writing starts at the beginning of the 16 pixel word in which the current drawing point is located.

WNR (continued)

EXAMPLE: WNR 1 0 06842 00000 06510 00000 00000.....

Restores bitplane 1 from a file of encoded data.

NOTES: Images transfer slightly faster in Flash mode.

This command is usually given to restore a screen previously saved with the RNR command.

For more information on specialized uses of this command, see Chapter 2, Section 2.8, "Pixel and RAM Commands".

PURPOSE: Used to directly manipulate screen pixels and to restore images previously saved with the RP command.

FORMAT: WP count, color 1, color 2,

count = unsigned word 0...65535 pixels
color 1, ... = unsigned word 0...511

REMARKS: This command transfers full or partial images from the host computer to the display, one pixel at a time, starting at the current drawing point. "Count" pixels are passed; their colors are specified as color1, color2, etc. Values are transmitted in the current transmission mode, ASCII or hex. The bitplane mask register is ignored.

A "count" of 0 moves the current drawing position to the upper left corner and transfers 322,560 words or 645,120 bytes, which may be impractical. If an entire screen is to be transferred, the encoded version of this command (WNP) can be used.

If the pixel "count" exceeds the right edge of the screen, reading will continue on the next lower pixel row, starting at the left.

The VX/PC has nine bits of color information for each pixel, organized as one pixel per plane. The Bitplane Write Mask Register (controlled by the B command), allows updating selected bitplanes. There are two versions of the RP command - one ignores the Bitplane Write Mask, the second utilizes it.

The first version is simply WP. All bits for a pixel are sent at one time; the bitplane mask register is ignored.

@WP, the second version of this command, writes only the bitplanes enabled by the B command. If a B511 command has been given, enabling all bitplanes, the effect will be the same as using the WP command without the '@'.

WP or @WP (continued)

EXAMPLES: WP 5 7 7 7 7 7 Displays five pixels of color 7 at the current drawing point.

WP 5 1 2 3 4 5 Draws five pixels on the screen, in colors 1...5.

B 255 WP 1 511

Enables bitplanes 1-8 (disabling bitplane 9 - the pattern is 011 111 111), and writes a pixel at the current drawing point. If the bit pattern of the color table index used is 111 111 111 (511 in decimal, white in the default color lookup table), the pixel will be drawn in white.

B 255 @WP 1 511

Enables bitplanes 1-8 (disabling bitplane 9 - the pattern is 011 111 111), and writes the pixel at the current drawing point. If the bit pattern of this pixel was 111 111 111 (511 in decimal, white in the default color lookup table), the resulting color table index will be 255. The bitplane mask allows only the first 8 bitplanes to be written when the symbol '@' is appended to the WNP command. Any bits not enabled are not affected.

B258 @WP 1 7

The bitplane mask register is set to 258 (pattern: 100 000 010). One pixel is then written (pattern: 000 000 111) to the screen. The color table index for this pixel is 2 - (the pattern of two is 000 000 010).

NOTES: Images transfer faster in Flash Mode (KF).

This command automatically uses the RE color drawing mode. However, the current drawing mode is left unchanged on exit.

For more information on specialized uses of this command, see Chapter 2, Section 2.8, "Pixel and RAM Commands".

WR
WRITE GRAPHICS RAM
RAM COMMAND

PURPOSE: Used to directly manipulate graphics RAM memory and to restore images previously saved with the RR command.

FORMAT: WR bitplane, count, word 1, word 2, ...

bitplane	=	byte value	1...9
count	=	unsigned word	0...65535 words
word 1,	=	unsigned word	0...65535 words

REMARKS: WR transfers full or partial images from the host computer directly to graphics RAM, sixteen bits at a time, to only the specified bitplane. Starting at the beginning of the word containing the current drawing point, "count" words are displayed on the VX/PC. Values are transmitted in the current transmission mode, ASCII or hex.

A "count" of 0 moves the current drawing position to the upper left corner and transfers an entire bitplane, which is 20,160 words or 40,320 bytes in hex mode. For long transfers, the encoded version of this command (WNR), can be used.

If the "count" exceeds the right edge of the screen, reading will continue on the next lower row, starting at the left.

The WR command can transfer images in either DMA or in normal (non-DMA) mode. DMA mode transfers data considerably faster than non-DMA mode. Combining VX/PC DMA mode, IBM DMA, and hex mode is the fastest method of transfer. An entire image can be transferred in as little as 1.5 seconds.

When not using DMA mode to transfer images, the WNR command transfers data faster.

WR (continued)

EXAMPLE: WR 1 3 3634 4897 32766

Writes three words (48 pixels) to bitplane 1.

NOTES: The low order bit of each word appears leftmost on the screen.

For more information on specialized uses of this command, see Chapter 2, Section 2.8, "Pixel and RAM Commands".

For more information on setting up host DMA transfers, see Appendix B, "Using Direct Memory Access".

XB
COMPLEX BOUNDARY FILL
GRAPHIC PRIMITIVE

PURPOSE: Used to fill an area, defined by a given color, with the current drawing color.

FORMAT: XB boundary color

boundary color = unsigned word 0...511

REMARKS: XB fills the region containing the current drawing point with the current drawing color. The fill stops stop at the point where it contacts either the "boundary color" or other locations of the current drawing color. The entire screen will be filled if the area to be filled is not completely surrounded by either the "boundary color" or the current drawing color.

This command draws in REPLACE (RE) color mode but restores the current mode after drawing.

Sending a control-E (hex 05) to the VX/PC while the XB command is executing will abort the fill.

The edges of the screen are considered fill boundaries.

EXAMPLES: XB 3 Fills an area with the current drawing color until either an area containing the "boundary color" (#3) or the current drawing color is found.

XF
COMPLEX FLOOD FILL
GRAPHIC PRIMITIVE

PURPOSE: Used to fill an area with the current drawing color.

FORMAT: XF (no arguments)

REMARKS: XF fills regions contiguous to the current drawing point and containing the same color as the location of the current drawing point. The fill continues in all directions until another color is found.

Like the XB command, the area to be filled must be bounded by another color or the entire screen will be flooded.

This command performs its drawing in the REPLACE (RE) mode but restores the current drawing mode after drawing.

Sending a control-E (hexadecimal 05) to the VX/PC while the XF command is executing will abort the fill.

The pattern register is ignored during fills, so the fill will always be a solid color.

The edges of the screen are considered fill boundaries.

XHC
SET CROSSHAIR POSITION TO CDP
CROSSHAIR COMMAND

PURPOSE: Used to move the crosshair to the current drawing point.

FORMAT: XHC (no arguments)

REMARKS: XHC sets the center of the crosshair to the position of the current drawing point. The size and visibility of the crosshair are not affected.

EXAMPLE: M100 100 XHC Sets the current drawing point (CDP) to X = 100, Y = 100. The XHC command then positions the crosshair at that coordinate.

XHF
CROSSHAIR OFF
CROSSHAIR COMMAND

PURPOSE: Turns the crosshair off.

FORMAT: XHF (no arguments)

XHF turns a crosshair off, making it invisible. The crosshair color or colors will be complemented so that the area covered by the crosshair returns to the original color the crosshair covered. Crosshair size and position are not affected.

When a crosshair color is complemented, all bits of the color table index number from which the color is chosen are reversed. Zeroes become ones and ones become zeroes. The red, green, and blue color components within the color table entry referenced are not changed.

This command should be used before any drawing or erasing on the screen. If the crosshair is left on, it may be destroyed or made invisible by color or drawing changes. For instance, using the command E7 to clear the screen to color #7 without turning the crosshair off will erase the crosshair from the screen. Later, when the crosshair is turned off with the XHF command, the crosshair will actually be redrawn.

The general procedure is to turn the crosshair off, draw or erase as required, and turn the crosshair back on. It will then be the complement of the colors it covers and will be positioned correctly.

XHN
CROSSHAIR ON
CROSSHAIR COMMAND

PURPOSE: Turns the crosshair on.

FORMAT: XHN (no arguments)

REMARKS: XHN turns the crosshair on, making it visible. The size size and position of the crosshair are not affected. The color or colors of the crosshair will be the complement of the color or colors it covers. All zeroes and ones in the binary equivalent of the color table index number will be reversed, producing a new color.

The default crosshair is full screen; the intersection at the center, coordinates X = 335, Y = 240.

Crosshairs are clipped to the current viewport boundaries in both 2-D and 3-D modes.

Only bits within enabled bitplanes are complemented when covered by the crosshair.

EXAMPLE: XHN Draws a crosshair at the last position established by the XHP command. The color of the crosshair is determined by complementing the color of the area it covers.

NOTE: Changing the contents of the color lookup table can affect the visibility of crosshairs. In the above example, if the contents of color table entry #504 are changed to a color approaching the color in table entry #7, the crosshair may not be visible.

XHP
SET CROSSHAIR POSITION
CROSSHAIR COMMAND

PURPOSE: Used to change the crosshair position.

FORMAT: XHP x, y

x = unsigned word 0...671 (coordinate)
y = unsigned word 0...479 (coordinate)

REMARKS: XHP positions the crosshair center at physical screen coordinates X,Y. The default positions at power up are: X = 336 and Y = 240, the screen center.

This command does not affect crosshair size or visibility. Positioning will occur whether or not the crosshair is visible.

The commands G and G0 do not change crosshair size or position. Neither does hardware RESET.

To set the crosshairs on when in 3-D mode, or to find the position of an object while in 3-D mode, these steps should be performed:

- 1) Use the RD command to return the 2-D coordinates of a point.
- 2) Use the XHP command to position the crosshair to the coordinates returned by the RD command.

EXAMPLE: XHP 200 200 Sets the crosshair at X=200, Y=200. It will only be visible if the XHN command has been given.

XHR
RETURN CROSSHAIR POSITION
CROSSHAIR COMMAND

PURPOSE: Used to transfer the crosshair position to the host computer.

FORMAT: XHR (no arguments)

Returns: x = unsigned word 0...671 (coordinate)
y = unsigned word 0...479 (coordinate)

REMARKS: This command queries the graphics processor for the current crosshair position. The values returned to the host computer are in physical screen units (X = 0 to 671, Y = 0 to 479). The command does not affect crosshair size, position, or visibility. The values are returned with no concern for crosshair visibility.

In ASCII mode, the two coordinates will be returned as 5 digit word values, padded to the left with zeroes as necessary. Each word will be delimited by a carriage return.

In hex mode, the two coordinates will be returned as unsigned word values, a series of sixteen binary bits.

EXAMPLE: XHR When the intersection of the two lines of the crosshair are at screen center, X=335 and Y=240, the values "00335" and "00240" will be returned in ASCII mode.

XHS
SET CROSSHAIR SIZE
CROSSHAIR COMMAND

PURPOSE: Used to set width and height of crosshair.

FORMAT: XHS width (also height)
width = unsigned word 0...671 pixels

REMARKS: XHS sets the total pixel width and height of the crosshair. Width and height are in physical screen coordinates (X = 0 to 671, Y = 0 to 479). The power up default is full screen in both horizontal and vertical directions.

Values for "width" of 0 and 480 and greater will also produce a full screen crosshair. Other values produce a crosshair with width and height equal.

EXAMPLE: XHS 10 Draws a ten pixel high and ten pixel wide crosshair centered at the point where the two lines of the current crosshair intersect. The color will be the background color complemented.

Z
ZOOM VIDEO IMAGE
VIDEO COMMAND

PURPOSE: Enlarges selected areas of the screen.

FORMAT: Z factor
factor = 1...16

REMARKS: Zoom is used, usually in conjunction with PAN, to enlarge specific areas of an image. Generally the area to enlarge is first selected by using PAN to move the area to the bottom left of the screen. Then the Zoom command is given along with a factor from 1 to 16.

The factor determines the magnification that will be displayed. A factor of 1 will not change the original image (unless already Zoomed), while a factor of 16 will replicate every pixel 16 times. During this expansion process, much of the original area will be "pushed" off the screen and clipped.

The image residing in graphics RAM remains unchanged by the Zoom process. Because the actual graphics memory is not changed, an attempt to print only the enlarged area will print the entire screen. Similarly, saving the image while it is Zoomed will save the entire image, not just the Zoomed area.

Zoom is useful for enlarging an area to be touched up. At maximum Zoom, a single pixel on the screen is quite large (16 scan lines high and 16 original pixels across). At this size, it is possible to change a single pixel from one color to another, perhaps to blend it into the background color.

EXAMPLES: Z 16

Expands the area nearest the lower left corner by a factor of 16. Approximately 1/16 of the original image will be visible after the command executes.

PAN 336 240 Z 5

This command combination first moves the middle of the original screen to the lower left. Then the area surrounding the lower left screen corner is expanded on a 5 pixel for one basis.

APPENDIX A

THE VX/PC COLOR TABLE

APPLICATION NOTE #13

This APPLICATION NOTE introduces the novice and the experienced computer graphics user to the flexible color table in the Vectrix VX/PC color graphics processor.

Several programs are included to aid in gaining an in-depth understanding of the color table and its varied uses. Included programs cover listing the default values in the table, displaying entire default or custom tables, and rearranging the table to display colors sequentially by shades. The programs are coded in IBM PC BASICA.

For more information on uses for the color table, refer to the Chapter 2, "SYSTEM OVERVIEW", Sections 2.3.2 and 2.3.3. Also check these individual commands in Chapter 4, "REFERENCE GUIDE": C, E, Q, RQ, and SQ.

THE VX/PC COLOR TABLE

INTRODUCTION

The user selects the colors generated by the VX/PC graphics processor through a color lookup table. This table not only allows selection of a color contained in the table, but also the definition of a new color for any table entry. When the VX/PC is powered on, the color table contains 512 predefined default colors. This allows the user to begin rapidly generating color graphics. As the need arises, the table can be customized to provide many different renditions or combinations of colors.

There are two versions of the VX/PC card sets available: the standard model with a 4096 color palette, and the enhanced version with a 16.8 million color palette. Both versions can simultaneously display 512 colors; the palette is the total number of different colors displayable.

The color tables for the two systems are organized the same. The one difference is that the standard version masks off the lower four bits of the values used to indicate the red, green, and blue contributions to a color. When the optional 16.8 million color card is installed, the system no longer masks off the lowest four bits, so 256^3 colors (0-255 red, 0-255 green, and 0-255 blue) are available.

COLOR TABLE ORGANIZATION

16.8 Million Color Version

The 16.8 million color table consists of 512 entries, each entry containing three bytes defining the numeric contribution of red, green, and blue to the color displayed on the monitor. By varying the values of each byte from 0 to 255 in any table entry, the user has complete control over the relative intensity of each of the primary colors. Selecting a specific color is as simple as selecting the table entry where any particular combination of red, green, and blue resides.

The table entries are numbered from 0 to 511. In the default table, the first entry contains the three bytes 0,0,0 which will produce the color (or lack of color) black. Entry #7 contains the three bytes 255,31,31. This will print red at full intensity with near minimum dilution by either green or blue. Entry #56 contains 31, 255, 31 which produces a bright green. The purest blue in the default color table, entry #488, contains 31,31,255.

Table entry number 511 contains the maximum value in each byte (255,255,255) which produces an even mixture of each of the primary colors, resulting in white.

The following program prints the default values for each entry in the color table, showing the relative amounts of red, green, and blue.

```

10 LPRINT "SLOT #","RED","GREEN","BLUE"           'Print headings
20 LET S = -1                                       'Table entry count
30 FOR B = 31 TO 255 STEP 32                       'Blue varies slowest
40 FOR G = 31 TO 255 STEP 32                       'Next is green
50 FOR R = 31 TO 255 STEP 32                       'Red is fastest
60 LET S = S + 1                                    'Increment entry #
70 IF S = 0 THEN LPRINT 0,0,0,0 : GOTO 90          'Entry 0 always 0,0,0
80 LPRINT S,R,G,B                                   'Print actual values
90 NEXT R : NEXT G : NEXT B

```

4096 Color Version

Users of the 4096 color version of the VX/PC will notice one difference in the color table. The lower four bits of each color table component are ignored. Hence, if the red value is 31, the system effectively uses 16. The maximum value for any entry is 240, not 255. In effect, the red, green, and blue values increment in steps of 16 instead of 1. Hence there are 16^3 or 4096 possible colors.

Even though color table values are interpreted as multiples of 16, they can actually contain any value from 0 to 255. This is provided to assure upward compatibility with the enhanced color table.

Examples of this process are given next:

```

Values between 0 and 15 are interpreted as 0
Values between 16 and 31 are interpreted as 16
.....
Values between 224 and 239 are interpreted as 224
Values between 240 and 255 are interpreted as 240

```


Next is a table of common colors, their position in the default color table, and the byte values assigned in the respective positions. The columns labeled ACTUAL BYTE VALUES indicate the actual red, green, and blue component values in the default table. The columns labeled MASKED BYTE VALUES show the actual values with the four lowest bits masked off.

COLOR	TABLE ENTRY	ACTUAL BYTE VALUES			MASKED BYTE VALUES		
		RED	GREEN	BLUE	RED	GREEN	BLUE
BLACK	0	0	0	0	0	0	0
RED	7	255	31	31	240	16	16
ORANGE	39	255	159	31	240	144	16
GREEN	56	31	255	31	16	240	16
YELLOW	63	255	255	31	240	240	16
TAN	93	191	127	63	176	112	48
BEIGE	247	255	223	127	240	208	112
PURPLE	268	159	63	159	144	48	144
PINK	295	255	159	159	240	144	144
BLUE	448	31	31	255	16	16	240
LAVENDER	487	255	159	255	240	144	240
CYAN	504	31	255	255	16	240	240
WHITE	511	255	255	255	240	240	240

A way to find specific colors is to use the E command. This sets the background color so you can quickly see if a color number is acceptable. For example, "E7" changes the entire screen to red, while "E504" changes the background to cyan.

DISPLAYING THE COLOR TABLE

The following program will initialize the color table to the default values and display them, numbering the colors.

The next example programs can be used with either the standard or the enhanced color tables. The code will not change between the two systems, nor will the colors. The shading and intensity of colors may vary slightly.

```

10 OPEN "VECTRIX" FOR OUTPUT AS #1
20 PRINT #1, "SV G RE E511" 'Initialize
30 PRINT #1, "C0 JM3 M160 457 $VX/PC COLOR TABLE" 'Print text
40 PRINT #1, "C0 JM2 M238 20 $DEFAULT COLORS" '
50 C = 0
100 PRINT #1, "P4 16 58 16 409 656 409 656 58" 'Draw borders
110 FOR X = 36 TO 636 STEP 20 'Draw grid
120 FOR Y = 58 TO 249 STEP 191
130 PRINT #1, "M";X;" ";Y;" L";X;" ";Y + 160
140 NEXT Y : NEXT X
160 FOR Y = 78 TO 218 STEP 20
170 PRINT #1, "M16 ";Y;" L656 ";Y
180 PRINT #1, "M16 ";Y + 171;" L 656 ";Y + 171
190 NEXT Y
200 FOR X = 17 TO 637 STEP 20 'Display colors
210 FOR Y = 59 TO 199 STEP 20 '0-255
220 PRINT #1, "C";C;" M";X;" ";Y;" RF 19 19"
230 C = C + 1
240 NEXT Y : NEXT X
260 FOR X = 17 TO 637 STEP 20
270 FOR Y = 250 TO 390 STEP 20 'Display colors
280 PRINT #1, "C";C;" M";X;" ";Y;" RF 19 19" '256-511
285 C = C + 1
290 NEXT Y : NEXT X
300 C = 0 : Y = 61
310 PRINT #1, "C0"
320 FOR X = 15 TO 650 STEP 20 'Print horizontal
330 PRINT #1, "M";X;" ";Y;" JM 1 JS 5 7 12 $";C 'entry numbers
340 C = C + 8
350 NEXT X
360 IF Y = 61 THEN LET Y = 251 : GOTO 320
400 X = 1 : B = 75 : GOSUB 500 'Print vertical
410 X = 657 : GOSUB 500 'entry numbers
420 B = 265 : GOSUB 500
430 X = 1 : GOSUB 500
440 INPUT A$
450 PRINT #1, "SI"
460 CLOSE : END
500 FOR N = 0 TO 7
510 Y = B + N * 20 'Calculate position
520 PRINT #1, "M";X;" ";Y;" JM 1 $";N 'and display
530 NEXT N : RETURN

```

NOTE: If the command G is removed from statement #10, the color table will not be initialized when this program is run. Hence, whatever color table is resident will be displayed.

WHY CHANGE THE COLOR TABLE?

For certain applications, the user may need to precisely manipulate the color table to achieve special effects to display depth cues and surfacing, light models and shading, color balancing, and for visual preference. Typical applications for these techniques include solids modelling, animation, paint programs, and graphics arts.

CHANGING COLOR TABLE ENTRIES

The "Q" command allows the user to change any entry in the table to any value. The correct format is:

```
"Q i n r1 g1 b1 r2 g2 b2..."
```

The "i" parameter indicates the first table entry to change. "n" indicates the total number of table entries to be changed, with "n" ranging from 1 to 512. The remaining values are the actual values to be inserted in each consecutive table entry for "n" entries. To change entry #56 (pure green) to pure red, "Q 56 1 255 0 0" would be entered. Entering "Q 1 3 255 0 0 0 255 0 0 0 255" will change entries 1,2, and 3 to pure red, green, and blue respectively.

PROGRAM EXAMPLES TO LOAD NEW TABLES

Greys are produced by mixing the three colors in even amounts: e.g. 100,100,100 or 192,192,192. The higher the numbers, the brighter the shade of grey. This program will load a grey scale of 256 entries, ranging from black in entry #0 to white in entry #255:

```
10 OPEN "VECTRIX" FOR OUTPUT AS #1
20 PRINT #1, "Q0,256"           'Change 256 entries
30 FOR I = 0 TO 255           '
40 PRINT #1, I,I,I           'Set R,G, and B to I
50 NEXT I                     '
60 INPUT AS
70 PRINT #1, "SI"
80 CLOSE : END
```

It should be noted that only the first 256 entries have been changed, leaving the second 256 as they were.

The next program demonstrates another way to load a color table. The first 100 table entries will be shades of red, the next 100 entries will be shades of green, and the next 100 will be shades of blue. It will then display each shade on the screen. Table entries 0 and 301 through 511 remain the same as they were in the default table.

```

10 OPEN "VECTRIX" FOR OUTPUT AS #1
20 PRINT #1, "SV G EØ RE" 'Initialize to black
30 LET A$ = "KR LØ 102 L602 Ø LØ -102 L-602 Ø KA"
40 LET M$ = "LOADING 100 SHADES OF "
50 LET M1$ = "COLOR TABLE ENTRIES "
60 PRINT #1, "C511 M34 334";A$
70 PRINT #1, "M34 189";A$ 'Print white borders
80 PRINT #1, "M34 44";A$
100 PRINT #1, "C511 M40 455 JM2$";M$;"RED"
110 PRINT #1, "Q 1 100" 'Load red shades int
120 FOR R = 56 TO 254 STEP 2 'entries 1-100
130 PRINT #1, R;" Ø Ø"
140 NEXT R
200 PRINT #1, "C511 M40 310 JM2$";M$;"GREEN"
210 PRINT #1, "Q101 100" 'Load green shades
220 FOR G = 56 TO 254 STEP 2 'entries 101-200
230 PRINT #1, "Ø ";G;" Ø"
240 NEXT G
300 PRINT #1, "C511 M40 165 JM2$";M$;"BLUE"
310 PRINT #1, "Q201 100" 'Load blue shades
320 FOR B = 56 TO 254 STEP 2 'entries 201-300
330 PRINT #1, "Ø Ø ";B
340 NEXT B
400 REM - DISPLAY NEW TABLE
410 PRINT #1, "CØ M40 455 RF 400 -15" 'Clear text
420 PRINT #1, "C100 JM2$";M1$;"1 THROUGH 100"
430 C = Ø : Y = 335 : GOSUB 800 'Display reds
440 PRINT #1, "CØ M40 310 RF 400 -15" 'Clear text
450 PRINT #1, "C200 JM2$";M1$;"101 THROUGH 200"
460 C = 100 : Y = 190 : GOSUB 800 'Display greens
470 PRINT #1, "CØ M40 165 RF 400 -15" 'Clear text
480 PRINT #1, "C300 JM2$";M1$;"201 THROUGH 300"
490 C = 200 : Y = 45 : GOSUB 800 'Display blues
500 INPUT A$
510 PRINT #1, "SI"
520 CLOSE : END
800 FOR X = 35 TO 629 STEP 6 'Loop to determine
810 C = C + 1 'color table entries
820 PRINT #1, "M";X;" ";Y;"C";C;"RF 5 100" 'and to print color
830 NEXT X 'bars
840 RETURN

```

Loading the color table with sequential values can be useful when many shades and intensities of a color are needed in an image.

APPENDIX A

MANAGING THE VX/PC BITPLANES

APPLICATION NOTE #14

APPLICATION NOTE #14, MANAGING THE VX/PC BITPLANES, provides an introduction to the bitplane architecture of the Vectrix VX/PC graphics processors. Included is information on drawing multiple images in graphics memory and selectively displaying them. Also included is an algorithm used to calculate color table set up and a program that uses this algorithm to hide two images in memory while displaying a third.

For more information, refer to the individual commands used, including B, Q, and the color commands in Chapter 4, "REFERENCE GUIDE".

MANAGING THE VX/PC BITPLANES

INTRODUCTION

The VX/PC has a flexible bit plane architecture which readily supports image separation and layering. By selecting specific bit planes, images can be generated and maintained in separate memory partitions. Then, using bit plane management techniques, selected images can be made visible (in the foreground), while others remain non-visible (in the background). These techniques can be used to buffer the display of one image while another is being created, allowing a smooth and instantaneous transition between images.

Understanding the working of the bitplanes can be enhanced by typing and running the last program in this Application Note, the Layered Drawing Demo, before reading the rest of the Note.

BIT PLANE BASICS

Every pixel on the screen monitor has nine associated bits of color information. Conceptually, these nine bits are organized in memory as bitplanes where each plane contains one bit for every possible screen coordinate. The nine bits yield a value that is an index to one of the 512 entries in the color table.

When the VX/PC is powered up, the Bit Plane Write Mask (BPWM) allows data to be written to all nine bit planes. However, certain applications require the planes to be divided into logical sets, so that an image can be drawn in one set while an image in another set is not affected. For example, to have three separate images in memory, the bit planes could be divided logically into three sets: planes 1, 2 and 3; 4, 5, and 6; and 7, 8, and 9. Another way the planes can be organized is four planes for image #1, and five planes for image #2. Five separate images can be manipulated by successively enabling planes 1-2, 3-4, 5-6, 7-8, and 9.

DRAWING MULTIPLE IMAGES

The general procedure for creating multiple images is this:

1. Decide on the number of colors needed for each image.
2. Allocate and select logical bit plane sets
3. Load color table for the first set.
4. Create image.
5. Select next set to write.
6. Load color table.
7. Create image.

Deciding the Number of Colors Possible

The user must consider the number of colors each image requires when dividing up the planes. The default situation, with the BPWM containing all ones, allows 2^9 (512) colors. When the planes are divided into a 3 x 3 x 3 arrangement, each image can use 2^3 (8) separate colors. When only one bit plane is enabled for writing, only 2^1 (2) colors can be displayed.

Allocating Logical Bit Plane Sets

Once the number of colors required and the division of the planes has been decided, the BPWM must be set to enable the selected planes for writing. All that is needed is to set the BPWM to the decimal equivalent of a binary value that represents the planes to be enabled. A binary one enables writing to a plane, while a binary zero in a particular bit position disables writing to that plane.

Bit plane number	1	2	3	4	5	6	7	8	9
Decimal value	1	2	4	8	16	32	64	128	256

When the VX/PC is turned on, the BPWM contains all ones, allowing all planes to be updated. To allow only the first three planes to accept data, the user would give the command "B7" ($1+2+4=7$), setting the first three planes "on" and the rest "off". To enable planes 7, 8, and 9, the command is "B448" ($64+128+256=448$). "B8" selects plane #4, while "B256" selects only plane #9.

Loading the Color Table

Bit plane manipulation requires loading the color table each time a new set of layers are selected for display. When a 3 x 3 x 3 bit scheme is desirable, 8 colors are available for each set of planes. Once selected, these colors must be loaded repetitively into all entries in the color table.

Every write to the screen involves two associated processes. The first is writing to the enabled bit planes to represent the selected color. When planes 1,2, and 3 are enabled, the write to the bitplanes can only occur in these bits, not allowing any image in the other planes to be changed. Secondly, the system reads the value of all nine bitplanes of the affected pixels, without concern for the state of the BPWM. This value is an index to the color table; the color is then displayed on the screen at all the pixels chosen.

When a 3 x 3 x 3 arrangement has been selected and the command "B7" has been given, only the first three planes can be written. The other bits are effectively "don't cares". This can be represented by:

```
Command "B7"
Color = 1
Bit plane number      1      2      3      4      5      6      7      8      9
Bit plane contents    1      0      0      x      x      x      x      x      x
Color table index = 1
```

Using the first three bits for an image allows eight possible colors, with the bit values ranging from all zeroes (0) to all ones (7). If color table entry #1 is selected as the current color, ones would be inserted into bit #1, zeroes into bits #2 and #3. The other planes are not enabled and cannot be written. If all "x" bits are zeroes, the pixel value will be 1, and color #1 will be displayed. However, if the "x" bits are not zeroes, another pixel value will be indicated, giving another color table index for that pixel.

If a new set of planes is selected and a new image drawn, images will be written to graphics memory only in the currently selected bitplanes. If overlap between the two images occurs, the overlapping pixels will contain an erroneous color table index value. This is because the color table index is computed from all nine bits, not just the bitplanes enabled by the B command.

Command "B56"

Color = 8

Bit plane number	1	2	3	4	5	6	7	8	9
Bit plane values	1	2	4	8	16	32	64	128	256
Bit plane contents	1	0	0	1	0	0	x	x	x

Color table index = 9

Here the command "B56" ($8+16+32=56$) selects the three middle bits for update, and color #8 is selected. An image is then drawn. Since only the three middle bits can be written, the bit values will be 1, 0, 0 for bits 4, 5, and 6. Though only bits 4, 5, and 6 can be written, all nine bits are used to determine the color index value. When the color table index for these pixels is calculated, the index will be to entry #9. Instead of referencing table entry #8, entry #9 is now being referenced (bitplanes #1 and #4, values $1 + 8 = 9$). So the color in table entry #9 will be displayed.

If the correct color does not reside in the entry indicated by the new index (#9), the displayed color will be wrong and unpredictable. Any new table entry generated in this manner should contain the color indicated by the enabled bits, not the color indicated by all nine bits. For example, if table entry #9 contains the same color values as entry #1, the above problem is corrected. References to both entry #1 and entry #9 will produce the same color.

To achieve correct colors when images can overlap, the color table must be loaded in such a way that the table entries to be referenced will always hold the correct colors. In the above example, with bits 1, 2, and 3 selected for update, every table entry whose value can have as the first three bits 1, 0, 0 must have the same color values loaded.

There are eight (2^3) combinations possible of the "do care" bits, and for each there are 64 (2^6) possible combinations of the "don't care" bits. For each of the eight possible combinations of the three selected bits there should be 64 entries in the color table.

DETERMINING THE COLOR TABLE SET-UP PATTERN

A detailed method to calculate color table loading criteria is described next.

Bit plane number	1	2	3	4	5	6	7	8	9
Bit plane values	1	2	4	8	16	32	64	128	256

1. The number of colors that any set of planes can use is 2^N , where N is the number of bits in that set. In the previous example, each set contains three bits, so each image can use eight colors. Each image can use any eight colors; there is no requirement for different sets to use the same colors.
2. The color table load pattern for each set is determined by the value of the first bit in that set, starting at the left with bit #1. Continuing the example, the bit sets are: 1, 2, and 3 / 4, 5, and 6 / 7, 8, and 9. The first bit plane numbers in each set are 1, 4, and 7. The values for these bits are 1, 8, and 64, respectively. This gives the number of times each color will be consecutively loaded into the table.

When the first set is selected with a "B7" command, the load pattern would be 1, indicating that the colors should be loaded one after another: color 0, color 1, color 7, color 0.

A "B56" command (8+16+32) would select the middle set, with a load pattern of 8. A load pattern of 8 means that each color would be loaded 8 times consecutively: entries 0-7 will contain color 0, 8-15 color 1, 64-71 color 0, etc.

When the third image is to be displayed, the command "B448" (64+128+256) selects the three upper bits for the third set. The pattern for these bits is 64, so each color is repeated 64 times: entries 0-63 contain color 0, 64-127 contain color 1....., entries 128-191 contain color #2, 192-255 contain color #3.

3. The color table load should refill all 512 entries with each change of image. This is done by repeating each pattern as many times as necessary to fill the table. With 8 colors and a pattern of 1, the pattern is repeated 64 times ($8 \times 1 \times 64 = 512$). For a pattern of 8 with 8 colors, the entire pattern is loaded 8 times ($8 \times 8 \times 8 = 512$).

SOME EXAMPLES OF LOGICAL SETS

# IMAGES	SETS	LAYER #	BIT PLANES	# COLORS	COLOR TABLE LOAD PATTERN	PATTERN REPETITION
2	4x5	1	1,2,3,4	16	each color 1 time	32 times
		2	5,6,7,8,9	32	each color 16 times	1 time
3	3x3x3	1	1,2,3	8	each color 1 time	64 times
		2	4,5,6	8	each color 8 times	8 times
		3	7,8,9	8	each color 64 times	1 time
4	2x2x2x3	1	1,2	4	each color 1 time	128 times
		2	3,4	4	each color 4 times	32 times
		3	5,6	4	each color 16 times	8 times
		4	7,8,9	8	each color 64	1 time
5	2x2x2x2x1	1	1,2	4	each color 1 time	128 times
		2	3,4	4	each color 4 times	32 times
		3	5,6	4	each color 16 times	8 times
		4	7,8	4	each color 64 times	2 times
		5	9	2	each color 256 times	1 time

** One of the colors is always #0, and will usually be black.

The general procedure for drawing two distinct images, including figuring the load pattern and repetition count, is detailed next. Assuming two images are required, the planes can be divided into one set of four bitplanes and one set of five. Sixteen colors could be used in the first, thirty-two in the second.

Bit plane number	1	2	3	4	5	6	7	8	9
Bit plane values	1	2	4	8	16	32	64	128	256

1. Select bitplanes for image #1 - "B15" (1+2+4+8=15)
2. Load the color table with 16 colors.

Since the load pattern is 1 (bit #1 is the first bit in the set) and the repetition count is 32 (512 table entries divided by 16 colors divided by 1 time each), the following pattern is correct.

```
Load table entries #0 - #15 with colors 1-16 - repetition #1
                  #16 - #31 with colors 1-16 - repetition #2
                  .
                  .
                  #495 - #511 with colors 1-16 - repetition #32
```

3. Draw first image.
4. Set bitplanes for image #2 - "B496" (16+32+64+128+256=496).
5. Load the color table for image #2.

The load pattern is 16 (bit #5 is the first bit in the set) and the repetition count is 1 (512 table entries divided by 32 colors divided by 16 times each). The following pattern can be used.

```
Load table entries #0 - #15 with color 1
                  #16 - #31 with color 2
                  .
                  .
                  #470 - #511 with color 32
```

The pattern is not repeated because the repetition count is 1.

6. Draw the second image. The images can now be switched by loading the color table with the colors and pattern used by each image.

A SAMPLE PROGRAM

The following program will draw random squares, vertical rectangles, and horizontal rectangles with three bit plane sets of three bits each. Each shape is drawn in a different image layer. When the default color table is intact, the images overlap each other and their colors mix.

After the three images are drawn, the user is asked to select individual image layers, whereupon the color table is redefined for each image.

When the first image is selected and the color table loaded, only the squares will be visible. When the second image is selected, the color table is loaded with the same colors but in a different pattern, bringing the second image, the vertical rectangles, into the "foreground". When the third image is selected, the horizontal rectangles will display immediately after the color table load.

Lines 110-170 contain values for the eight colors to be used, in this order: Black, Red, Green, Blue, Yellow, Magenta, Cyan, and White.

```
10 REM - LAYERED DRAWING DEMO
20 OPEN "VECTRIX" FOR OUTPUT AS #1           'Open file as device
30 CLS : PRINT : PRINT "PRESS ANY KEY TO BEGIN : "
40 PRINT : PRINT "WHEN THE BEEP SOUNDS, ";
50 PRINT "PRESS ANY KEY TO RETURN TO IBM ";
60 PRINT "MODE FOR INPUT"
70 A$ = INKEYS : IF A$ = "" THEN 70
80 PRINT #1,"SV G E0 RE"                     ' Initialize
100 DATA 0,0,0
110 DATA 255,0,0
120 DATA 0,255,0
130 DATA 0,0,255
140 DATA 255,255,0
150 DATA 255,0,255
160 DATA 0,255,255
170 DATA 255,255,255
200 PRINT #1,"B7 E0"                         ' Clear planes 1,2,3
210 FOR I=1 TO 3 : C = 1
220 FOR J=1 TO 7
230 GOSUB 700
240 PRINT #1,"C" C "M" X Y "RF" S S         'Draw squares
250 C=C+1
260 NEXT J : NEXT I
```

```

300 PRINT #1,"B56 E0"
310 FOR I=1 TO 3 : C = 8
320 FOR J=1 TO 7
330 GOSUB 720
340 PRINT #1,"C" C "M" X Y "RF" W H
350 C=C+8
360 NEXT J : NEXT I
400 PRINT #1,"B448 E0"
410 FOR I=1 TO 3 : C = 64
420 FOR J=1 TO 7
430 GOSUB 700
440 PRINT #1,"C" C "M" X Y "RF" H W
450 C=C+64
460 NEXT J : NEXT I
500 BEEP
510 A$ = INKEY$ : IF A$ = "" THEN 510
520 PRINT #1,"SI" : CLS
530 PRINT "THE COLOR TABLE LOAD TAKES A FEW MOMENTS"
540 PRINT : PRINT
550 PRINT "ENTER 1,2, OR 3 TO SELECT IMAGE";
560 INPUT " (RETURN IF DONE) ", Y$
570 IF Y$ = "" THEN CLOSE : END
580 IF Y$ = "1" THEN GOTO 1000
590 IF Y$ = "2" THEN GOTO 1100
600 IF Y$ = "3" THEN GOTO 1200
610 GOTO 500
700 S=INT(RND * 130) +10
710 H=INT(RND * 250) +100
720 W=INT(RND * 20)+10
730 X=INT(RND * 571)
740 Y=INT(RND * 379)
750 RETURN
1000 PRINT #1,"SV Q0, 512"
1010 FOR I=1 TO 64
1020 RESTORE
1030 FOR J=1 TO 8
1040 READ R,G,B : PRINT #1, R G B
1050 NEXT J : NEXT I : GOTO 500
1100 PRINT #1,"SV Q0,512"
1110 FOR I=1 TO 8
1120 RESTORE
1130 FOR J=1 TO 8
1140 READ R,G,B
1150 FOR K=1 TO 8
1160 PRINT #1, R G B
1170 NEXT K : NEXT J : NEXT I : GOTO 500
1200 PRINT #1,"SV Q0,512"
1210 RESTORE
1220 FOR I=1 TO 8
1230 READ R,G,B
1240 FOR J=1 TO 64
1250 PRINT #1, R G B
1260 NEXT J : NEXT I : GOTO 500

```

```

'Clear planes 4,5,6
'
'Draw horizontal
'rectangles
'
'Clear planes 7,8,9
'
'Draw vertical
'rectangles
'
'Set IBM mode
'THE COLOR TABLE LOAD TAKES A FEW MOMENTS"
'
'ENTER 1,2, OR 3 TO SELECT IMAGE";
' (RETURN IF DONE) ", Y$
' IF Y$ = "" THEN CLOSE : END
' IF Y$ = "1" THEN GOTO 1000
' IF Y$ = "2" THEN GOTO 1100
' IF Y$ = "3" THEN GOTO 1200
'
' Return random:
' Height
' Width
' X Position
' Y Position
'
'Reload table to
'display only
'squares in
'
'Reload table to
'display only
'rectangles in
'planes 4,5,6
'
'Reload table to
'display only
'rectangles in
'planes 7,8,9
'

```


A P P E N D I X B

USING DIRECT MEMORY ACCESS

Contents	Page
INTRODUCTION	3
VX/PC DMA	5
SN Command	
SR Command	
SW Command	
IBM and VX/PC DMA	9
Using IBM DMA Transfer Data To The VX/PC	
Source Code for TODMA	
Using IBM DMA To Transfer Data From The VX/PC	
Source Code for FROMDMA	

INTRODUCTION

Direct Memory Access (DMA) allows high speed data transfers. Standard data transfers from memory to a peripheral require software to fetch one byte from memory, check the peripheral's status, and send the byte to a port. For DMA, the software sets up a controller, which then handles the actual transfer automatically.

DMA reduces data transfer time, not processing time. Most data is processed in some way when the VX/PC cards receive it. Exceptions are unencoded RAM transfers (RR & WR) in hex mode. DMA is particularly useful when transferring images or bitplanes with these two commands. An entire RAM image (about 360k) can be written in as little as 1.5 seconds using hex mode, flash mode, and the proper DMA command. This does not include disk access time.

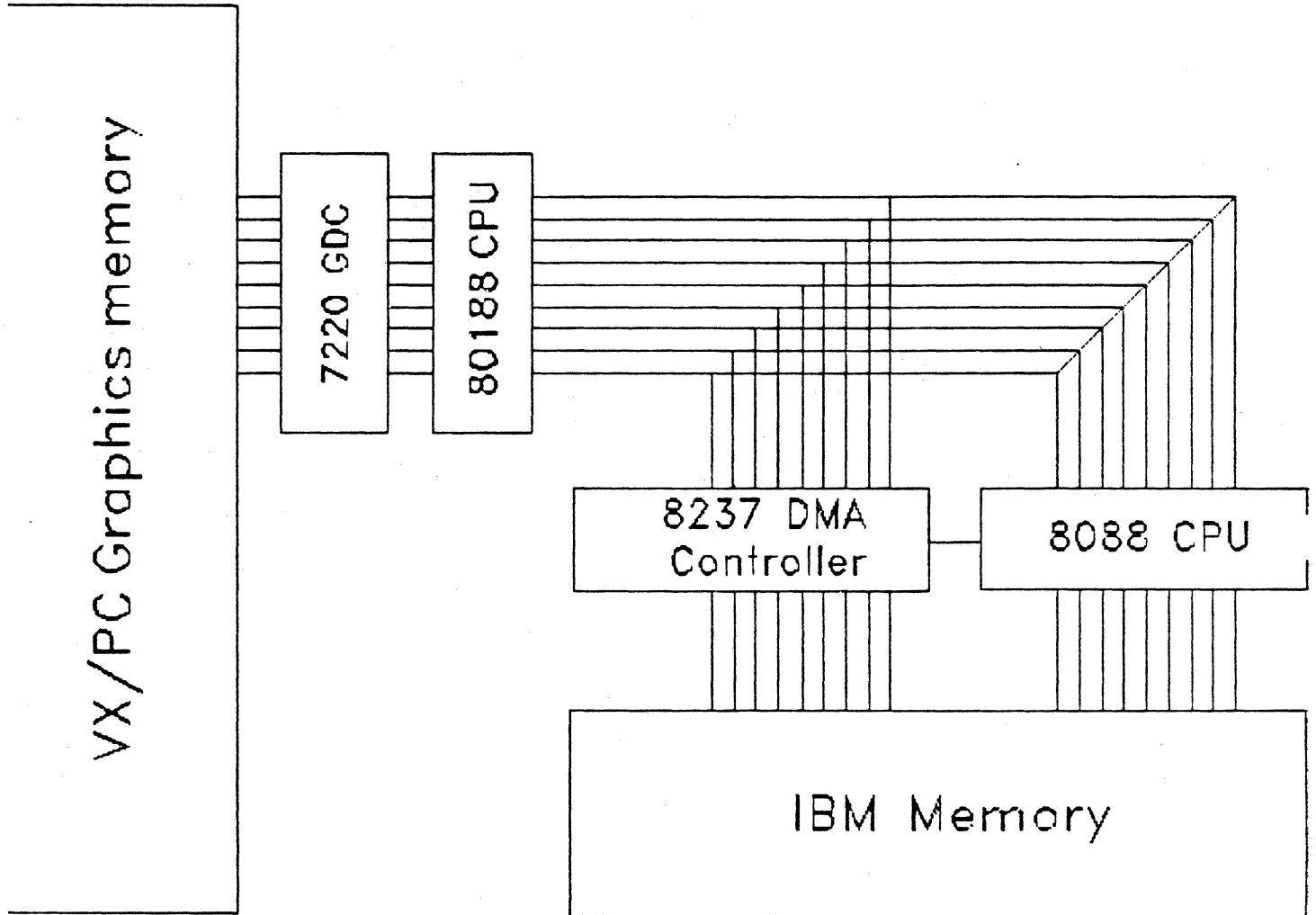
The VX/PC and the IBM PC/XT have separate DMA capabilities which can work together for the fastest transfer time to or from the VX/PC card set. There are three DMA devices involved. They are:

- 80188 DMA on the VX/PC card set
- 7220 DMA on the VX/PC card set
- 8237 DMA on the IBM system board

The SR and SW commands set up the DMA process for the card set only. This does not require the user to set up the 8237 controller on the IBM. If the IBM end of the DMA is desired, the linkable modules TODMA and FROMDMA may be used to set it up. If the IBM's 8237 controller is used, however, the proper DMA command must have been sent to the VX/PC prior to calling the subroutines. The IBM side of a DMA transfer cannot be used without the VX/PC be set up for DMA transfers using the SR or SW commands.

Whenever the VX/PC receives an SR or SW command, it uses DMA channel 1. No other device may use this channel when these commands are in effect. The VX/PC's DMA mode is disabled when it receives the SN command. At that time, another device may use channel 1.

Illustration 1 DMA Block Diagram



TODMA

The program TESTTO serves two functions. The source code included on the VX/PC disk demonstrates using both Vectrix and IBM DMA to move a command file from IBM memory to the VX/PC graphics memory. Secondly, TESTTO can be used directly to move any command file. When the command file was created using hex mode, and flash mode is set, maximum speed is obtained.

FROMDMA

FROMDMA moves the contents of VX/PC graphics memory to the IBM computer using both the Vectrix and IBM DMA functions. The data is transferred in the WR (Write RAM) format.

VECTRIX DMA

The VX/PC command set has three DMA commands. They are:

SR	Sets DMA Read Mode
SW	Sets DMA Write Mode
SN	Sets Non-DMA Mode

The SR command sets up a DMA channel that allows the host computer to read the contents of the VX/PC card's memory.

The SW command sets up a DMA channel, allowing the host computer to write to the VX/PC card.

The SN command disables DMA mode.

SN
SET NON-DMA MODE
OPERATING MODE

PURPOSE: Used to disable DMA transfers.

FORMAT: SN (no arguments)

REMARKS: SN disables the DMA (Direct Memory Access) mode on the VX/PC card. When DMA is disabled, all data transfers are made using the standard I/O protocols.

On power up, DMA is not enabled.

For more information on these protocols and the standard I/O ports, see Chapter 3, Section 3.3.1, "Ports Used".

SR
SET DMA READ MODE
OPERATING MODE

PURPOSE: Allows host to read data from the VX/PC using DMA.

FORMAT: SR (no arguments)

REMARKS: The SR command sets up the DMA controller on the VX/PC, allowing the host computer to read data from the VX/PC using DMA transfers. On power up, DMA is not enabled.

While DMA mode is active on the VX/PC, data may be read from the VX/PC card set using either standard (non-DMA) input protocols, or by using the host computer's DMA controller.

DMA can be used with any data transfer, and is particularly useful when transferring images with the unencoded RR (Read RAM) command in hex mode and flash mode. An image can be read in as little as 1.5 seconds using DMA transfers on both the VX/PC and IBM ends.

SR and its companion command, SW, cannot be in effect at the same time.

Once enabled, DMA read remains in effect until an SW or SN command is sent, or until the system is turned off. DMA read mode is not canceled by a RESET, G, or G0 command.

SW
SET DMA WRITE MODE
OPERATING MODE

PURPOSE: Allows host to write data to the VX/PC DMA channel.

FORMAT: SW (no arguments)

REMARKS: The SW command sets up the DMA controller on the VX/PC, allowing the host computer to write data to the VX/PC using DMA transfers.

While DMA mode is active on the VX/PC, data may be written to the card set using either standard (non-DMA) input protocols, or by using the host computer's DMA controller.

DMA can be used with any data transfer, and is particularly useful when transferring images with the unencoded WR (Write RAM) command in hex mode and flash mode. An entire image can be transferred in as little as 1.5 seconds using DMA on both the VX/PC and IBM ends.

SW and its companion command, SR, cannot be in effect at the same time.

Once enabled, DMA write remains in effect until an SR or SN command is sent, or until the system is turned off. DMA write mode is not canceled by a RESET, G, or G0 command.

IBM AND VX/PC DMA

Two linkable modules, TODMA and FROMDMA are included on the VX/PC disk. These programs set up the IBM DMA function. Before either are called from a program, the VX/PC end of the transfer should set up with the SR or SW command.

Unless the IBM DMA has been specifically set up, any data will be sent to or received by the 8088 CPU. The VX/PC DMA commands do not affect the IBM DMA.

Using IBM DMA Transfer Data To The VX/PC

In order to use the module TODMA.OBJ with a calling program, perform the following steps:

1. The calling program must put the VX\PC into DMA write mode by sending the SW command. Once the command is sent, the VX/PC stays in DMA write mode until it receives the SR or SN command, or until powered down.

***** W A R N I N G *****

If the SN command is sent instead of the SW command, the module returns showing completion, but the VX/PC will not receive data. If the SR command is sent instead of SW, the module will not return. Use Ctrl-Alt-Del to reboot.

2. The calling program must pass two parameters to the module by pushing them onto the stack.
 - a. The address of the data to be transferred is pushed first. This address is the offset from the DS register.
 - b. The address of the byte count is then pushed on the stack. The byte count, an integer variable, is the number of bytes to be transferred minus one. A count of zero transfers one byte. The maximum count is 65535, which transfers 65536 bytes. This address is the offset from the ES register.
3. The calling program must use a far call to TODMA.
4. The user must link TODMA.OBJ with the calling program.

All registers are preserved. The parameters are stripped from the stack prior to returning to the calling program.

The DMA transfers have to be completed successfully before TODMA can return to the calling program.

TODMA source code is shown next, and can be found on the VX/PC disk.

```
*****
;
;
;           TODMA.ASM -- DMA transfer of data to VX/PC
;
;   This module receives the following parameters passed on the
;   stack:
;       1. The address of the data to be transferred.
;       2. The address of a variable which holds the number
;          of bytes to be transferred minus one.
;
;   The module performs one or two transfers, depending on the
;   location of the data. If the data wraps around a page
;   boundary, two transfers are necessary, due to a limitation
;   of the 8237 DMA Controller.
;
;   TODMA requires 20 bytes of stack space. It is up to the
;   calling program to make sure that much stack space is
;   available. The parameters are removed from the stack by
;   this module prior to returning to the calling program.
;
;***** EQUATES *****
DMA      EQU      0                ;address of DMA controller's registers
;
;***** CODE *****
C        SEGMENT PARA PUBLIC
        ASSUME CS:C
        PUBLIC  TODMA
TODMA    PROC      FAR
;
;***** Save calling program's registers *****
;
        PUSH    BP
        MOV     BP,SP              ;used as an index to the stack
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
```

```

;*****
;
; Convert DS value plus beginning address of data into a page
; and offset which can be processed by the 8237. The byte count
; is then added to these values to determine if the data area
; wraps around a page boundary.
;
MOV     BX,DS           ;BX = Data Segment
MOV     AX,WORD PTR[BP+8] ;AX = Data offset
MOV     SI,AX           ;SI = Data offset
MOV     CX,4            ;CX = Shift value
SHR     AX,CL           ;AX = Shifted data offset
ADD     BX,AX           ;BX = New data segment
SHL     AX,CL           ;AX = Shifted data offset reshifted
SUB     SI,AX           ;SI = New data offset
MOV     DX,BX           ;DX = New data segment
AND     DX,0F000H       ;DX = Upper nibble of new DS
SHL     BX,CL           ;BX = New DS, shifted
ADD     BX,SI           ;BX = Shifted new DS plus new offset
MOV     CX,12           ;CX = Shift value
SHR     DX,CL           ;DX = Page number for DMA ctrl
MOV     SI,WORD PTR[BP+6] ;SI = address of byte count
MOV     DI,ES:WORD PTR[SI] ;DI = byte count
MOV     AX,BX           ;AX = Shifted new DS plus new offset
ADD     AX,DI           ;AX = Shifted new DS plus new offset
; plus count
JNC     T0100           ;Does not wrap around page boundary
;
;*****
;
; Data wraps around a page boundary, requires two transfers.
;
PUSH    AX              ;future count for second DMA transfer
;
;*****
;
; Compute count for first transfer
;
MOV     DI,0FFFFH       ;DI = FFFFH
SUB     DI,BX           ;DI = FFFFH Minus shifted new DS
; plus new offset
CALL    DSETUP          ;Call DMA transfer setup
;
POP     DI              ;Pop count for second DMA transfer
MOV     BX,0            ;offset for second transfer
INC     DX              ;next page
T0100: CALL    DSETUP   ;Call DMA transfer setup

```

```

;*****
;
;   Restore calling program's registers
;
    POP     DI
    POP     SI
    POP     DX
    POP     CX
    POP     BX
    POP     AX
    POP     BP
;
;**** Return to calling program, strip stack parameters ****
;
    RET     4
TODMA     ENDP
;
;*****
;
;   Set up 8237 DMA Controller
;
;       BX register = offset into page
;       DL register = page number for 8237
;       DI register = byte count minus one
;
DSETUP:
    CLI             ;clear interrupts
    MOV     AX,49H  ;read from memory command
    OUT     DMA+12,AL ;reset internal flip/flop in 8237
    PUSH   AX      ;waste time
    POP     AX
    OUT     DMA+11,AL ;send command byte to register
    MOV     AX,BX   ;get offset into page
    OUT     DMA+2,AL ;send low byte
    MOV     AL,AH   ;
    OUT     DMA+2,AL ;send high byte
    MOV     AX,DX   ;get page number
    OUT     83H,AL  ;send page number
    MOV     AX,DI   ;get byte count minus one
    OUT     DMA+3,AL ;send low byte
    MOV     AL,AH   ;
    OUT     DMA+3,AL ;send high byte
    STI             ;set interrupt flag
    MOV     AL,1    ;enable channel 1
    OUT     DMA+10,AL
D050:
    IN     AX,DMA+8 ;check status port for DMA complete
    TEST   AL,2
    JZ     D050     ;DMA not complete
    MOV     AL,3    ;DMA complete, disable channel 1
    OUT     DMA+10,AL
    RET             ;DMA complete, return
C
    ENDS
    END

```

Using IBM DMA To Transfer Data From The VX/PC

In order to use the module FROMDMA.OBJ with your calling program, use the following steps:

1. The calling program must put the VX/PC into DMA read mode by sending the SR command. Once the command is sent, the VX/PC stays in DMA read mode until it receives the SW or SN command, or until powered down.

***** W A R N I N G *****

If either the SW or SN command is sent instead of SR before calling this routine, the VX/PC will hang or parity errors will occur in IBM memory. The computer should be turned off, then on. The Ctrl-Alt-Del sequence is not recommended.

2. The calling program must pass two parameters to the module by pushing them onto the stack.
 - a. The address of the data destination is pushed first. This address is offset from the DS register.
 - b. The address of the byte count is then pushed on the stack. The byte count, an integer variable, is the number of bytes to be transferred minus one. A count of zero transfers one byte. The maximum count is 65535, which transfers 65536 bytes. This address is offset from the ES register.
3. The calling program must use a far call to FROMDMA.
4. FROMDMA.OBJ must be linked with the calling program.

All registers are preserved. The parameters are stripped from the stack prior to returning to the calling program. The DMA transfers have to be completed successfully before FROMDMA can return to the calling program.

FROMDMA source code is shown next, and can be found on the VX/PC disk.

```

;*****
;
;
;          FROMDMA.ASM -- DMA transfer of data from VX/PC
;
;  This module received the following parameters passed on the
;  stack:
;
;      1. The address of the data area to be filled.
;
;      2. The address of a variable which holds the number
;         of bytes to be transferred minus one.
;
;  The module performs one or two transfers, depending on
;  the location of the data. If the data wraps around a
;  page boundary, two transfers are necessary, due to the
;  limitation of the 8237 DMA Controller.
;
;  This module requires 20 bytes of stack space. It is up to
;  the calling program to make sure that much stack space
;  is available. The parameters are removed from the stack
;  by this module prior to returning to the calling program.
;
; ***** EQUATES *****
DMA      EQU      0          ;address of DMA controller's registers
;
; ***** CODE *****
C        SEGMENT PARA PUBLIC
        ASSUME CS:C
        PUBLIC FROMDMA
FROMDMA  PROC      FAR
;
; ***** Save calling program's registers *****
;
        PUSH     BP
        MOV      BP,SP      ;used as an index to the stack
        PUSH     AX
        PUSH     BX
        PUSH     CX
        PUSH     DX
        PUSH     SI
        PUSH     DI

```

```

;*****
;
; Convert DS value plus beginning address of data into a page
; and offset which can be processed by the 8237. The byte
; count is then added to these values to determine if the data
; area wraps around a page boundary.
;
MOV     BX,DS           ;BX = Data segment
MOV     AX,WORD PTR[BP+8] ;AX = Data offset
MOV     SI,AX          ;SI = Data offset
MOV     CX,4           ;CX = Shift value
SHR     AX,CL          ;AX = Shifted data offset
ADD     BX,AX          ;BX = New data segment
SHL     AX,CL          ;AX = Shifted data offset reshifted
SUB     SI,AX          ;SI = New data offset
MOV     DX,BX          ;DX = New data segment
AND     DX,0F000H      ;DX = Upper nibble of new DS
SHL     BX,CL          ;BX = New DS, shifted
ADD     BX,SI          ;BX = Shifted new DS plus new offset
MOV     CX,12          ;CX = Shift value
SHR     DX,CL          ;DX = Page number for DMA ctrl
MOV     SI,WORD PTR[BP+6] ;SI = Address of byte count
MOV     DI,ES:WORD PTR[SI] ;DI = Byte count
MOV     AX,BX          ;AX = Shifted new DS plus new offset
ADD     AX,DI          ;AX = Shifted new DS plus new offset
; plus count
JNC     T0100          ;Does not wrap around page boundary
;
;*****
;
; Data wraps around a page boundary and will require two
; transfers.
;
PUSH    AX             ;Future count for second DMA transfer
MOV     DI,0FFFFH      ;DI = FFFFH
SUB     DI,BX          ;DI = FFFFH minus shifted new DS plus
; new offset
CALL    DSETUP         ;Call DMA transfer setup
POP     DI             ;Pop count for second DMA transfer
INC     DX             ;next page
MOV     BX,0           ;offset for second transfer
T0100: CALL    DSETUP  ;Call DMA transfer setup
;***** Restore calling program's registers *****
;
POP     DI
POP     SI
POP     DX
POP     CX
POP     BX
POP     AX
POP     BP

```

```

;*****
;
;       Return to calling program, stripping parameters from stack.
;
;       RET         4
FROMDMA ENDP
;
;*****
;
;       Set up 8237 DMA Controller
;       BX register = offset into page
;       DL register = page number for 8237
;       DI register = byte count minus one
;
DSETUP:
    CLI                     ;clear interrupts
    MOV     AX,45H          ;write to memory command
    OUT    DMA+12,AL        ;reset internal flip/flop in 8237
    PUSH   AX              ;waste time
    POP    AX
    OUT    DMA+11,AL        ;send command byte to register
    MOV    AX,BX            ;get offset into page
    OUT    DMA+2,AL         ;send low byte
    MOV    AL,AH
    OUT    DMA+2,AL         ;send high byte
    MOV    AX,DX            ;get page number
    OUT    83H,AL           ;send page number
    MOV    AX,DI            ;get byte count minus one
    OUT    DMA+3,AL         ;send low byte
    MOV    AL,AH
    OUT    DMA+3,AL         ;send high byte
    STI                     ;set interrupt flag
    MOV    AL,1             ;enable channel 1
    OUT    DMA+10,AL
    MOV    AX,0
D050:
    IN     AX,DMA+8         ;check status port for DMA complete
    TEST   AL,2
    JZ     D050             ;DMA not complete
    MOV    AL,3             ;DMA complete, disable channel 1
    OUT    DMA+10,AL
    RET
C       ENDS
       END

```


GLOSSARY OF COMPUTER GRAPHICS TERMS

Absolute Coordinates	A method of ordering screen coordinates where points are defined in terms of preset coordinates, usually representing the upper right quadrant of the Cartesian coordinate system.
Aliasing	The jagged, "stair-step" representation of diagonals or curves common to most raster displays.
Analog	Discrete representation of a physical quantity; not a digital value.
Analog Monitor	A monitor which displays a virtually infinite number of colors because red, green, and blue inputs are analog voltage levels rather than simply on or off, as with digital or TTL levels.
Anti-aliasing	Modification of raster pixel addressing making stair-step lines appear smooth. Pixels at edges or lines are mixed with the background color in proportions relative to the amount of the boundary actually crossing a particular pixel.
ASCII	Standardized 8-bit data code used for transmitting characters; American Standard Code for Information Interchange.
Aspect Ratio	The ratio of width to height. The standard ratio of a television screen is 4:3.
Baud	A unit of transmission rate equal to the number of signals sent per second; one bit per second in a binary system. 9600 baud is 9600 bits/second or 8 bits, 1 stop bit and 1 start bit per byte, 960 bytes per second.
Bit Map	Digital representation of an image in which memory bits correspond to pixels.

Bit Plane	A collection of memory which holds one bit of color information for each pixel in the frame buffer.
Blanking	The period of the video display cycle between the end of one scan line and the start of the next.
Boundary Color	The color of an outline or region defining an area to be filled.
Boundary Fill	A fill which continues until a specified boundary is reached.
Centronics Interface	An interface developed by Centronics Corp. It allows transmission of 7 or 8 bit data over a parallel data line.
Brightness	An attribute of visual perception in which a source appears to reflect more or less light.
Clipping	The elimination of parts of vectors or areas outside pre-defined boundaries.
Composite Video	A video signal which contains both picture information and sync signals for aligning that picture.
Color Lookup Table	A table which contains distinct red, green, and blue intensity levels for every color value (or number) displayable.
Color Saturation	A measurement of the degree to which a color appears to be free of white light.
Core System	Graphics standards proposed by the American National Standards Institute (ANSI) endorsed by SIGGRAPH.
Crosshair	A positioning cursor.
Current Drawing Color	The color with which a graphic will be drawn, usually the last color used.

Current Drawing Point	The point from which graphic primitives will begin or continue. Usually the last point referenced.
Cursor	Visible marker which can be moved about a display by an input controller.
Defaults	Preprogrammed settings that function until replaced by new settings.
Delimiter	A special character marking the beginning or end of a string of commands, or separating command parameters.
Device Driver	Software adapting the host computer to a graphics device. The device driver is usually device-dependent.
Digital Monitor	A TTL monitor that can display only 8 true colors. Apparent colors can be displayed by dithering, or using different intensity levels.
Digitizer	Any input device such as a tablet or video camera which converts analog data to a digital format.
Display File	Sequence of display commands that create, change, or refresh graphics.
Dithering	Simulating colors by creating patterns consisting of different colored dots.
DMA	Direct Memory Access: transfer of data between the host computer's memory and the graphics display controller without passing through the host CPU.
Duplex Transmission	Simultaneous, independent two-way transmission via a data bus.
Field	When in interlaced mode, a field is 1/2 of the total picture refresh cycle. A field consists of odd or even lines. Two fields are scanned to produce a frame.
Field Frequency	The number of fields displayed per second.

Fill	Spreading color and pattern to cover an enclosed area.
Flicker	A fluctuation of display intensity between refresh cycles when the interval exceeds the phosphor persistence.
Frame	When viewing interlaced video, a frame is composed of two fields. In non-interlaced video, one frame is equal to one field.
Frame Buffer	Local raster memory that stores bit maps, also the device containing the graphics memory.
GKS Graphics Standard	Graphical Kernel System
Grey Scale	Variations in the luminance value of white light. In an RGB system, the equal combination of red, green, and blue light.
Half Duplex	Half-duplex transmission is one-way but permits direction reversal.
Hex (Hexadecimal)	The base 16 representation of numbers. The digits 0-9 are paired with letters A through F to represent decimal numbers 0-15.
Hue	Color obtained by mixing intensities of red, green, and blue phosphors.
Image	A displayed view of an object or parts of an object.
Image Processing	Computer analysis of graphic data.
Image Transformation	A manipulation of an image with respect to size, orientation, or position on a display.
Interlaced	Alternating between odd and even scan lines during successive refresh passes in order to accelerate display updates.
Interrupt	A break in a normal machine cycle or program caused by an external device or circuit.

Key-frame Animation	Computer animation controlling display changes through a succession of frames.
Light Pen	A graphic input device which allows direct interaction with a monitor.
Matrix	An array of numbers used to calculate graphic transformations.
Memory management	Allocation and deallocation of memory addresses to display file or data segments. Used to optimize memory efficiency.
NTSC	Abbreviation for the National Television System Committee, and used to identify the color encoding method adopted by the committee.
Non-interlaced	Refresh of all raster lines on CRT during each scan.
Paging	Organizing a display into groups of images that can be individually recalled from the host computer.
Pan	Movement or windowing about an image.
Parallel Interface	An interface which allows a computer to transfer information a predefined number of bits at a time.
Parity Check	Checks that a word has an even or odd number of bits. Used to check that bits are not lost in transmission.
Pixel	An individual dot on the display device. Also called picture element or pel.
Polygon Fill	Coloring or crosshatching an area defined by a boundary.
Primitive	Basic display element: a point, line, alphanumeric character or marker.
Primitive Attribute	A characteristic of a display primitive, such as color, intensity, or width.

Raster	A predetermined pattern of scanning lines providing uniform coverage of the display.
Refresh Cycle	Complete scan of display area by electron beam to maintain image illumination on a CRT screen.
Relative Coordinates	A method of ordering screen coordinates where points are defined in terms of the current location rather than a preset point.
Resolution	CRT precision as measured by the number of scan lines comprising display area, and the number of pixels that can be displayed horizontally.
RGB	Red-Green-Blue: referring to the use of three separate color signals for finer color differentiation than is possible with the "NTSC" TV system.
Rotation	Moving a figure around an axis.
RS-232	A serial interface standard permitting linkage of graphics terminal or other peripherals.
Run-length Encoding	A method of compressing picture data by storing counts of identical consecutive pixels and the color of those pixels.
Serial Transmission	Sending sequential data, one bit at a time, via a single data line.
Scaling	Enlarging or reducing size of a figure.
Stylus	Special pen used with a data or graphics tablet.
Tablet	See Graphics Tablet.
Transformation Matrix	An array of X,Y,and Z coefficients for calculating a geometric transformation.

Translation	A method of orienting or duplicating a figure on a display device.
Upload	Transferring information from one device to another.
Vector generator	A microprocessor that translates move and draw instructions into displayed lines.
Vertical Retrace	Return of electron beam to upper left corner of a CRT display surface after completion of a refresh cycle.
Viewing Pyramid	A representation of the perspective techniques used in a computer; a pyramid where the base is the projection plane and the apex is the eyepoint.
Viewplane	Logical surface for 2D projections. Can be extended to create a 3D viewing volume.
Viewport	The area of the display screen chosen for displaying images, defining areas where clipping takes place.
Warmstart	Resetting system to default parameters without turning machine off and on.
Wire Frame	3D image constructed as a series of line segments outlining surfaces with no shading.
Z-clipping	Defining the foreground and background limits of a 3-D display with front and rear planes parallel to the viewplane.

FURTHER READING

Angell, Ian:

A Practical Introduction to Computer Graphics,
Halsted Press, div. Wiley and Sons, New York 1981

Foley, James, Ph.D. and A. Van Dam:

Fundamentals of Interactive Computer Graphics
Wiley and Sons, New York 1981

Giloi, Wolfgang:

Interactive Computer Graphics
Prentice Hall, Englewood Cliffs, New Jersey 1978

Greenburg, Donald et al:

The Computer Image
Addison-Wesley, Reading MA 1982

Myers, Roy:

Microcomputer Graphics
Addison-Wesley, Reading MA 1982

Newman, William and Sproul, Robert:

Principles of Interactive Computer Graphics
McGraw Hill 1973.

Pavlidis, Theo:

Algorithms for Graphics and Image Processing
Computer Science Press, Rockville MD 1982

Schachter, Bruce J. et al:

Computer Image Generation
Wiley and Sons, New York 1983

INDEX

\$ Command, 4-12

3D, 2-16..2-27, 4-22

Commands, 4-5, 4-29, 4-58,
4-81, 4-85, 4-91, 4-97
use in 2D mode, 2-26

Coordinates, 2-16

Perspective Scaling, 2-26, 4-85

Rotation, 2-19...2-24, 4-58

Transformations, 2-16..2-26

Translation, 2-21...2-27, 4-91

Viewing Pyramid, 2-17...2-18, 4-84

Viewport, 2-21, 2-24, 4-97

A

A Command, 2-7, 2-27, 4-14, 4-49

Absolute Coordinates, 4-42, 4-65

Adapter

Color Graphics, 1-10, 3-3

Monochrome, 1-9, 1-13, 3-3

Adjust Character Angle, 4-30, 4-49

Advanced Programming Manual, 4-96

Animation, 2-8, 4-15

ANSI.SYS, 1-19

APRINT, 1-26, 2-30..2-32, 3-12

Arc

Boundary Filled Wedge, 4-99

Command, 4-15, 4-49, 4-59

Originate, 4-50

Wedge, 4-100

Argument Values, 4-7...4-9

ASCII Characters, 2-7, 4-31

See Also Character Set, Text Commands

ASCII Mode, 2-5, 4-8, 4-10, 4-16,

4-19, 4-44, 4-67, 4-70, 4-74, 4-77,

4-74, 4-77, 4-78, 4-102, 4-105,

4-117

Aspect Ratios, 2-26

Assembly Language, 3-11..3-14, 4-9,

4-12, 4-96

Automatic Mode Changes, 2-5

Axes

Rotation Around, 2-19..2-26, 4-59

B

B Command, 2-5, 2-12, 2-13, 4-15,

4-53, 4-60, 4-61, 4-64, Appendix A

Bi-Directional

Commands, 3-6, 4-24..4-27, 4-62,

4-67, 4-69, 4-72, 4-74, 4-77, 4-78,

4-117

Device Driver, 3-4

Example, 3-7

Bitplanes, 2-8, 4-4, 4-15, 4-19, 4-62,

4-70, 4-74, 4-78, 4-102, 4-107,

Appendix A

Mask Register, 4-22, 4-74

Write Mask Command, 4-16

Blank Mode, 3-8, 4-43

Boards

See VX/PC Cards

Border Colors, 4-83

Boundary Color, 4-100, 4-111, 4-112

Boundary Fill

Complex, 4-111

Brackets

Card Support, 1-3, 1-10, 1-15

Byte Values, 4-8...4-10

C

C Command, 2-5, 4-16, 4-65

Card Support Brackets, 1-3, 1-10, 1-15

Cards

See VX/PC Cards

Change Character Magnification, 4-34

Change Color, 4-16

Change Color Lookup Table, 4-56

Character Cell, 4-32, 4-34, 4-36

Character Commands, 4-4

Character Set, 2-7, 4-12, 4-35

Angle, 4-30

Custom, 4-32

Default Spacing, 4-32

In 3D mode, 2-8

Circles, 4-14, 4-50, 4-99

Filled, 4-100

Clear Screen, 4-19

Clipping, 2-17..2-18, 4-8, 4-41,

4-65, 4-86, 4-97

2D, 2-27

CODEFRAG.ASM, 3-14

Color

Borders, 4-83

Boundary, 4-100, 4-105, 4-111, 4-112

Changes, 4-16

Commands, 4-4

Complementing, 4-61

Current Drawing,

See Current Drawing Color

Drawing Modes,

See Color Drawing Modes

Fill, 4-100, 4-20

Lookup Table,

See Color Lookup Table

Manipulations,

Defined, 2-4

See also Color Commands

Mixing, 4-53

Number of, 2-8, 4-15, Appendix A

Replacement, 4-60, 4-61, 4-64

Color and Alignment Test, 1-21

Color Commands, 4-4, 4-16, 4-19,

4-56, 4-77, 4-89

See also Color Drawing Modes

Color Drawing Modes, 2-12...2-15,

4-5, 4-22, 4-36, 4-53, 4-60, 4-61,

4-64, 4-95

Descriptions, 2-12

Example Program, 2-15

Table, 2-13

Color Graphics Adapter, 1-10, 3-3

Color Lookup Table Commands, 2-8..2-11,

4-16, 4-22, Appendix A

Default, 2-10,

Color Lookup Table Commands (continued)

- Define Entries, 4-56
- Indexes, 2-8..2-11, 4-15, 4-56, 4-77
- Masked bits, 2-10
- Return To Default, 4-89
- Return Value, 4-77
- RGB components, 2-8..2-11
- Saving, 3-7
- Color Manipulations,
 - Defined, 2-4
 - See also Color Commands
- Color Printers, 1-5, 1-26, 2-31
- Color Printing, 2-30...2-34
 - Codes, 2-32
 - Commands, 2-32, 4-6, 4-24..4-27
 - Data Format, 3-13
 - Dithered, 2-30...2-34, 4-26, 4-27
 - From BASICA, 2-32
 - Non-Dithered, 2-30, 4-23, 4-24
 - Reverse, 4-24, 4-26
 - Software,
 - See MPRINT, QPRINT, APRINT
 - See Also Hardcopy
- Color Table
 - See Color Lookup table
- Command Files, 1-22, 1-26, 2-6
 - SPHERE, 1-25
 - TEST01.DAT, 1-26
- COMMAND.COM, 1-18
- Commands
 - See Also Individual Commands
 - 3D, 4-5, 4-29, 4-58, 4-81,
 - 4-85, 4-91, 4-97
 - Arguments, 4-8..4-10
 - Bi-Directional,
 - See Bi-Directional Commands
 - Color,
 - See Color Commands
 - Color Printing,
 - See Color Printing, Commands
 - Crosshair,
 - See Crosshair Commands
 - Dithering, 2-33..2-34, 4-26, 4-27
 - DMA,
 - Appendix B
 - Graphic Primitives,
 - See Graphics Primitives
 - Hardcopy, 4-5
 - See Color Printing, Commands
 - Miscellaneous, 4-6, 4-80, 4-96, 4-101
 - Operating Modes,
 - see Operating Modes
 - Pixel, 2-28, 3-8..3-10, 4-5, 4-69,
 - 4-74, 4-102, 4-107
 - RAM, 3-8..3-10, 4-5, 4-72, 4-78,
 - 4-105, 4-109
 - Text Primitives, 2-7, 4-4, 4-12,
 - 4-30, 4-32, 4-34, 4-37
 - Video, 4-6, 4-51, 4-52, 4-55, 4-119
- Complex Boundary Fill, 4-111
- Complex Flood Fill, 4-112
- Compression, 2-28, 3-8..3-10
- CONFIG.SYS, 1-19, 3-4

Connectors

- BNC, 1-17
- DB-9, 1-11, 1-16, 1-17
- Control-E, 4-20, 4-111, 4-112
- Coordinates
 - 2D, 4-40
 - 3D, 2-16, 4-41
 - Absolute, 4-42
 - Relative, 4-46
 - Screen, 4-62
- COPY Command, 1-21, 1-25, 1-26, 2-6, 3-8
- Crosshair Commands, 2-29, 4-5, 4-113..4-118
- Crosshatching, 4-36
- Current Drawing Color, 4-12, 4-14, 4-16,
 - 4-18, 4-20, 4-61, 4-65, 4-66, 4-99,
 - 4-100, 4-111, 4-112
 - defined, 2-3
- Current Drawing Point, 4-12, 4-14, 4-18
 - 4-19, 4-22, 4-29, 4-40, 4-46, 4-47,
 - 4-48, 4-50, 4-54, 4-62, 4-65, 4-66,
 - 4-69, 4-71, 4-73..4-75, 4-78, 4-79,
 - 4-91, 4-94, 4-102..4-113
 - defined, 2-3

D

- D Command, 2-7, 4-18
- Dashed Lines, 4-49
- Data Transmission, 2-5..2-6, 3-8..3-10
- Defaults, 4-22
 - Bitplanes, 4-15
 - Character Set, 2-7
 - Character Size, 4-34
 - Character Spacing, 4-38
 - Color Lookup Table, 2-8..2-11, 4-88
 - Crosshair, 4-108, 4-115
 - Pattern Register, 4-49
 - System, 4-22
 - Video Timing, 4-101
 - Viewport, 2-97
- Define Color Lookup Table Value, 4-56
- Define Viewport Command, 4-97
- Delimiters, 4-11
- Design New Character, 4-32
- Device Drivers, 1-19, 1-21, 3-4..3-5
- Direct Memory Access, Appendix B
- Direction Of Rotation, 2-19, 4-58
- Directory, Root, 1-18
- Disk,
 - VX/PC, See VX/PC Disk
- Display Character String, 4-12
- Display Default Character Set, 4-35
- Dithered Printing,
 - See Color Printing
- DMA, 4-79, 4-109, Appendix B
- Dot Command, 4-18
- Drawing Color,
 - See Current Drawing Color
- Drawing Point,
 - See Current Drawing Color
- Driver, See Device Drivers
- Dual Monitor System, 1-10, 1-13, 1-17,
 - 2-4, 3-3, 4-83, 4-90
- Duplicating Figures, 2-22, 4-91

E

E Command, 2-5, 4-19
EDLIN.COM, 1-19
Effects
 Telephoto, 4-87
 Wide Angle, 4-88
Encoded Pixels, 4-69, 4-102
Encoded RAM, 4-72, 4-105
Encoding, Run Length, 2-28..2-29
 See Also Individual Commands
Erase Screen Command, 4-19
Errors
 In Vectrix Mode, 4-90
 Single Monitor System, 3-3
Eye position, 2-16..2-18, 4-87..4-88

F

F Command, 2-7, 2-27, 4-20, 4-49
Fills
 Color 4-20
 Complex Boundary, 4-111
 Complex Flood, 4-112
 Patterns, 4-36
Filled Polygon, 4-20
Flash Mode, 3-8..3-9, 4-45, 4-76, 4-77,
 4-79, 4-104, 4-106
Flood Fill, 4-112
FROMDMA, Appendix B, 3-14
Function Keys, 4-83, 4-90
 Setting, 3-4

G

G, G0 Commands, 4-15, 4-22, 4-23,
 4-29, 4-33, 4-35, 4-36, 4-55, 4-116
GETCHR, 3-13
Go Warmstart, 4-21, 4-22
Graphics Memory
 Bitplanes, 2-8, 4-15
 Organization, 2-8
 Unchanged By Zoom, 4-119
Graphics Primitives, 4-4, 4-14, 4-18,
 4-20, 4-36, 4-47..4-50, 4-54, 4-62
 4-65, 4-67, 4-99, 4-100, 4-105, 4-106
Grey Lookup Table, 2-8..2-11
 See Also Color Lookup Table

H

Hardcopy, 2-30..2-34
 Commands, 4-24..4-27
 See Also Color Printing
Hex Mode, 2-5, 3-8, 4-7..4-10, 4-16,
 4-19, 4-28, 4-63, 4-70, 4-73, 4-74,
 4-76, 4-77, 4-99, 4-101, 4-102, 4-104
CHRS() Format, 2-6
 Trapping Control Z, 3-8
Hidden Drawing, 2-8, Appendix A
HNP Command, 2-32, 4-24
HNR Command, 2-32, 4-25
Horizontal Text Spacing, 4-37
HP Command, 2-32, 4-26
HR Command, 2-32, 4-28
HX Command, 2-5, 3-8, 4-28, 4-44
 See Also Hex Mode

I

I Command, 2-21, 2-29, 4-29, 4-86, 4-92
IBM mode, 2-4, 3-3, 4-83
Image
 Compression, 2-28, 3-8, 4-10..4-11,
 4-69..4-71, 4-72..4-73
 Loading and Saving, 3-8, 4-69, 4-72,
 4-74, 4-78, 4-102, 4-105, 4-107, 4-109
 Manipulation, 4-55, 4-94, 4-119
 Mirrored, 4-81
Increasing Speed, 3-8
 See Also Run Length Encoding
 See Also Hex Mode
Indexes, Color Lookup Table
 See Color Lookup Table
Initialize Transformation Matrix, 2-21,
 2-29, 4-29, 4-86, 4-92
Installation
 of Card Set, 1-6...1-16
 of Driver, 1-18..1-19
 of Monitor, 1-17
 of Printer, 1-26
 of Light Pen, 1-28
 Problems, 1-23
 Testing, 1-21

J

JA Command, 4-30, 4-38
JD Command, 2-7, 4-32, 4-34, 4-38,
JM Command, 4-34, 4-38
JN Command, 2-8, 4-33, 4-35
JR Command, 2-12, 4-36, 4-49, 4-50,
 4-61, 4-66
JS Command, 4-38, 4-31
Jump Table, 4-96

K

K2 Command, 4-40, 4-63
K3 Command, 2-21, 4-41, 4-48, 4-54,
 4-65, 4-86
KA Command, 4-42, 4-47, 4-48
KB Command, 4-43
KD Command, 2-5, 4-44
KF Command, 3-8, 3-9, 4-45
KR Command, 4-43, 4-46, 4-47

L

L Command, 2-7, 2-27, 4-47, 4-49
Least Significant Byte (LSB), 2-6
Light Pen,
 Command, 4-67
 Connector Pins, 1-30
 Installing, 1-28
 Testing, 1-31
Line Spacing, 4-38, 4-31
Line Command, 2-7, 2-27, 4-47, 4-49
Lookup Table
 See Color Lookup Table

M

M Command, 2-7, 2-27, 4-48

Manipulation
Color, 4-4
See Also Image, Manipulations
Matrix Transformation, 4-22
See Also 3D, 3D Commands
Memory Card, 1-11
See also VX/PC Cards
MIDASDRV, 1-19, 3-4, 3-14
Mirror Images, 4-81
Miscellaneous Commands, 4-6, 4-80,
4-96, 4-101
Mixing Colors, 4-53
MODEKEYS, 1-21
Modes
3D, See 3D
ASCII, See ASCII Mode
Blank, 3-8, 4-43, 4-77
COBOL, 2-4
Color Drawing,
See Color Drawing Modes
Flash, 3-8, 4-19, 4-45, 4-73, 4-76,
4-77, 4-78, 4-104, 4-106
Hex, See Hex Mode
IBM, 2-4, 3-4, 4-83
MONO, 2-4
Operating,
See Operating Modes
Vectrix, 2-4, 3-4, 4-90
Monitor
Installation, 1-17
Monochrome, See Monochrome Monitor
RGB, See RGB Monitor
Monochrome Monitor, 1-5, 1-10, 1-13,
1-17, 2-4, 3-3
Most Significant Byte (MSB), 2-6
Move Command, 2-7, 2-27, 4-48
Moving Figures, 4-90
See Also Translation, 3D Commands
MPRINT, 1-26, 2-30..2-32, 3-12

N
N Command, 2-12, 4-49, 4-60
Non-Dithered Printing, 2-30, 4-24, 4-25
Normal Perspective, 4-87

O
OA Command, 2-7, 2-27, 4-50
Object Space Coordinates, 2-16, 4-41
See Also 3D Commands
OF Command, 4-51
ON Command, 4-52
Operating Modes, 4-4, 4-28, 4-40..4-45,
4-83, 4-90
Options, System, 1-5
OR Command, 2-12, 4-37, 4-53
OR Mode, See OR Command
Origin
2D, 2-16, 4-4, 4-40
3D, 2-16, 4-41
External, 2-19..2-26, 4-59
Internal, 2-23, 4-59
See Also 3D Commands
Originate Arc Command, 2-7, 2-27, 4-50

P
P Command, 2-7, 2-27, 4-49, 4-55
PAN Command, 4-55
Pattern Register,
One Dimensional, 4-49, 4-20, 4-60,
4-99, 4-100, 4-112
Two Dimensional, 4-36, 4-60
Perspective, 2-18
Command (SP), 4-85..4-88
Decreasing, 4-88
Fore-shortening, 4-87
Normal, 4-87
Pie Charts, 4-99, 4-100
Pixel Commands, 2-28..2-29, 3-8..3-10,
4-5, 4-69, 4-74, 4-102, 4-107
Polygon
Command, 2-7, 2-27, 4-49, 4-55
Filled, 4-20
Ports Addresses, 3-11
Primitives
Graphics, See Graphics Primitives
Text, See Text Primitives
Printer
See Color Printer
Printers Supported, 2-31
Printing, See Color Printing
Problem Determination, 1-23
Processor Card, 1-11
See also VX/PC Cards
Programs
2D Input, 2-35
3D Input, 2-36
Color Drawing Modes, 2-15
Color printing, 2-33
FROMDMA, Appendix B
Layered Drawing, Appendix A
Save Color Lookup Table, 3-7, Appendix A
Save Pixels, 3-9
Save RAM, 3-10
TODMA, Appendix B
Utilities, 3-4, 3-5
PROM Version, 4-80
PUTCHR, 3-12
Pyramid,
See Viewing Pyramid

Q
Q Command, 2-5, 2-17..2-18, 4-16, 4-56
QPRINT, 1-26, 2-30..2-32, 3-12

R
R Command, 2-19..2-24, 4-58
RA Command, 2-12..2-15, 4-37, 4-60
Radius
of Filled Wedge Arc, 4-100
of Arc, 4-14, 4-50
of Wedge Arc, 4-99
RAM Commands, 2-28, 3-8, 4-5, 4-72,
4-78, 4-105, 4-109
Example Program, 3-10
RAM Size Installed, 4-80
RC Command, 2-12, 4-14, 4-20, 4-37,
4-50, 4-61, 4-99, 4-100

RD Command, 4-9..4-10, 4-62, 4-116
 RE Command, 2-12, 4-36, 4-60, 4-64, 4-95, 4-108
 Read Graphics RAM, 2-28, 4-78
 Encoded, 2-28, 4-72, 4-105
 Read Pixels, 2-28, 4-74
 Encoded, 2-28, 4-69, 4-102
 Read Port, 3-11
 Rectangle Fill Command, 4-65
 Rectangle Fill Pattern, 4-36, 4-60
 Registers,
 Pattern, 2-12, 4-20, 4-49, 4-60, 4-99, 4-100, 4-112
 Bitplane Mask, 2-12..2-13, 4-15, 4-22, 4-74
 REPLACE ALL Mode, 2-12..2-15, 4-37, 4-60
 REPLACE COMPLEMENT Mode, 2-12, 4-14, 4-20, 4-37, 4-50, 4-61, 4-99, 4-100
 REPLACE Mode, 2-12, 4-36, 4-60, 4-64, 4-95, 4-108
 Requirements
 Hardware, 1-4
 Software, 1-18
 Reset
 Soft, 4-22, 4-33
 Port, 3-11
 RESET.EXE, 2-4, 3-5
 Resolution,
 Printer, 4-7
 Screen, 3-14
 Return Color Lookup Table To Default, 4-89
 Return Color Lookup Table Value, 4-77
 Return Crosshair Position, 4-117
 Return Drawing Point, 4-62
 Return Light Pen Position, 4-67
 Return Version Number, 4-80
 Reversal
 Color, 4-61
 Reverse Printing, 4-25, 4-27
 Revision Level, 4-80
 RF Command, 2-27, 4-65, 4-82
 RGB Monitor, 1-17, 1-29, 2-4, 3-3
 RL Command, 4-67
 RNP Command, 2-28, 3-8..3-10, 4-69, 4-102
 RNR Command, 2-28, 3-8..3-10, 4-72, 4-105
 Root Directory, 1-18
 Rotation, 2-19..2-24, 4-28, 4-58
 2D Use, 2-27
 Direction of, 2-19
 Using Translation, 4-58
 RP Command, 2-28, 4-74
 RQ Command, 4-8, 4-77
 RR Command, 4-78
 Run Length Encoding, 2-28, 4-106, 4-72, 4-78, 4-102..4-106
 RV Command, 4-80

S
 S Command, 2-25..2-27, 4-81
 Scale, 2-25..2-26
 Command, 4-80
 Select Absolute Coordinates, 4-42
 Select ASCII Decimal Mode, 4-44
 Select Blank Video Mode, 4-43

Select Flash Video Mode, 4-45
 Select Hex Mode, 4-28
 Select IBM Mode, 4-83
 Select Relative Coordinates, 4-46
 Select Vectrix Mode, 4-90
 Set 2D Coordinates, 4-40
 Set 3D Coordinates, 4-41
 Set Character Spacing, 4-38
 Set Color, 4-16
 Set Crosshair Position, 4-116
 Set Crosshair Position To CDP, 4-113
 Set Crosshair Size, 4-118
 Set Pattern Register, 4-49
 Set Perspective Scaling, 4-85
 Set Rectangle Fill Pattern, 4-36
 SHELL Command, 2-33
 SI Command, 1-23, 2-4, 4-83
 SI.EXE, 3-5
 Single Monitor System, 1-10, 1-13, 1-17, 2-4, 3-3, 4-83, 4-90
 Slant Angle, 4-30, 4-38
 Soft Reset, 4-22, 4-33
 Software Installation,
 See Installation, Driver
 Source Files, 3-14
 CODEFRAG.ASM, 3-12, 3-13
 FROMDMA.ASM, Appendix B
 MIDASDRV.ASM, 3-5
 MPRINT.ASM, 1-26, 2-30..2-32, 3-12
 TODMA.ASM, Appendix B
 SP Command, 4-85
 Spacing
 Character, 4-31, 4-38
 SPHERE.DAT, 1-25
 SQ Command, 4-89
 Start Angle
 Arc, 4-15, 4-49, 4-59
 Filled Wedge Arc, 4-99
 Wedge Arc, 4-100
 Status Port, 3-11
 Subroutines
 Assembly Language, 3-12, 3-13
 SV Command, 1-21, 2-4, 3-5, 3-10, 4-90
 SV.EXE, 3-5
 Switches,
 Monitor, 1-17
 System, 1-12
 System
 Options, 1-5
 Requirements, 1-4
 System Disk, 1-18..1-20
 Command files, 1-25, 1-28, 1-29
 CONFIG.SYS, 1-19

T
 T Command, 2-21..2-24, 4-91
 TB Command, 4-94
 Telephoto Effect, 4-87
 Test Pattern, 1-24
 Testing
 Color Printer, 1-26
 Installation, 1-21
 Light Pen, 1-29

Text Primitives, 2-7, 4-4, 4-12, 4-30,
4-32, 4-34, 4-35, 4-38

TODMA.ASM, Appendix B, 3-14

TODMA.OBJ, Appendix B

Transfer Block Command, 4-94

Transformation, 2-4

3D, 2-16...2-27

See Also Rotation, Scaling,
and Translation

Matrix, 4-22 4-29

Rotation, 2-19...21, 2-23, 4-58

Scaling, 2-25...2-27, 4-81, 4-85

Translate, 2-21...2-22, 4-91

Translation, 2-21...2-22, 4-29

2D use, 2-27

Command, 4-90...4-93

To Origin, 2-23, 4-91

Turn Video Off, 4-51

Turn Video On, 4-52

U

U Command, 4-96

Unfilled Polygons, 4-54

Unsigned Numbers, 4-9

Upload Code Command, 4-96

Utility Programs, 3-4, 3-5

V

V Command, 4-97

Vectrix mode, 2-4

Vertical Spacing, 4-38

Video Commands, 4-5, 4-51, 4-55, 4-119

Viewing Pyramid, 2-17...2-18, 4-85...4-88

Viewport, 2-18, 4-22, 4-40...4-41, 4-58,
4-65, 4-108

2D, 2-27, 4-97

3D, 4-97

Command, 4-97

Style, 2-25

VX/PC Cards, 1-11

Built in IBM color card, 3-3

Installing, 1-6...1-18

Standard Configuration, 2-8

VX/PC Disk, 1-18, 1-21, 1-25, 1-26, 1-28

Command files

TEST01.DAT, 1-21, 1-23

SPHERE, 1-25

Programs,

APRINT, 1-26, 2-30...2-32, 3-12

MIDASDRV.COM, 1-18...1-19, 3-4

MODEKEYS, 1-21

MPRINT, 1-26, 2-30...2-32, 3-12

QPRINT, 1-26, 2-30...2-32, 3-12

RESET.EXE, 2-4, 3-5

SI.EXE, 3-5

SV.EXE, 3-5

TEST02.BAS, 1-28

Source Files,

See Source Files

W

WA Command, 2-7, 4-99

Wait Frames Command, 4-101

Warmstart Command, 4-22, 4-23

WB Command, 2-7, 4-100

Wedge Arc

Boundary Filled, 2-7, 4-100

Wedge Arc Command, 2-7, 4-99

WF Command, 4-101

Wide Angle Effect, 4-88

WNP Command, 2-8, 2-28, 3-9, 4-69, 4-102

WNR Command, 4-105

Word Values, 4-8...4-10

WP Command, 2-28, 4-107

WR Command, 2-28, 4-109

Write Graphics RAM, 4-109

Encoded, 4-105

Write Pixels, 4-107

Encoded, 4-102

Write Port, 3-11

X

XB Command, 4-21, 4-111

XF Command, 4-21, 4-112

XHC Command, 4-113

XHF Command, 4-114

XHN Command, 4-115

XHP Command, 4-116

XHR Command, 4-117

XHS Command, 4-118

Z

Z Command, 4-119

Z coordinate, 2-16...2-24

Zoom Video Image Command, 4-119

Appendix 3. Port Addresses

This appendix contains the addresses for the peripherals in both the VX384 and the VX/PC. Remember that I/O space and memory space are not the same. No VX384 memory space addresses are listed, since no peripherals exist in the memory space of the VX384.

VX384 I/O Space Addresses

0000H	GDC parameter register
0002H	GDC command register
1000H	8255 I/O controller port A (HOST parallel port)
1001H	8255 I/O controller port B (PRINTER port)
1002H	8255 I/O controller port A (HOST parallel control signals)
1003H	8255 I/O controller command register
2000H	8259 interrupt controller register 0
2001H	8259 interrupt controller register 1
3000H	8251 Serial UART data register
3001H	8251 Serial UART command register
4000H-41FFH	Red color lookup table
4200H-43FFH	Secondary red lookup table
5000H-51FFH	Green color lookup table
5200H-53FFH	Secondary green lookup table
6000H-61FFH	Blue color lookup table
6200H-63FFH	Secondary blue lookup table
7000H	GDC DMA output channel
8000H	8279 keyboard controller data register
8FFFH	8279 keyboard controller command register

VX/PC Memory Space Addresses

0:2000H-0:23FFH	Red color lookup table
0:2400H-0:27FFH	Secondary red lookup table
0:2800H-0:2BFFH	Green color lookup table
0:2C00H-0:2FFFH	Secondary green lookup table
0:3000H-0:33FFH	Blue color lookup table
0:3400H-0:37FFH	Secondary blue lookup table
0:3800H (write only)	DMA direction register:
	bit 0 = 1 for DMA read
	= 0 for DMA write
	bit 1 (Rev. B only)
	= 1 to enable the VX/PC as IBM DMA device 1
	= 0 to disable as an IBM DMA device

VX/PC I/O Space Addresses

0000H	GDC parameter register
0002H	GDC command register
0080H	GDC DMA parameter register (generates acknowledge)
0082H	GDC DMA command register (generates acknowledge)
0100H	(write only) Control register: <ul style="list-style-type: none">bit 0 = NOT zoom0 \ Horizontal zoom factor forbit 1 = NOT zoom1 \ display. You must also setbit 2 = NOT zoom2 / the GDC zoom.bit 3 = NOT zoom3/bit 4 = 1 if in IBM color card mode = 0 if in VX modebit 5 = 1 selects upper lookup table for display = 0 selects lower lookup table for displaybit 6 = 1 if not at minimum zoom (Z1) 0 if at minimum zoom (Z1)bit 7 = 1 to halt timing generation to RAM (do not use) = 0 to refresh video RAM
0180H	(read only) IBM color card color register
0200H	(write only) Register to send data to the IBM through IBM I/O address 03DEH.
0280H	(read only) Register to get data from the IBM through IBM I/O address 03DDH.
0300H	(read only) Status register: <ul style="list-style-type: none">(bits 0 through 5 are as defined in the IBM Technical reference manual under color card.)bit 0 = 80 columnbit 1 = Graphicsbit 2 = NOT light pen switchbit 3 = Video enablebit 4 = 640 modebit 5 = Blink enablebit 6 = 1 when channel to IBM is clear = 0 when channel to IBM is busybit 7 = 1 when no data is available from the IBM = 0 when data is ready from the IBM channel

Permission to reprint granted by NEC Electronics, Inc., One Natick Executive Park, Natick, MA 01760. The information in this document is subject to change without notice. NEC Electronics, Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties or merchantability and fitness for a particular purpose. NEC Electronics, Inc. makes no commitment to update nor to keep current the information contained in this document. July 1983.

Description

The μPD7220 Graphics Display Controller (GDC) is an intelligent microprocessor peripheral designed to be the heart of a high-performance raster-scan computer graphics and character display system. Positioned between the video display memory and the microprocessor bus, the GDC performs the tasks needed to generate the raster display and manage the display memory. Processor software overhead is minimized by the GDC's sophisticated instruction set, graphics figure drawing, and DMA transfer capabilities. The display memory supported by the GDC can be configured in any number of formats and sizes up to 256K 16-bit words. The display can be zoomed and panned, while partitioned screen areas can be independently scrolled. With its light pen input and multiple controller capability, the GDC is ideal for advanced computer graphics applications.

For a more detailed description of the GDC's operation, please refer to the GDC Design Manual.

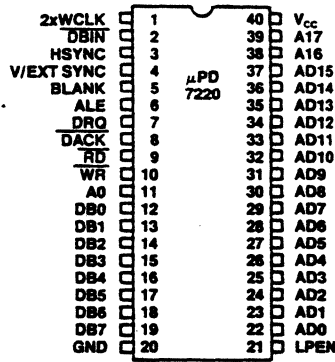
System Considerations

The GDC is designed to work with a general purpose microprocessor to implement a high-performance computer graphics system. Through the division of labor established by the GDC's design, each of the system components is used to the maximum extent through six-level hierarchy of simultaneous tasks. At the lowest level, the GDC generates the basic video raster timing, including sync and blanking signals. Partitioned areas on the screen and zooming are also accomplished at this level. At the next level, video display memory is modified during the figure drawing operations and data moves. Third, display memory addresses are calculated pixel by pixel as drawing progresses. Outside the GDC at the next level, preliminary calculations are done to prepare drawing parameters. At the fifth level, the picture must be represented as a list of graphics figures drawable by the GDC. Finally, this representation must be manipulated, stored, and communicated. By handling the first three levels, the GDC takes care of the high-speed and repetitive tasks required to implement a graphics system.

Features

- Microprocessor Interface
 - DMA transfers with 8257- or 8237-type controllers
 - FIFO Command Buffering
- Display Memory Interface
 - Up to 256K words of 16 bits
 - Read-Modify-Write (RMW) Display Memory cycles in under 800ns
 - Dynamic RAM refresh cycles for nonaccessed memory
- Light Pen Input
- External video synchronization mode
- Graphics Mode
 - Four megabit, bit-mapped display memory
- Character Mode
 - 8K character code and attributes display memory
- Mixed Graphics and Character Mode
 - 64K if all characters
 - 1 megapixel if all graphics
- Graphics Capabilities
 - Figure drawing of lines, arc/circles, rectangles, and graphics characters in 800ns per pixel
 - Display 1024-by-1024 pixels with 4 planes of color or grayscale
 - Two independently scrollable areas
- Character Capabilities
 - Auto cursor advance
 - Four independently scrollable areas
 - Programmable cursor height
 - Characters per row: up to 256
 - Character rows per screen: up to 100
- Video Display Format
 - Zoom magnification factors of 1 to 16
 - Panning
 - Command-settable video raster parameters
- Technology
 - Single +5 volt, NMOS, 40-pin DIP
- DMA Capability
 - Bytes or word transfers
 - 4 clock periods per byte transferred

Pin Configuration



Pin Identification

Pin			
No.	Symbol	Direction	Function
1	2xWCLK	IN	Clock Input
2	DBIN	OUT	Display Memory Read Input Flag
3	HSYNC	OUT	Horizontal Video Sync Output
4	V/EXT SYNC	IN/OUT	Vertical Video Sync Output or External VSYNC Input
5	BLANK	OUT	CRT Blanking Output
6	ALE (RAS)	OUT	Address Latch Enable Output
7	DRQ	OUT	DMA Request Output
8	DACK	IN	DMA Acknowledge Input
9	RD	IN	Read Strobe Input for Microprocessor Interface
10	WR	IN	Write Strobe Input for Microprocessor Interface
11	A0	IN	Address Select Input for Microprocessor Interface
12-19	DB0 to 7	IN/OUT	Bidirectional Data Bus to Host Microprocessor
20	GND	—	Ground
21	LPEN	IN	Light Pen Detect Input
22-34	AD0 to 12	IN/OUT	Address and Data Lines to Display Memory
35-37	AD13 to 15	IN/OUT	Utilization Varies with Mode of Operation
38	A16	OUT	Utilization Varies with Mode of Operation
39	A17	OUT	Utilization Varies with Mode of Operation
40	V _{cc}	—	+ 5V ± 10%

Character Mode Pin Utilization

Pin			
No.	Name	Direction	Function
35-37	AD13 to 15	OUT	Line Counter Bits 0 to 2 Outputs
38	A16	OUT	Line Counter Bit 3 Output
39	A17	OUT	Cursor Output and Line Counter Bit 4*

Mixed Mode Pin Utilization

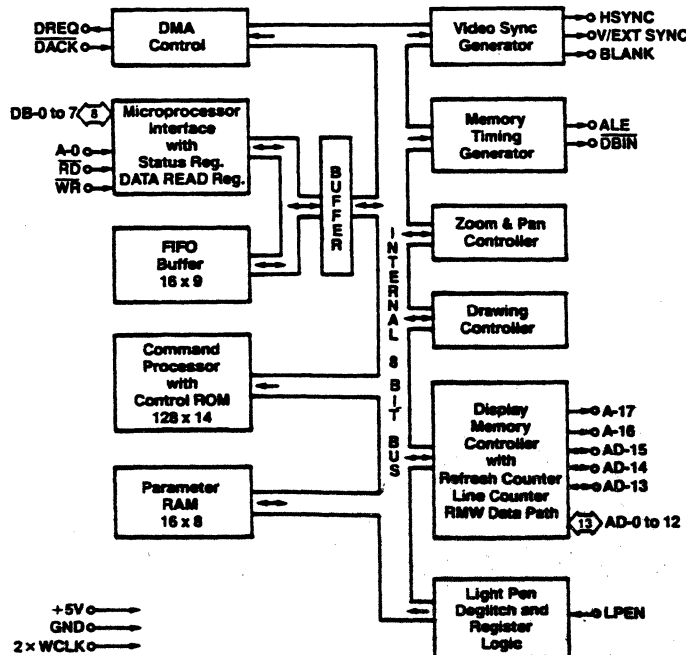
Pin			
No.	Name	Direction	Function
35-37	AD13 to 15	IN/OUT	Address and Data Bits 13 to 15
38	A16	OUT	Attribute Blink and Clear Line Counter* Output
39	A17	OUT	Cursor and Bit-Map Area* Flag Output

*Output 10 clock cycles after trailing edge of HSYNC. See figure for timing example.

Graphics Mode Pin Utilization

Pin			
No.	Name	Direction	Function
35-37	AD13 to 15	IN/OUT	Address and Data Bits 13 to 15
38	A16	OUT	Address Bit 16 Output
39	A17	OUT	Address Bit 17 Output

Block Diagram



GDC Components

Microprocessor Bus Interface

Control of the GDC by the system microprocessor is achieved through an 8-bit bidirectional interface. The status register is readable at any time. Access to the FIFO buffer is coordinated through flags in the status register and operates independently of the various internal GDC operations, due to the separate data bus connecting the interface and the FIFO buffer.

Command Processor

The contents of the FIFO are interpreted by the command processor. The command bytes are decoded, and the succeeding parameters are distributed to their proper destinations within the GDC. The command processor yields to the bus interface when both access the FIFO simultaneously.

DMA Control

The DMA control circuitry in the GDC coordinates transfers over the microprocessor interface when using an external DMA controller. The DMA Request and Acknowledge handshake lines directly interface with a μ PD8257 or μ PD8237 DMA controller, so that display data can be moved between the microprocessor memory and the display memory.

Parameter RAM

The 16-byte RAM stores parameters that are used repetitively during the display and drawing processes. In character mode, this RAM holds four sets of partitioned display area parameters; in graphics mode, the drawing pattern and graphics character take the place of two of the sets of parameters.

Video Sync Generator

Based on the clock input, the sync logic generates the raster timing signals for almost any interlaced, non-interlaced, or "repeat field" interlaced video format. The generator is programmed during the idle period following a reset. In video sync slave mode, it coordinates timing between multiple GDCs.

Memory Timing Generator

The memory timing circuitry provides two memory cycle types: a two-clock period refresh cycle and the read-modify-write (RMW) cycle which takes four clock periods. The memory control signals needed to drive the display memory devices are easily generated from the GDC's ALE and DBIN outputs.

Zoom & Pan Controller

Based on the programmable zoom display factor and the display area entries in the parameter RAM, the zoom and pan controller determines when to advance to the next memory address for display refresh and when to go on to the next display area. A horizontal zoom is produced by slowing down the display refresh rate while maintaining the video sync rates. Vertical zoom is accomplished by repeatedly accessing each line a number of times equal to the horizontal repeat. Once the line count for a display area is exhausted, the controller accesses the starting address and line count of the next display area from the parameter RAM. The system microprocessor, by modifying a display

area starting address, can pan in any direction, independently of the other display areas.

Drawing Controller

The drawing processor contains the logic necessary to calculate the addresses and positions of the pixels of the various graphics figures. Given a starting point and the appropriate drawing parameters, the drawing controller needs no further assistance to complete the figure drawing.

Display Memory Controller


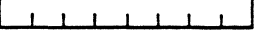
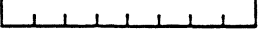
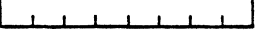
The display memory controller's tasks are numerous. Its primary purpose is to multiplex the address and data information in and out of the display memory. It also contains the 16-bit logic unit used to modify the display memory contents during RMW cycles, the character mode line counter, and the refresh counter for dynamic RAMs. The memory controller apportions the video field time between the various types of cycles.

Light Pen Deglitcher

Only if two rising edges on the light pen input occur at the same point during successive video fields are the pulses accepted as a valid light pen detection. A status bit indicates to the system microprocessor that the light pen register contains a valid address.

Programmer's View of GDC

The GDC occupies two addresses on the system microprocessor bus through which the GDC's status register and FIFO are accessed. Commands and parameters are written into the GDC's FIFO and are differentiated based on address bit A0. The status register or the FIFO can be read as selected by the address line.

A0	READ	WRITE
0	Status Register 	Parameter into FIFO 
1	FIFO Read 	Command into FIFO 

GDC Microprocessor Bus Interface Registers

Commands to the GDC take the form of a command byte followed by a series of parameter bytes as needed for specifying the details of the command. The command processor decodes the commands, unpacks the parameters, loads them into the appropriate registers within the GDC, and initiates the required operations.

The commands available in the GDC can be organized into five categories as described in the following section.

GDC Command Summary

Video Control Commands

1. RESET Resets the GDC to its idle state.
2. SYNC Specifies the video display format.
3. VSYNC Selects master or slave video synchronization mode.
4. CCHAR Specifies the cursor and character row heights.

Display Control Commands

1. START Ends Idle mode and unblanks the display.
2. BCTRL Controls the blanking and unblanking of the display.
3. ZOOM Specifies zoom factors for the display and graphics characters writing.
4. CURS Sets the position of the cursor in display memory.
5. PRAM Defines starting addresses and lengths of the display areas and specifies the eight bytes for the graphics character.
6. PITCH Specifies the width of the X dimension of display memory.

Drawing Control Commands

1. WDAT Writes data words or bytes into display memory.
2. MASK Sets the mask register contents.
3. FIGS Specifies the parameters for the drawing controller.
4. FIGD Draws the figure as specified above.
5. GCHRD Draws the graphics character into display memory.

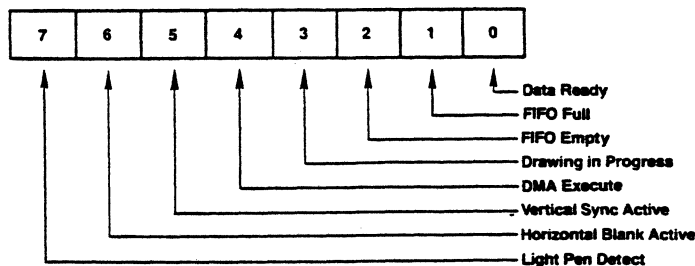
Data Read Commands

1. RDAT: Reads data words or bytes from display memory.
2. CURD: Reads the cursor position.
3. LPRD: Reads the light pen address.

DMA Control Commands

1. DMAR Requests a DMA read transfer.
2. DMAW Requests a DMA write transfer.

Status Register Flags



Status Register (SR)

SR-7: Light Pen Detect

When this bit is set to 1, the light pen address (LAD) register contains a deglitched value that the system microprocessor may read. This flag is reset after the 3-byte LAD is moved into the FIFO in response to the light pen read command.

SR-6: Horizontal Blanking Active

A 1 value for this flag signifies that horizontal retrace blanking is currently underway.

SR-5: Vertical Sync

Vertical retrace sync occurs while this flag is a 1. The vertical sync flag coordinates display format modifying commands to the blanked interval surrounding vertical sync. This eliminates display disturbances.

SR-4: DMA Execute

This bit is a 1 during DMA data transfers.

SR-3: Drawing in Progress

While the GDC is drawing a graphics figure, this status bit is a 1.

SR-2: FIFO Empty

This bit and the FIFO Full flag coordinate system microprocessor accesses with the GDC FIFO. When it is 1, the Empty flag ensures that all the commands and parameters previously sent to the GDC have been interpreted.

SR-1: FIFO Full

A 1 at this flag indicates a full FIFO in the GDC. A 0 ensures that there is room for at least one byte. This flag needs to be checked before each write into the GDC.

SR-0: Data Ready

When this flag is a 1, it indicates that a byte is available to be read by the system microprocessor. This bit must be tested before each read operation. It drops to a 0 while the data is transferred from the FIFO into the microprocessor interface data register.

FIFO Operation & Command Protocol

The first-in, first-out buffer (FIFO) in the GDC handles the command dialogue with the system microprocessor. This flow of information uses a half-duplex technique, in which the single 16-location FIFO is used for both directions of data movement, one direction at a time. The FIFO's direction is controlled by the system microprocessor through the GDC's command set. The host microprocessor coordinates these transfers by checking the appropriate status register bits.

The command protocol used by the GDC requires differentiation of the first byte of a command sequence from the succeeding bytes. The first byte contains the operation code and the remaining bytes carry parameters. Writing into the GDC causes the FIFO to store a flag value alongside the data byte to signify whether the byte was written into the command or the parameter address. The command processor in the GDC tests this bit as it interprets the entries in the FIFO.

The receipt of a command byte by the command processor marks the end of any previous operation. The number of parameter bytes supplied with a command is cut short by the receipt of the next command byte. A read operation from the GDC to the microprocessor can be terminated at any time by the next command.

The FIFO changes direction under the control of the system microprocessor. Commands written into the GDC always put the FIFO into write mode if it wasn't in it already.

If it was in read mode, any read data in the FIFO at the time of the turnaround is lost. Commands which require a GDC response, such as RDAT, CURD and LPRD, put the FIFO into read mode after the command is interpreted by the GDC's command processor. Any commands and parameters behind the read-evoking command are discarded when the FIFO direction is reversed.

Read-Modify-Write Cycle

Data transfers between the GDC and the display memory are accomplished using a read-modify-write (RMW) memory cycle. The four clock period timing of the RMW cycle is used to: 1) output the address, 2) read data from the memory, 3) modify the data, and 4) write the modified data back into the initially selected memory address. This type of memory cycle is used for all interactions with display memory including DMA transfers, except for the two clock period display and RAM refresh cycles.

The operations performed during the modify portion of the RMW cycle merit additional explanation. The circuitry in the GDC uses three main elements: the Pattern register, the Mask register, and the 16-bit Logic Unit. The Pattern register holds the data pattern to be moved into memory. It is loaded by the WDAT parameters or, during drawing, from the parameter RAM. The Mask register contents determine which bits of the read data will be modified. Based on the contents of these registers, the Logic Unit performs the selected operations of REPLACE, COMPLEMENT, SET, or CLEAR on the data read from display memory.

The Pattern register contents are ANDed with the Mask register contents to enable the actual modification of the memory read data, on a bit-by-bit basis. For graphics drawing, one bit at a time from the Pattern register is combined with the Mask. When ANDed with the bit set to a 1 in the Mask register, the proper single pixel is modified by the Logic Unit. For the next pixel in the figure, the next bit in the Pattern register is selected and the Mask register bit is moved to identify the pixel's location within the word. The Execution word address pointer register, EAD, is also adjusted as required to address the word containing the next pixel.

In character mode, all of the bits in the Pattern register are used in parallel to form the respective bits of the modify data word. Since the bits of the character code word are used in parallel, unlike the one-bit-at-a-time graphics drawing process, this facility allows any or all of the bits in a memory word to be modified in one RMW memory cycle. The Mask register must be loaded with 1s in the positions where modification is to be permitted.

The Mask register can be loaded in either of two ways. In graphics mode, the CURS command contains a four-bit dAD field to specify the dot address. The command processor converts this parameter into the one-of-16 format used in the Mask register for figure drawing. A full 16 bits can be loaded into the Mask register using the MASK command. In addition to the character mode use mentioned above, the 16-bit MASK load is convenient in graphics mode when all of the pixels of a word are to be set to the same value.

The Logic Unit combines the data read from display memory, the Pattern Register, and the Mask register to generate the data to be written back into display memory. Any one of four operations can be selected: REPLACE,

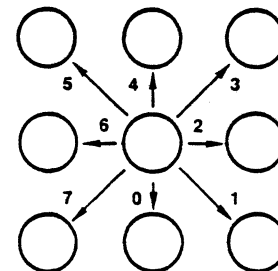
COMPLEMENT, CLEAR or SET. In each case, if the respective Mask bit is 0, that particular bit of the read data is returned to memory unmodified. If the Mask bit is 1, the modification is enabled. With the REPLACE operation, the Pattern Register data simply takes the place of the read data for modification enabled bits. For the other three operations, a 0 in the modify data allows the read data bit to be returned to memory. A 1 value causes the specified operation to be performed in the bit positions with set Mask bits.

Figure Drawing

The GDC draws graphics figures at the rate of one pixel per read-modify-write (RMW) display memory cycle. These cycles take four clock periods to complete. At a clock frequency of 5MHz, this is equal to 800ns. During the RMW cycle the GDC simultaneously calculates the address and position of the next pixel to be drawn.

The graphics figure drawing process depends on the display memory addressing structure. Groups of 16 horizontally adjacent pixels form the 16-bit words which are handled by the GDC. Display memory is organized as a linearly addressed space of these words. Addressing of individual pixels is handled by the GDC's internal RMW logic.

During the drawing process, the GDC finds the next pixel of the figure which is one of the eight nearest neighbors of the last pixel drawn. The GDC assigns each of these eight directions a number from 0 to 7, starting with straight down and proceeding counterclockwise.



Drawing Directions

Figure drawing requires the proper manipulation of the address and the pixel bit position according to the drawing direction to determine the next pixel of the figure. To move to the word above or below the current one, it is necessary to subtract or add the number of words per line in display memory. This parameter is called the pitch. To move to the word to either side, the Execute word address cursor, EAD, must be incremented or decremented as the dot address pointer bit reaches the LSB or the MSB of the Mask register. To move to a pixel within the same word, it is necessary to rotate the dot address pointer register to the right or left.

The table below summarizes these operations for each direction.

Dir	Operations to Address the Next Pixel
000	EAD + P → EAD
001	EAD + P → EAD dAD (MSB) = 1: EAD + 1 → EAD dAD → LR
010	dAD (MSB) = 1: EAD + 1 → EAD dAD → LR
011	EAD - P → EAD dAD (MSB) = 1: EAD + 1 → EAD dAD → LR
100	EAD - P → EAD
101	EAD - P → EAD dAD (LSB) = 1: EAD - 1 → EAD dAD → RR
110	dAD (LSB) = 1: EAD - 1 → EAD dAD → RR
111	EAD + P → EAD dAD (LSB) = 1: EAD - 1 → EAD dAD → RR

Where P = Pitch, LR = Left Rotate, RR = Right Rotate,
EAD = Execute Word Address, and
dAD = Dot Address stored in the Mask Register.

Whole word drawing is useful for filling areas in memory with a single value. By setting the Mask register to all 1s with the MASK command, both the LSB and MSB of the dAD will always be 1, so that the EAD value will be incremented or decremented for each cycle regardless of direction. One RMW cycle will be able to effect all 16 bits of the word for any drawing type. One bit in the Pattern register is used per RMW cycle to write all the bits of the word to the same value. The next Pattern bit is used for the word, etc.

For the various figures, the effect of the initial direction upon the resulting drawing is shown below:

Dir	Line	Arc	Character	Slant Char	Rectangle	DMA
000						
001						
010						
011						
100						
101						
110						
111						

Note that during line drawing, the angle of the line may be anywhere within the shaded octant defined by the DIR value. Arc drawing starts in the direction initially specified by the DIR value and veers into an arc as drawing proceeds. An arc may be up to 45 degrees in length. DMA transfers are done on word boundaries only, and follow the arrows indicated in the table to find successive word addresses. The slanted paths for DMA transfers indicate the GDC changing both the X and Y components of the word address when moving to the next word. It does not follow a 45 degree diagonal path by pixels.

Drawing Parameters

In preparation for graphics figure drawing, the GDC's Drawing Processor needs the figure type, direction and drawing parameters, the starting pixel address, and the pattern from the microprocessor. Once these are in place within the GDC, the Figure Draw command, FIGD, initiates the drawing operation. From that point on, the system microprocessor is not involved in the drawing process. The GDC Drawing Controller coordinates the RMW circuitry and address registers to draw the specified figure pixel by pixel.

The algorithms used by the processor for figure drawing are designed to optimize its drawing speed. To this end, the specific details about the figure to be drawn are reduced by the microprocessor to a form conducive to high-speed address calculations within the GDC. In this way the repetitive, pixel-by-pixel calculations can be done quickly, thereby minimizing the overall figure drawing time. The table below summarizes the parameters.

Drawing Type	DC	D	D2	D1	DM
Initial Value*	0	8	8	-1	-1
Line	Δx	2 ΔD - Δx	2(ΔD - Δx)	2 ΔD	-
Arc**	r sin φ	r - 1	2(r - 1)	-1	r sin θ ↓
Rectangle	3	A - 1	B - 1	-1	A - 1
Area Fill	B - 1	A	A	-	-
Graphic Character***	B - 1	A	A	-	-
Write Data	W - 1	-	-	-	-
DMAW	D - 1	C - 1	-	-	-
DMAR	D - 1	C - 2	(C - 2)/2†	-	-
Read Data	W	-	-	-	-

*Initial values for the various parameters remain as each drawing process ends.

**Circles are drawn with 8 arcs, each of which span 45°, so that sin φ = 1/√2 and sin θ = 0.

***Graphic characters are a special case of bit-map area filling in which B and A ≤ 8. If A = 8 there is no need to load D and D2.

Where:

-1 = all ONES value.

All numbers are shown in base 10 for convenience. The GDC accepts base 2 numbers (2s complement notation where appropriate).

- = No parameter bytes sent to GDC for this parameter.

Δx = The larger of Δx or Δy.

ΔD = The smaller of Δx or Δy.

r = Radius of curvature, in pixels.

φ = Angle from major axis to end of the arc. φ ≤ 45°

θ = Angle from major axis to start of the arc. θ ≤ 45°

↑ = Round up to the next higher integer.

↓ = Round down to the next lower integer.

A = Number of pixels in the initially specified direction.

B = Number of pixels in the direction at right angles to the initially specified direction.

W = Number of words to be accessed.

C = Number of bytes to be transferred in the initially specified direction. (Two bytes per word if word transfer mode is selected.)

D = Number of words to be accessed in the direction at right angles to the initially specified direction.

DC = Drawing count parameter which is one less than the number of RMW cycles to be executed.

DM = Dots masked from drawing during arc drawing.

† = Needed only for word reads.

Graphics Character Drawing

Graphics characters can be drawn into display memory pixel-by-pixel. The up to 8-by-8 character display is loaded into the GDC's parameter RAM by the system microprocessor. Consequently, there are no limitations on the character set used. By varying the drawing parameters and drawing direction, numerous drawing options are available. In area fill applications, a character can be written into display memory as many times as desired without reloading the parameter RAM.

Once the parameter RAM has been loaded with up to eight graphics character bytes by the appropriate PRAM command, the GCHRD command can be used to draw the bytes into display memory starting at the cursor. The zoom magnification factor for writing, set by the zoom command, controls the size of the character written into the display memory in integer multiples of 1 through 16. The bit values in the PRAM are repeated horizontally and vertically the number of times specified by the zoom factor.

The movement of these PRAM bytes to the display memory is controlled by the parameters of the FIGS command. Based on the specified height and width of the area to be drawn, the parameter RAM is scanned to fill the required area.

For an 8-by-8 graphics character, the first pixel drawn uses the LSB of RA-15, the second pixel uses bit 1 of RA-15, and so on, until the MSB of RA-15 is reached.

The GDC jumps to the corresponding bit in RA-14 to continue the drawing. The progression then advances toward the LSB of RA-14. This snaking sequence is continued for the other 6 PRAM bytes. This progression matches the sequence of display memory addresses calculated by the drawing processor as shown above. If the area is narrower than 8 pixels wide, the snaking will advance to the next PRAM byte before the MSB is reached. If the area is less than 8 lines high, fewer bytes in the parameter RAM will be scanned. If the area is larger than 8 by 8, the GDC will repeat the contents of the parameter RAM in two dimensions, as required to fill the area with the 8-by-8 mozaic. (Fractions of the 8-by-8 pattern will be used to fill areas which are not multiples of 8 by 8.)

Parameter RAM Contents: RAM Address RA 0 to 15

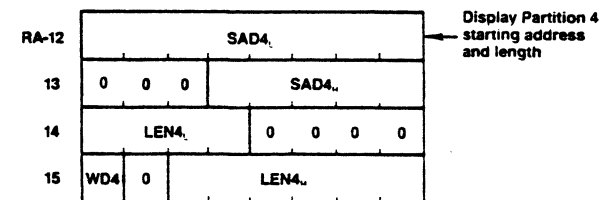
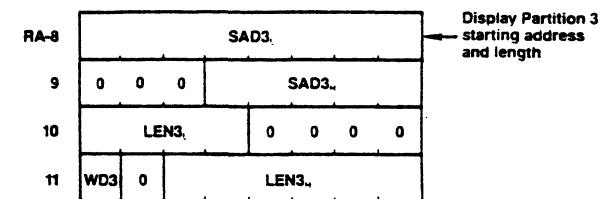
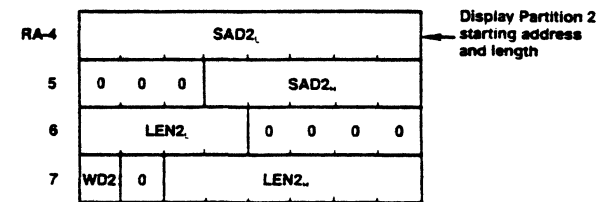
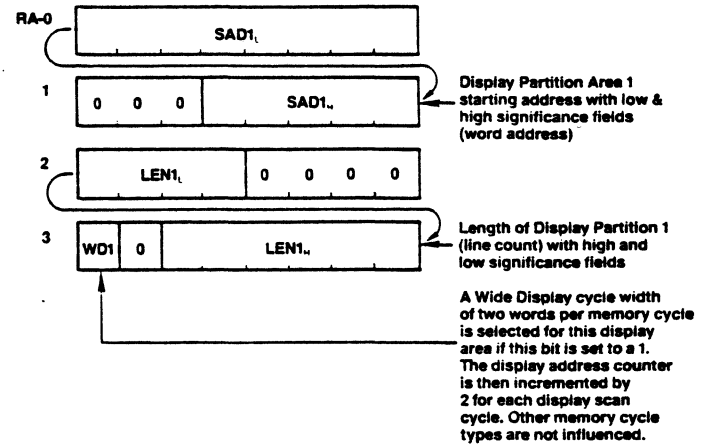
The parameters stored in the parameter RAM, PRAM, are available for the GDC to refer to repeatedly during figure drawing and raster-scanning. In each mode of operation the values in the PRAM are interpreted by the GDC in a predetermined fashion. The host microprocessor must load the appropriate parameters into the proper PRAM locations. PRAM loading command allows the host to write into any location of the PRAM and transfer as many bytes as desired. In this way any stored parameter byte or bytes may be changed without influencing the other bytes.

The PRAM stores two types of information. For specifying the details of the display area partitions, blocks of four bytes are used. The four parameters stored in each block include the starting address in display memory of each display area, and its length. In addition, there are two mode bits for each area which specify whether the area is a bit-mapped graphics area or a coded character area, and whether a 16-bit or a 32-bit wide display cycle is to be used for that area.

The other use for the PRAM contents is to supply the pattern for figure drawing when in a bit-mapped graphics area or mode. In these situations, PRAM bytes 8 through 16 are reserved for this patterning information. For line, arc, and rectangle drawing (linear figures) locations 8 and 9 are loaded into the Pattern Register to allow the GDC to draw dotted, dashed, etc. lines. For area filling and graphics bit-mapped character drawing locations 8 through 15 are referenced for the pattern or character to be drawn.

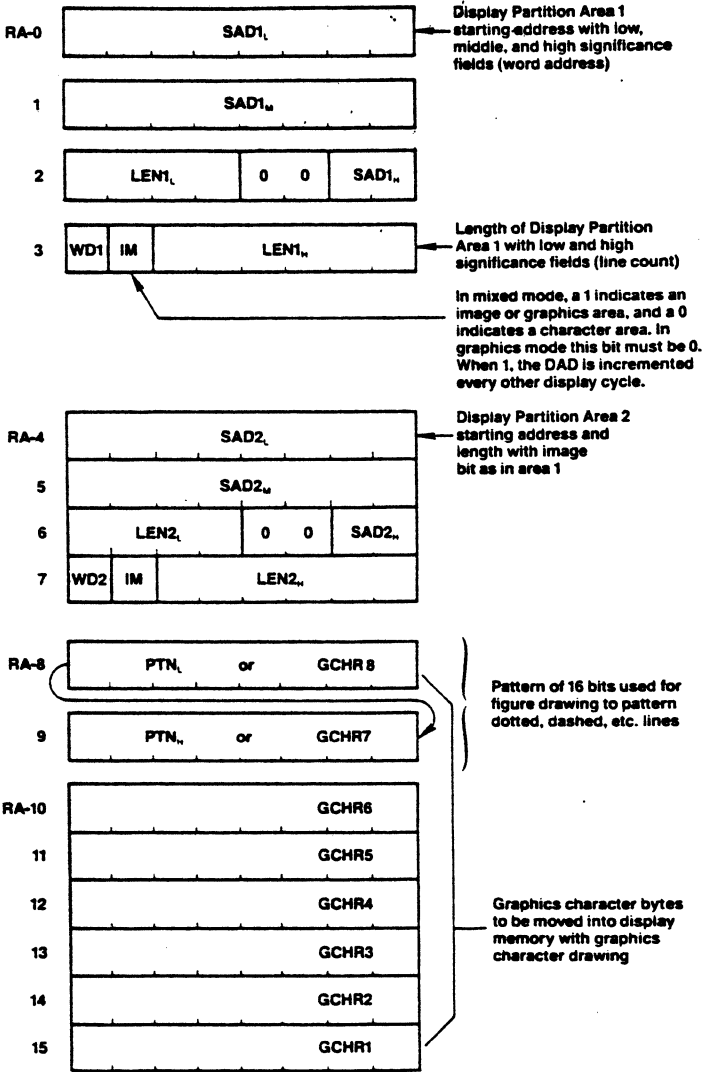
Details of the bit assignments are shown for the various modes of operation.

Character Mode



Graphics and Mixed Graphics and Character Modes

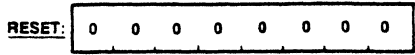
Command Bytes Summary



RESET	0 0 0 0	0 0 0 0
SYNC	0 0 0 0	1 1 1 DE
VSYNC	0 1 1 0	1 1 1 M
CCHAR	0 1 0 0	1 0 1 1
START	0 1 1 0	1 0 1 1
BCTRL	0 0 0 0	1 1 0 DE
ZOOM	0 1 0 0	0 1 1 0
CURS	0 1 0 0	1 0 0 1
PRAM	0 1 1 1	SA
PITCH	0 1 0 0	0 1 1 1
WDAT	0 0 1	TYPE 0 MOD
MASK	0 1 0 0	1 0 1 0
FIGS	0 1 0 0	1 1 0 0
FIGD	0 1 1 0	1 1 0 0
GCHRD	0 1 1 0	1 0 0 0
RDAT	1 0 1	TYPE 0 MOD
CURD	1 1 1 0	0 0 0 0
LPRD	1 1 0 0	0 0 0 0
DMAR	1 0 1	TYPE 1 MOD
DMAW	0 0 1	TYPE 1 MOD

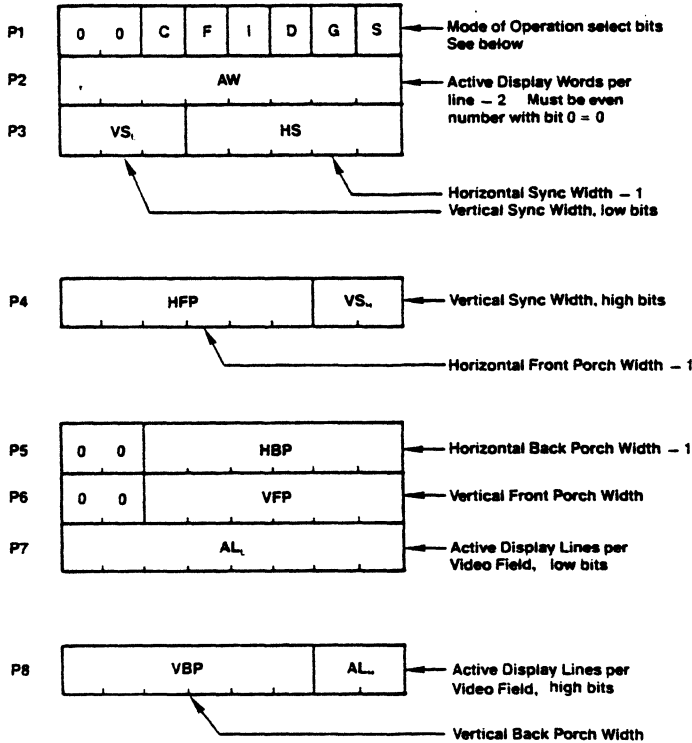
Video Control Commands

Reset



Blank the display, enter idle mode, and initialize within the GDC:
 — FIFO
 — Command Processor
 — Internal Counters

This command can be executed at any time and does not modify any of the parameters already loaded into the GDC. If followed by parameter bytes, this command also sets the sync generator parameters as described below. Idle mode is exited with the START command.



In graphics mode, a word is a group of 16 pixels. In character mode, a word is one character code and its attributes, if any. The number of active words per line must be an even number from 2 to 256. An all-zero parameter value selects a count equal to 2^n where n = number of bits in the parameter field for vertical parameters. All horizontal widths are counted in display words. All vertical intervals are counted in lines.

Horizontal Back Porch Constraints

- In general:
HBP \geq 3 Display Word Cycles (6 clock cycles).
- If the IMAGE or WD modes change within one video field:
HBP \geq 5 Display Word Cycles (10 clock cycles).
- If interlace or mixed mode is used:
HBP \geq 5 Display Word Cycles (10 clock cycles).

Horizontal Front Porch Constraints

- If the display ZOOM function is used at other than 1X:
HFP \geq 2 Display Word Cycles (4 clock cycles).
- If the GDC is used in the video sync Slave mode:
HFP \geq 4 Display Word Cycles (8 clock cycles).
- If the Light Pen is used:
HFP \geq 6 Display Word Cycles (12 clock cycles).
- If interlace mode is used:
HFP \geq 3 Display Word Cycles (6 clock cycles).

Horizontal SYNC Constraints

- If Interlaced display mode is used:
HS \geq 5 Display Word Cycles (10 clock cycles).

Modes of Operation Bits

C	G	Display Mode
0	0	Mixed Graphics & Character
0	1	Graphics Mode
1	0	Character Mode
1	1	Invalid

I	S	Video Framing
0	0	Noninterlaced
0	1	Invalid
1	0	Interlaced Repeat Field for Character Displays
1	1	Interlaced

Repeat Field Framing: 2 Field Sequence with 1/2 line offset between otherwise identical fields.

Interlaced Framing: 2 Field Sequence with 1/2 line offset. Each field displays alternate lines.

Noninterlaced Framing: 1 field brings all of the information to the screen.

Total scanned lines in interlace mode is odd. The sum of VFP + VS + VBP + AL should equal one less than the desired odd number of lines.

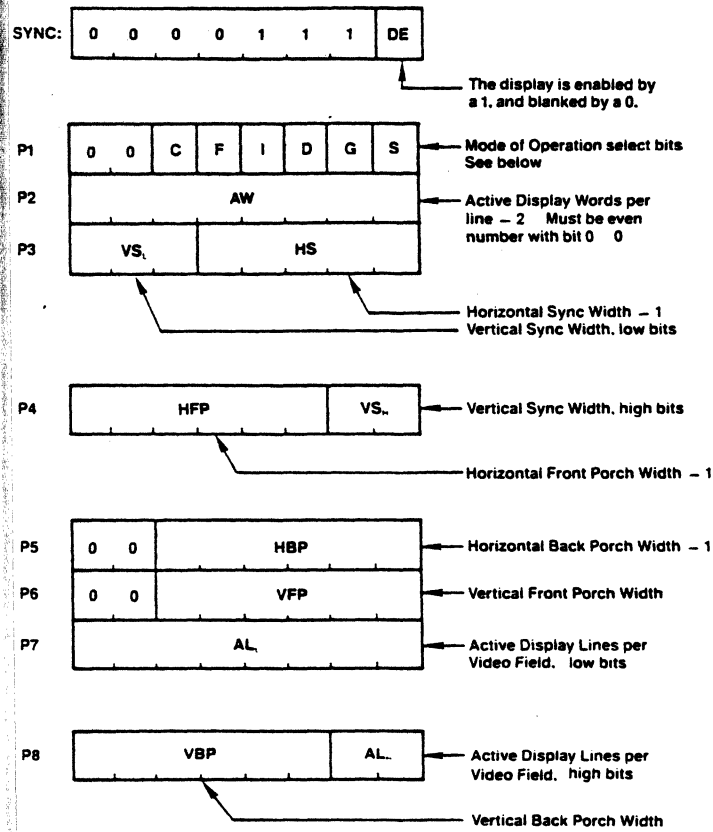
D	Dynamic RAM Refresh Cycles Enable
0	No Refresh — STATIC RAM
1	Refresh — Dynamic RAM

Dynamic RAM refresh is important when high display zoom factors or DMA are used in such a way that not all of the rows in the RAMs are regularly accessed during display raster generation and for otherwise inactive display memory.

F	Drawing Time Window
0	Drawing during active display time and retrace blanking
1	Drawing only during retrace blanking

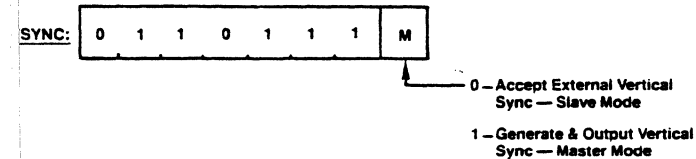
Access to display memory can be limited to retrace blanking intervals only, so that no disruptions of the image are seen on the screen.

SYNC Format Specify



This command also loads parameters into the sync generator. The various parameter fields and bits are identical to those at the RESET command. The GDC is not reset nor does it enter idle mode.

Vertical Sync Mode



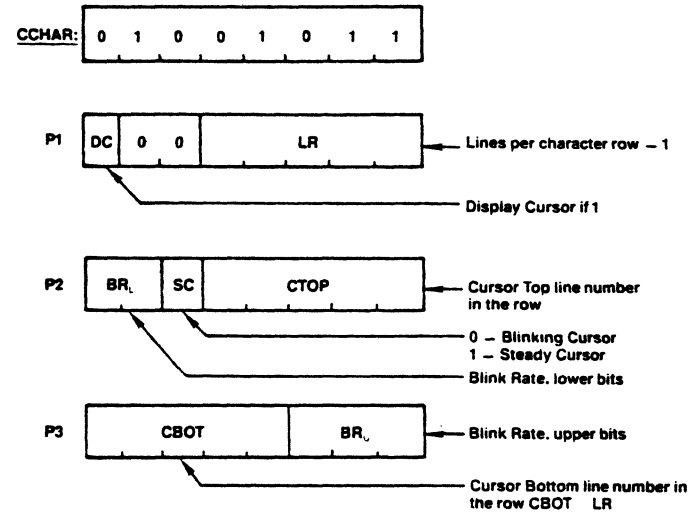
When using two or more GDCs to contribute to one image, one GDC is defined as the master sync generator, and the others operate as its slaves. The VSYNC pins of all GDCs are connected together.

A few considerations should be observed when synchronizing two or more GDCs to generate overlaid video via the VSYNC INPUT/OUTPUT pin. As mentioned above, the horizontal front porch (HFP) must be 4 or more display cycles wide. This is equivalent to eight or more clock cycles. This gives the slave GDCs time to initialize their internal video sync generators to the proper point in the video field to match the incoming vertical sync pulse (VSYNC). This resetting of the generator occurs just after the end of the incoming VSYNC pulse, during the HFP interval. Enough time during HFP is required to allow the slave GDC to complete the operation before the start of the SYNC interval.

Once the GDCs are initialized and set up as Master and Slaves, they must be given time to synchronize. It is a good idea to watch the VSYNC status bit of the Master GDC and

wait until after one or more VSYNC pulses have been generated before the display process is started. The START command will begin the active display of data and will end the video synchronization process, so be sure there has been at least one VSYNC pulse generated for the Slaves to synchronize to.

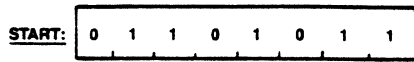
Cursor & Character Characteristics



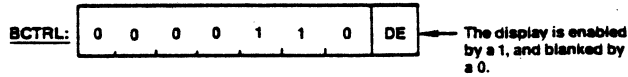
In graphics mode, LR should be set to 0. The blink rate parameter controls both the cursor and attribute blink rates. The cursor blink-on time = blink-off time = $2 \times BR$ (video frames). The attribute blink rate is always $\frac{1}{2}$ the cursor rate but with a $\frac{3}{4}$ on- $\frac{1}{4}$ off duty cycle. All three parameter bytes must be output for interlace displays, regardless of mode. For interlace displays in graphics mode, the parameter $BR_L = 3$.

Display Control Commands

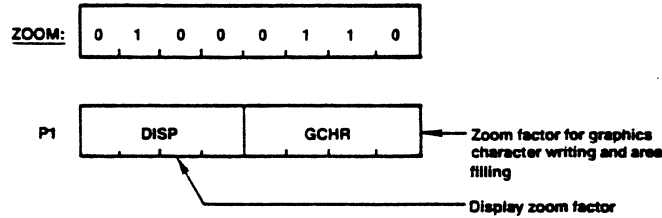
Start Display & End Idle Mode



Display Blanking Control

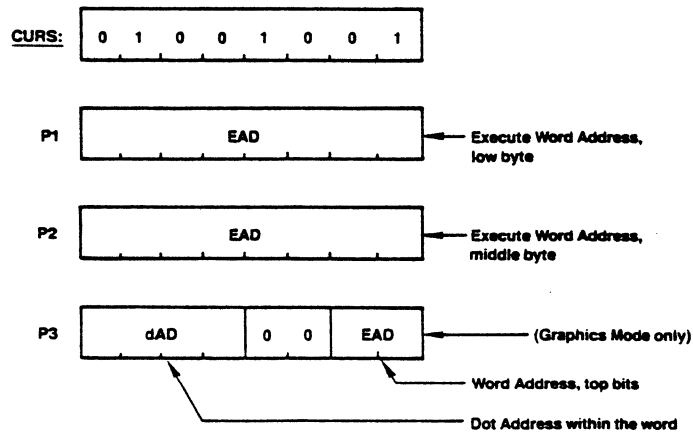


Zoom Factors Specify



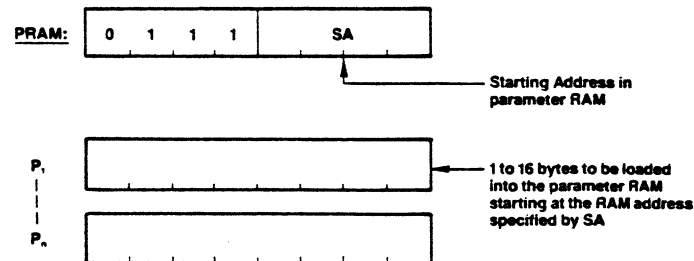
Zoom magnification factors of 1 through 16 are available using codes 0 through 15, respectively.

Cursor Position Specify



In character mode, the third parameter byte is not needed. The cursor is displayed for the word time in which the display scan address (DAD) equals the cursor address. In graphics mode, the cursor word address specifies the word containing the starting pixel of the drawing; the dot address value specifies the pixel within that word.

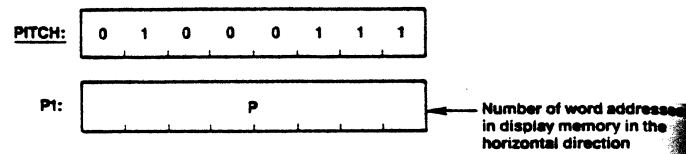
Parameter RAM Load



From the starting address, SA, any number of bytes may be loaded into the parameter RAM at incrementing addresses, up to location 15. The sequence of parameter bytes is terminated by the next command byte entered into

the FIFO. The parameter RAM stores 16 bytes of information in predefined locations which differ for graphics and character modes. See the parameter RAM discussion for bit assignments.

Pitch Specification

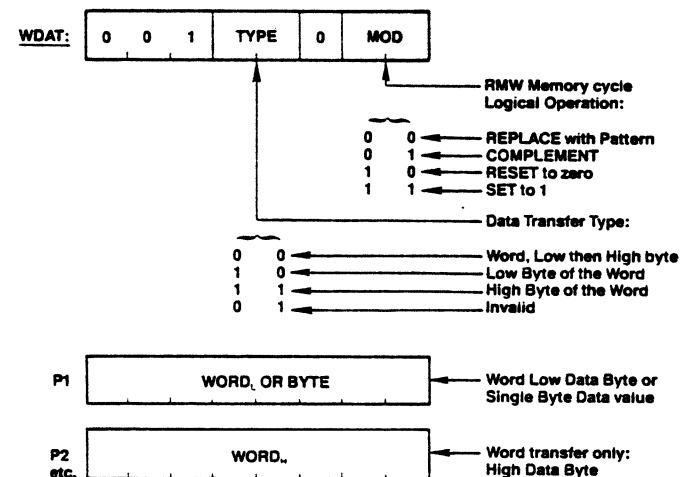


This value is used during drawing by the drawing process to find the word directly above or below the current word, and during display to find the start of the next line.

The Pitch parameter (width of display memory) is set by two different commands. In addition to the PITCH command, the RESET (or SYNC) command also sets the pitch value. The "active words per line" parameter, which specifies the width of the raster-scan display, also sets the Pitch of the display memory. Note that the AW value is two less than the display window width. The PITCH command must be used to set the proper memory width larger than the window width.

Drawing Control Commands

Write Data into Display Memory



Upon receiving a set of parameters (two bytes for a word transfer, one for a byte transfer), one RMW cycle into Video Memory is done at the address pointed to by the cursor EAD. The EAD pointer is advanced to the next word, according to the previously specified direction. More parameters can then be accepted.

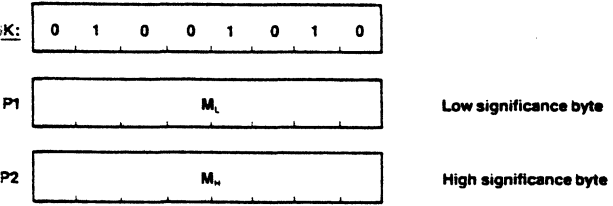
For byte writes, the unspecified byte is treated as all zeros during the RMW memory cycle.

In graphics bit-map situations, only the LSB of the WDAT parameter bytes is used as the pattern in the RMW operations. Therefore it is possible to have only an all ones or all zeros pattern. In coded character applications all the bits of the WDAT parameters are used to establish the drawing pattern.

The WDAT command operates differently from the other commands which initiate RMW cycle activity. It requires

Parameters to set up the Pattern register while the other commands use the stored values in the parameter RAM. In all of these commands, the WDAT command must be preceded by a FIGS command and its parameters. Only the first three parameters need be given following the FIGS command, to set up the type of drawing, the DIR direction, and the DC value. The DC parameter +1 will be the number of RMW cycles done by the GDC with the first set of WDAT parameters. Additional sets of WDAT parameters will see a DC value of 0 which will cause only one RMW cycle to be executed per set of parameters.

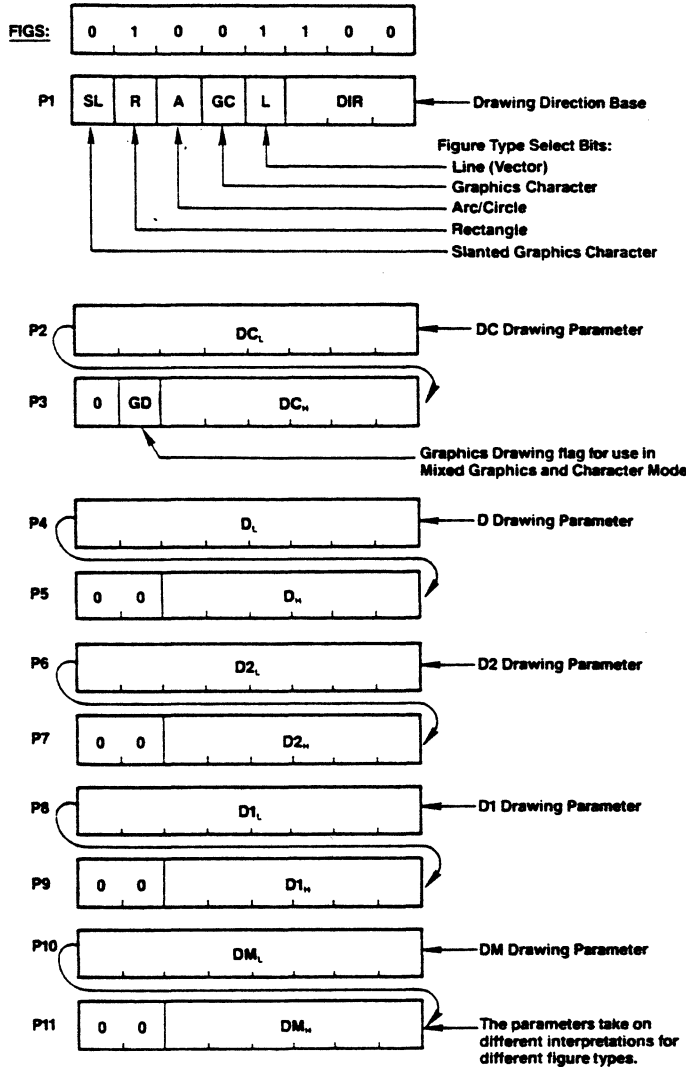
Mask Register Load



The MASK command sets the value of the 16-bit Mask register of the figure drawing processor. The Mask register controls which bits can be modified in the display memory during a read-modify-write cycle.

The Mask register is loaded both by the MASK command and the third parameter byte of the CURS command. The MASK command accepts two parameter bytes to load a 16-bit value into the Mask register. All 16 bits can be individually one or zero, under program control. The CURS command on the other hand, puts a "1 of 16" pattern into the Mask register based on the value of the Dot Address Register, dAD. If normal single-pixel-at-a-time graphics figure drawing is desired, there is no need to do a MASK command at all since the CURS command will set up the proper pattern to address the proper pixels as drawing progresses. For coded character DMA, and screen setting and clearing operations using the WDAT command, the MASK command should be used after the CURS command if its third parameter byte has been output. The Mask register could be set to all "ONES" for any "word-at-a-time" operation.

Figure Drawing Parameters Specify

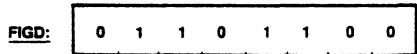


Valid Figure Type Select Combinations

SL	R	A	GC	L	Operation
0	0	0	0	0	Character Display Mode Drawing, Individual Dot Drawing, DMA, WDAT, and RDAT
0	0	0	0	1	Straight Line Drawing
0	0	0	1	0	Graphics Character Drawing and Area filling with graphics character pattern
0	0	1	0	0	Arc and Circle Drawing
0	1	0	0	0	Rectangle Drawing
1	0	0	1	0	Slanted Graphics Character Drawing and Slanted Area Filling

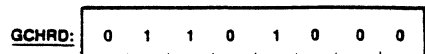
Only these bit combinations assure correct drawing operation.

Figure Draw Start



On execution of this instruction, the GDC loads the parameters from the parameter RAM into the drawing processor and starts the drawing process at the pixel pointed to by the cursor, EAD, and the dot address, dAD.

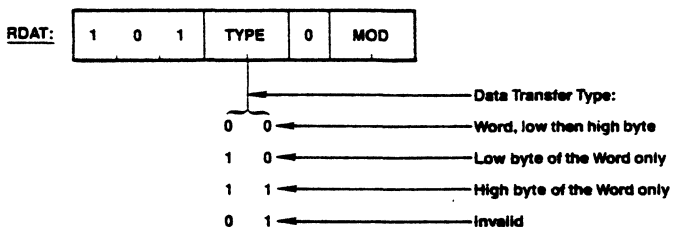
Graphics Character Draw and Area Filling Start



Based on parameters loaded with the FIGS command, this command initiates the drawing of the graphics character or area filling pattern stored in Parameter RAM. Drawing begins at the address in display memory pointed to by the EAD and dAD values.

Data Read Commands

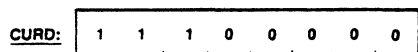
Read Data from Display Memory



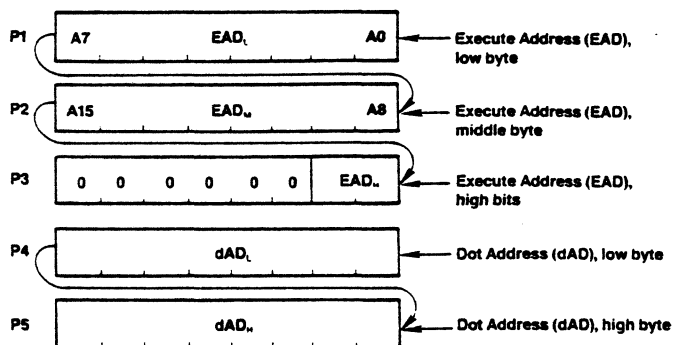
Using the DIR and DC parameters of the FIGS command to establish direction and transfer count, multiple RMW cycles can be executed without specification of the cursor address after the initial load (DC = number of words or bytes).

As this instruction begins to execute, the FIFO buffer direction is reversed so that the data read from display memory can pass to the microprocessor. Any commands or parameters in the FIFO at this time will be lost. A command byte sent to the GDC will immediately reverse the buffer direction back to write mode, and all RDAT information not yet read from the FIFO will be lost. MOD should be set to 00 if no modification to video buffer is desired.

Cursor Address Read



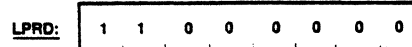
The following bytes are returned by the GDC through the FIFO:



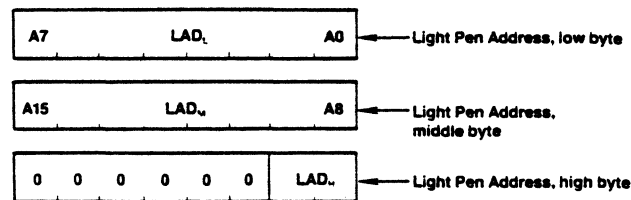
The Execute Address, EAD, points to the display memory word containing the pixel to be addressed.

The Dot Address, dAD, within the word is represented as a 1-of-16 code for graphics drawing operations.

Light Pen Address Read



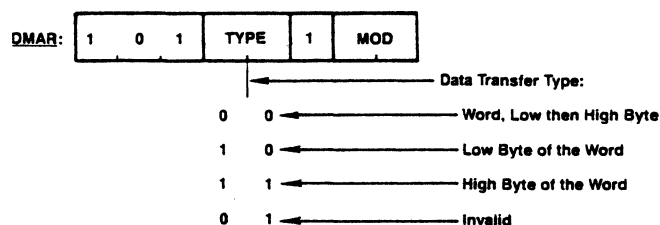
The following bytes are returned by the GDC through the FIFO:



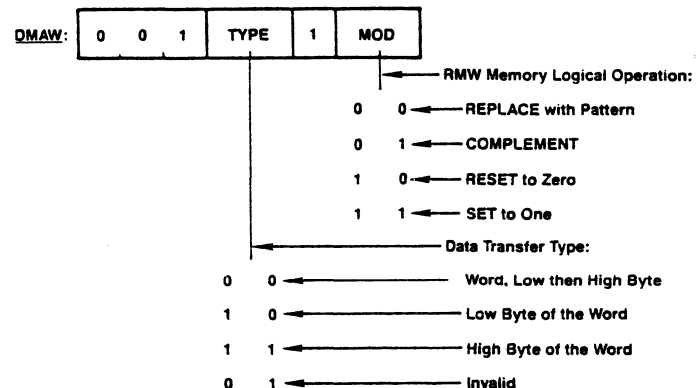
The light pen address, LAD, corresponds to the display word address, DAD, at which the light pen input signal is detected and deglitched.

The light pen may be used in graphics, character, or mixed modes but only indicates the word address of light pen position.

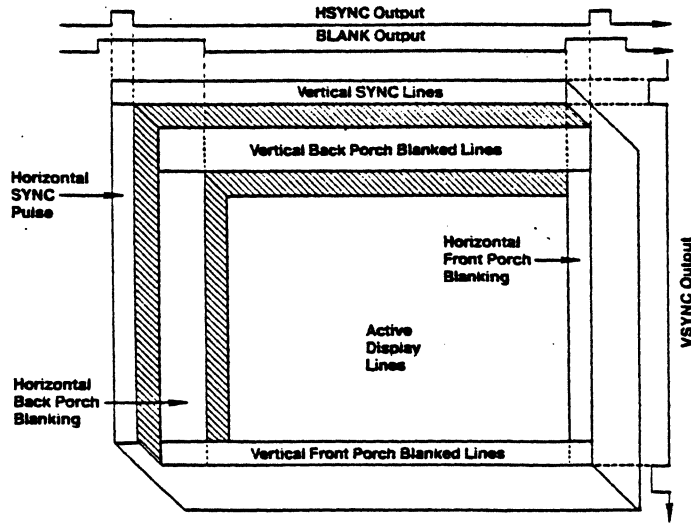
DMA Read Request



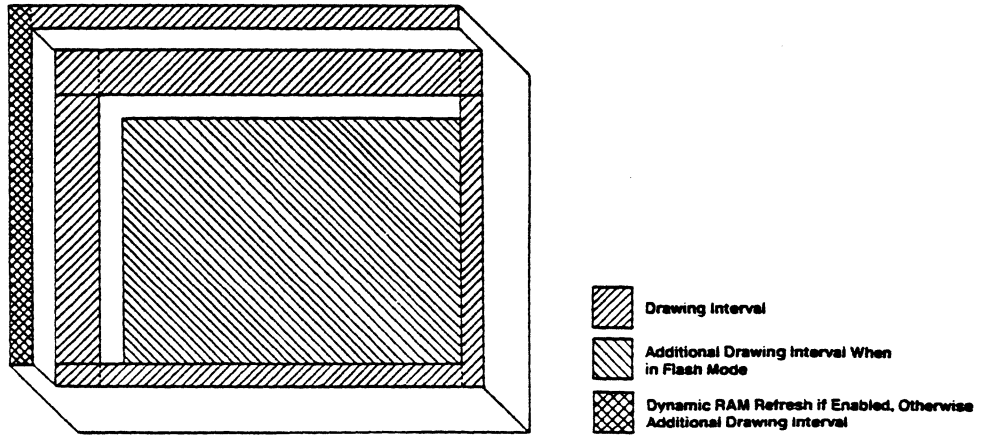
DMA Write Request



Video Field Timing



Drawing Intervals



MA Request Intervals

