JPL PUBLICATION 84-89

The Multimission Image Processing Laboratory's Virtual Frame Buffer Interface

T.L. Wolfe

December 15, 1984

NASA
National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 84-89

The Multimission Image Processing Laboratory's Virtual Frame Buffer Interface

T.L. Wolfe

December 15, 1984

NASA
National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

CONTENTS

Figures

Tables

Abstract

Large image processing systems use multiple frame buffers with differing architectures and vendor supplied interfaces. This variety of architectures and interfaces creates software development, maintenance and portability problems for application programs. Several machine-independent graphics standards such as ANSI Core and GKS are available, but none of them are adequate for image processing. Therefore the Multimission Image Processing Laboratory project has implemented a programmer level virtual frame buffer interface. This interface makes all frame buffers appear as a generic frame buffer with a specified set of characteristics. This document defines the virtual frame buffer interface and provides information such as FORTRAN subroutine definitions, frame buffer characteristics, sample programs, etc. It is intended to be used by application programmers and system programmers who are adding new frame buffers to a system.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Large image processing systems today have multiple frame buffers with differing architectures and vendor supplied interfaces. This variety, while necessary for various jobs, creates a software development, maintenance and portability problem. At the start of the Multimission Image Processing Laboratory (MIPL) project, it was apparent that a machine-independent interface for the various existing and proposed frame bufffers could eliminate or reduce the problems caused by the different devices.

Several machine-independent computer graphics standards such as ANSI Core and GKS were proposed, but none of them adequately addressed the needs of image processing. They allowed operations such as drawing vectors and polygon fills to be machine independent but did not (and still do not) acccess pixels in ways useful for image processing. Both of the major standards (Core and GKS) have standarized escapes to perform operations not covered by the standard, but escaping from the standard leaves one without a standard.

The solution chosen by the MIPL project was a virtual frame buffer. This is a programmer level interface that makes all frame buffers appear as a generic frame buffer with a specified set of characteristics. The virtual frame buffer interface converts generic commands to actual device commands.

The virtual frame buffer characteristics or functions are adequate for over 90% of all application programs and are dynamic in nature. Further capabilities will be added as programming needs change and/or as new hardware is acquired.

It is obvious that a virtual frame buffer interface can not make all frame buffers look exactly the same, especially for the more esoteric hardware capabilities. Programs can query the virtual

interface to determine if a device has a given capability and if not, the program can abort or can simulate the capability in software. Some capabilities are currently simulated in software by the virtual interface for some devices.

The virtual frame buffer consists of a definition of capabilities (see Chapter 2) and FORTRAN subroutines (see Chapter 3) that are called by application programs. The subroutines for the various "real" frame buffers are in separate VAX/VMS shared libraries. This allows the modification, correction or enhancement of the virtual interface without affecting application programs. Included in these libraries are the manufacturer provided routines that access the device directly. Programs can use the virtual interface up to a point and then access the device directly, if necessary.

Using the virtual frame buffer interface does not preclude the use of any of the proposed standards in the future. It could replace the virtual interface or be layered above or below it (see Fig. 1-1).

Besides the virtual frame buffer interface, other system capabilities are needed to make the use of frame buffers easy. These other system capabilities are discussed briefly in this document, and what has already been implemented is indicated.

1.2 THE DISPLAY SYSTEM

The display system consists of a number of physical and logical devices and software. The devices and software interact to give the user a simple and effective way to manipulate images.

The hardware consists of a variety of frame buffers, monitors, video switches, and intractive I/O devices such as trackballs, tablets, joysticks, etc. These devices make up a pool of hardware resources that may be allocated by users to build image processing workstations that fit job requirements.

The software consists of programs and commands that allocate and manage hardware resources and a virtual frame buffer interface. Together the hardware and software make up the MIPL display system.

1.3 IMAGE PROCESSING WORKSTATION

An image processing workstation is a physical location containing a table, chair, terminal, and some number of monitors, interactive I/O devices and virtual frame buffers (see Fig. 1-2). With one command (or a few) users can allocate workstations with the resources they need to do a given image processing task.

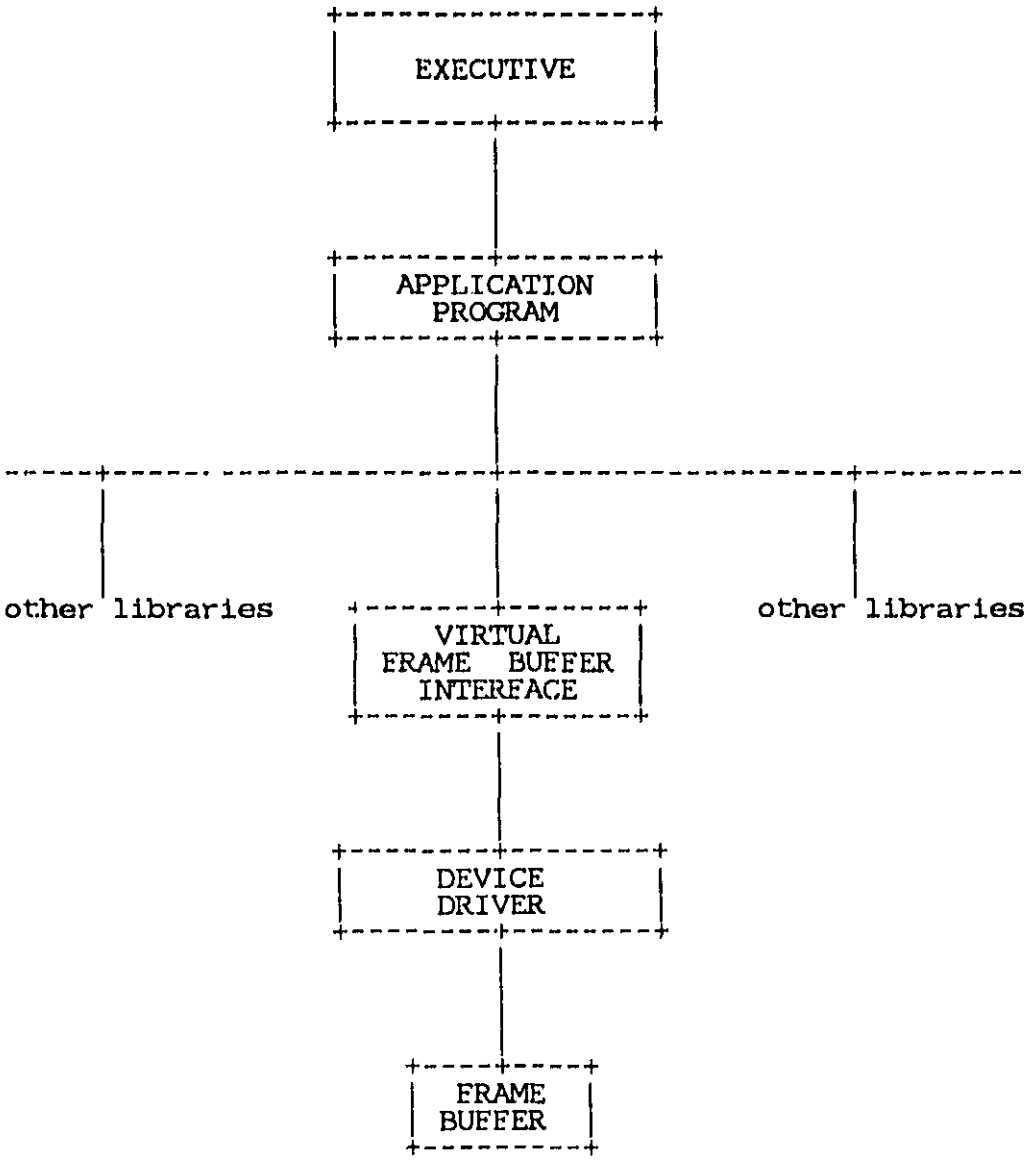

Fig. 1-1. Software System Block Diagram

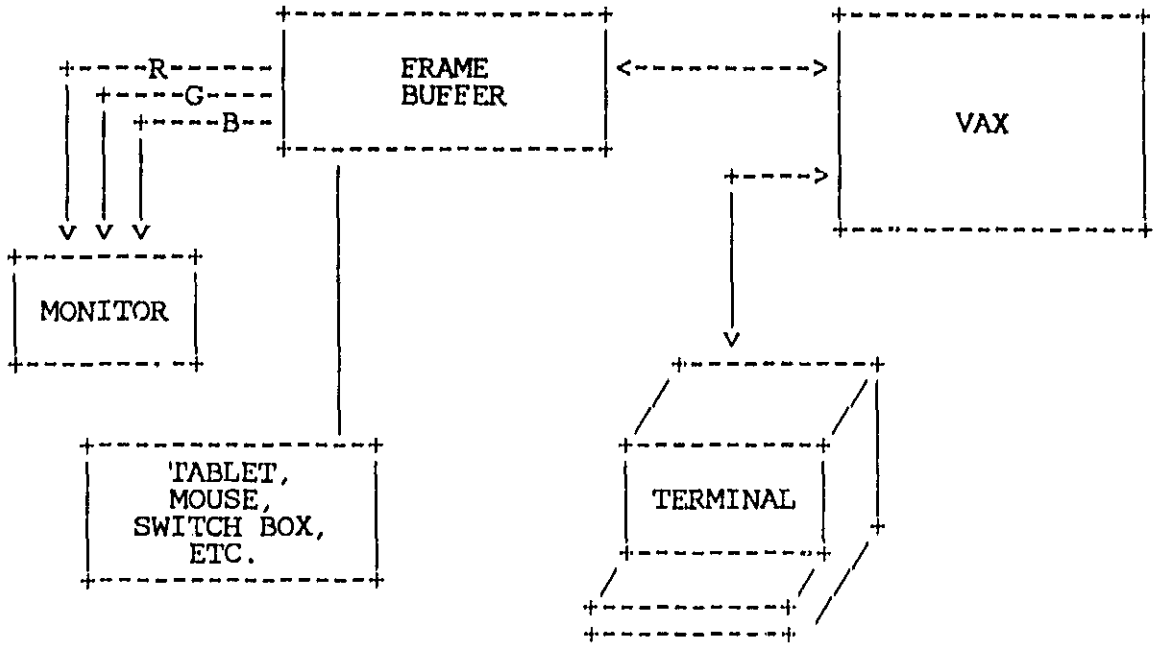

Fig. 1-2. Image Processing Workstation

Hardware may be allocated for a workstation piece by piece or by a standard configuration. Standard workstation configurations such as full color, pseudocolor or monochrome may be allocated by the user. The number of image memory planes, the video resolution (high or low), the number and types of monitors, and the number and types of interactive I/O devices are all part of a standard configuration. Standard workstations simplify the users' need to know and interact with the system. Users just allocate a workstation that fits their current needs.

1.4 RESOURCE MANAGER

The MIPL display system as described requires a sophisticated resource manager to manage and control access to the hardware. It needs to connect frame buffers to monitors (see Fig. 1-3) and interactive I/O devices and keep track of who has what and what is available. Users should be able to query the resource manager to determine what resources are free for their use.

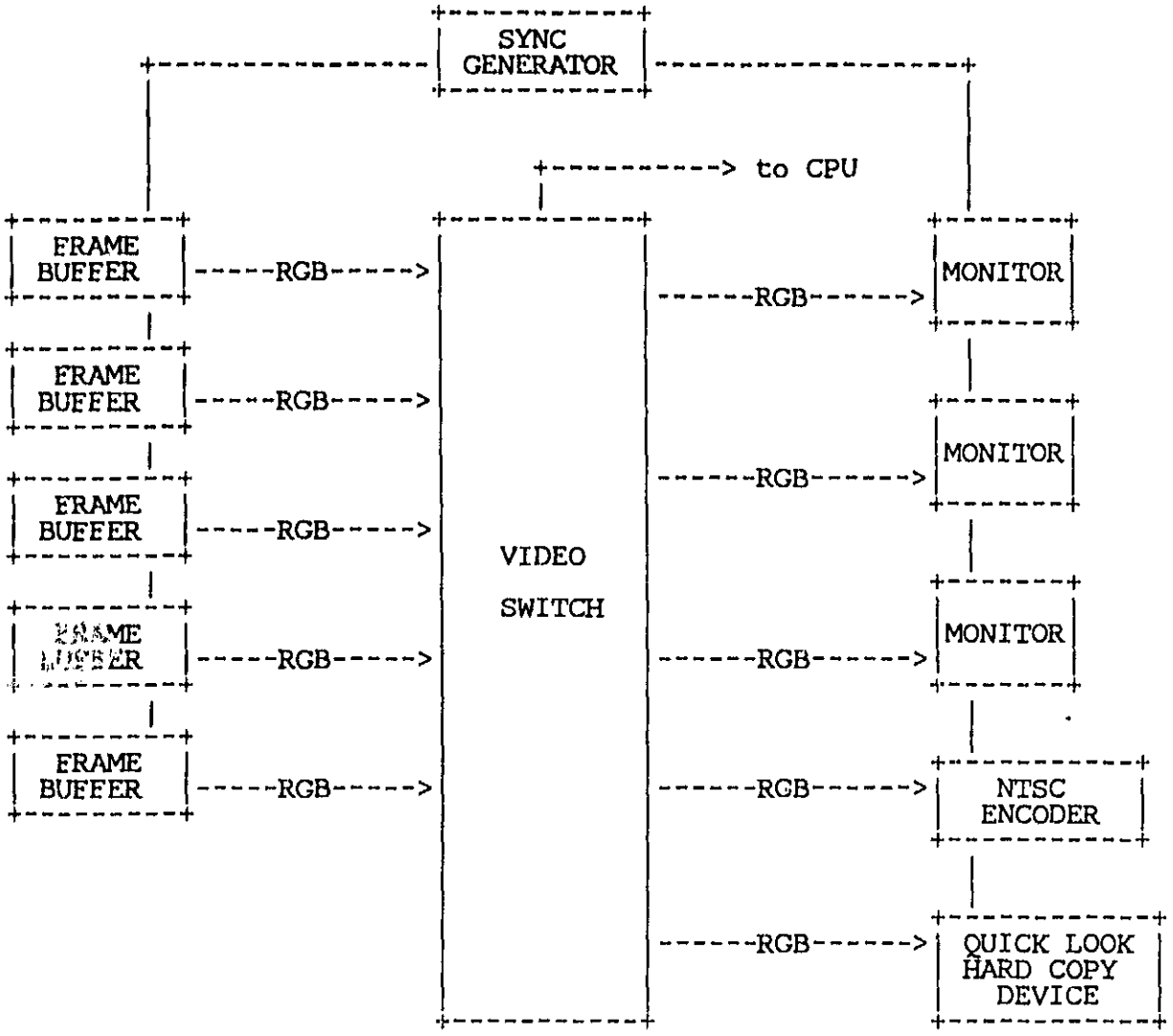

Fig. 1-3. Video Switch System

Currently only a primitive frame buffer resource manager has been implemented. It allocates complete frame buffers, provides information on frame buffer availability and allocates default frame buffers. Some terminals have been assigned default frame buffers. On these terminals a user may allocate or deallocate the "DEFAULT" frame buffer. Currently all frame buffers are "hardwired" to monitors and interactive I/O devices.

CHAPTER 2

VIRTUAL FRAME BUFFER DEFINITION

2.1 FRAME BUFFER CONFIGURATIONS

Within the virtual frame buffer interface software, a frame buffer may be configured into one of several standard configurations depending on the available hardware. Full color, pseudocolor and monochrome are the standard device configurations and all have the following default characteristics:

1. The access window of all image memory planes is set to the complete image memory plane.

2. The display window of all image memory planes is moved to the upper left hand corner of each image memory plane (coordinates 1,1).

3. All cursors are set to type 1, no auto tracking and "off."

4. The alphanumeric font generator is turned off.

5. The display of the graphics overlay plane is turned off.

Each of the three standard device configurations has some additional characteristics:

1. For full color devices, image memory plane 1 is connected to LUT 1 (red), 2 to LUT 2 (green) and 3 to LUT 3 (blue). Image memory plane 4 is connected to the graphics overlay plane LUT. All LUTs are set to no bypass and LUT section 1.

2. For pseudocolor devices, image memory plane 1 is connected to LUT 1 (red), LUT 2 (green) and LUT 3 (blue). Image memory plane 2 is connected to the graphics overlay LUT. All LUTs are set to no bypass and LUT section 1.

3. For monochrome devices, image memory plane 1 is connected to LUT 1 (or all three LUTs in full color displays). Image memory plane 2 is connected to the graphics overlay plane LUT. All LUTs are set to no bypass and LUT section 1.

2.2 COORDINATE SYSTEM

The coordinate system chosen for the virtual frame buffer is the standard line/sample convention currently used for MIPL images. The upper left-hand corner of the image memory as seen on the monitor is coordinate (1,1) (Note: not [0,0]). Coordinates increase downward and to the right. Coordinates are integer values that are locations of pixels in the image memory plane.

2.3 IMAGE MEMORY PLANES

An image memory plane (IMP) in the virtual frame buffer is a two-dimensional array of 8-bit pixels. This limits the number of different pixel values that may be displayed to 256 (0 to 255). Image memory planes are logical entities having a given size and are numbered starting with one (1). The display software maintains the connection between logical and physical image memory planes. A user may query the system to obtain information about the number and size of the image memory planes in a given frame buffer since some frame buffers allow the image memory planes to be configured into several sizes. Whatever the configuration, all image memory planes in the frame buffer are the same size.

2.4 LOOK UP TABLES

Each look up table (LUT) is connected to a Digital to Analog Converter (DAC) that sends a video signal to a monitor. By convention, LUT 1 is assumed to send the RED signal for full color and pseudocolor images. LUT 2 sends GREEN, and LUT 3 sends BLUE.

LUTs may be bypassed (pixel values sent directly to the DAC), or the pixel value may be passed through the LUT for conversion to another value before being sent to the DAC. LUTs may also have more than one section or conversion table. Although each section or table in a LUT has 256 entry points (one for each possible pixel value) each may have more than an 8-bit output value. This difference is sometimes used to load the LUT with a more accurate gamma correction curve than 8-bit will provide. The user may query the system to obtain information about the number of sections in each LUT and the size of the output value. Sections are numbered starting with one (1).

2.5 GRAPHICS OVERLAY PLANE

Displaying graphics overlay data is logically the same as displaying data in any image memory plane. The graphics overlay plane and LUT may be thought of as a pseudocolor frame buffer. Any image memory plane may be used as a graphics overlay plane by connecting it to the graphics overlay LUT. The graphics overlay LUT consists of three tables (red, green and blue) that look and act exactly like the regular LUT tables. The overlay process is accomplished by replacing output pixels with overlay pixels. This takes place only if the graphics pixel has a nonzero value. The display of graphics overlay data may be turned on or off under software control. Like the regular LUTs, the graphics overlay LUT may be bypassed. In some frame buffers, displaying graphics overlay data is not handled in this way. Instead, the software tries to simulate the operation.

2.6 FRAME BUFFER UNIT NUMBERS

Provisions have been made in the design of the virtual interface to allow a user to allocate any number of frame buffers. The user accesses each device by a unique unit number assigned by the user at allocation time. Valid unit numbers are positive integers starting at one (1). Unit number zero (0) has a special meaning. Actions performed on unit zero are performed on all display devices allocated to the user that are "active."

NOTE: Currently, users can allocate only a single frame buffer; unit numbers are ignored.

2.7 INTERACTIVE I/O DEVICES

Users may allocate any number of interactive I/O devices (joysticks, tablets, etc.). In general, however, users will allocate the one(s) near their workstations. A user accesses each device by a unique unit number assigned by the user at allocation time. Valid unit numbers are positive integers starting with one (1).

NOTE: Currently, interactive I/O devices are hard-wired to a particular frame buffer and may not be used by any other device.

Because of the variety of absolute and incremental interactive I/O devices which the virtual interface is expected to support, the returned values have been normalized to the range of -1.0 to +1.0. This will simplify application programs that do not need to know the particular device they are accessing.

2.8 MONITORS

Display monitors are connected to the system via a video switcher that is under software control. The user allocates a display monitor to a particular display device. Valid monitor numbers are positive integers starting with one (1). The system maintains a table of monitor characteristics. When a monitor is allocated, this table is checked to determine if it is compatible with the display device. Monitors come in two sizes, low resolution (approximately 512 x 512) and high resolution (approximately 1024 x 1024), and two types of color, color (red, green, blue) and monochrome.

NOTE: Currently, the video switcher is not under software control. Monitors are associated with a particular frame buffer.

2.9 SUBROUTINE NAMING CONVENTION

The first two letters in the name of the display interface routines are always 'XD'. The third letter indicates the major hardware or function associated with the routine:

A - Alphanumeric Font Generator
C - Cursor Generator
D - The Complete Display Device
G - Graphics Overlay Plane
I - Image Memory Plane
T - Text Generation in Image Memory Planes
X - Interactive I/O Devices
(tablet, joystick, etc.)

The rest of the routine name tries to give some idea of what the routine does.

The text generation routines are separate from the image memory plane routines because there are so many IMP routines and because they interact in a special way.

2.10 SUBROUTINE RETURN CODES

The virtual frame buffer interface routines are functions. They may be used as subroutines by just calling them or as functions by using them in expressions (they must be declared LOGICAL).

If used as functions, the returned status values are designed to allow the user to use the value as a logical value or as an integer indicating the completion status of the function.

The VAX uses the least significant bit of a value as a Boolean flag to indicate TRUE or FALSE. All routines return TRUE (1) if successfully completed. They return FALSE (even integers, least significant bit 0) if an error has occurred. The application program may then use this returned value as a logical or integer value. This allows a great deal of flexibility in the application programs as to how errors are handled. The returned error codes are listed with the description of each of the virtual frame buffer routines in Chapter 3.

2.11 TEXT GENERATION

In most cases the MIPL frame buffers do not have alphanumeric font generators because they are of limited value for our needs. Instead, a series of routines is supplied that will allow the user to write text directly into the image memory planes as pixel values. In order not to modify the image, the text can be written into the graphics overlay plane.

The Hershey character fonts are supplied to give the user several choices. Users may also design their own fonts. See Appendix C for further details.

2.12 ACCESS WINDOW

Each image memory plane has associated with it an access window. The access window defines the area of an image memory plane where the pixels may be modified. Pixels outside the access window can not be modified by the user.

2.13 DISPLAY WINDOW

Each image memory plane has associated with it a display window. The display window coordinates define the upper left corner of the array of pixels scanned by the DACs. Only the upper left corner is defined because the size of the window displayed on the monitor is determined by the hardware. Some hardware allows the size of the window to be changed dynamically.

2.14 VIRTUAL FRAME BUFFER

Many "real" frame buffers do not have all of the capabilities defined by the virtual frame buffer described in this document. In these cases the software simulates the required functions. With

some frame buffers, it is not even possible to simulate a function. In these cases the software just does the best it can.

The following is a description of the virtual frame buffer defined by MIPL software (see Fig. 2-1):

1. A number of 8-bit image memory planes. The approximate sizes are 512 x 512, 1024 x 1024, 2048 x 2048 and 4096 x 4096. Image memory planes of larger sizes may be configured into smaller ones. Each image memory plane has associated with it an access window and display window. Any image memory plane can be connected to any or all LUTs/DACs. including the graphics overlay LUT.

2. Each display device has one or more (three for full color) LUTs/DACs. Each LUT contains one or more pixel transformation tables (sections). Each table has 256 entries, one for every possible pixel value. LUTs may be bypassed under software control, allowing pixel values to be sent directly to the DACs. Which LUT section is used for pixel conversion is also under software control. LUTs/DACs are independent of each other and have their own zoom factor.

3. The graphics overlay LUT is different than the standard LUTs in that it has three tables (red, green and blue in full color devices). Graphics overlay works by pixel substitution. If graphics overlay is "ON", nonzero overlay pixels are substituted for regular pixels before going to the DACs. In this way graphics overlay data is written on top of the output image. Any of the image memory planes may be connected to the graphics overlay LUT.

4. The alphanumeric font generator works the same as the graphics overlay (pixel substitution). Font information is written on top of graphics overlay data. Many MIPL frame buffers do not have a font generator because of its limited usefulness.

5. One or more cursors are available. Cursors also work by pixel substitution and are written on top of the graphics overlay data. Each cursor has a number of forms and blink rates. The cursor form also includes the color of the cursor. The center of the cursor pattern may be moved over every pixel in any image memory plane. The location of the cursor may be read or controlled by software.

6. The virtual frame buffer allows arithmetic and logical operations between image memory planes and allows area fill, histogram generation, drawing vectors, etc.

7. Software can read and write all LUT tables.

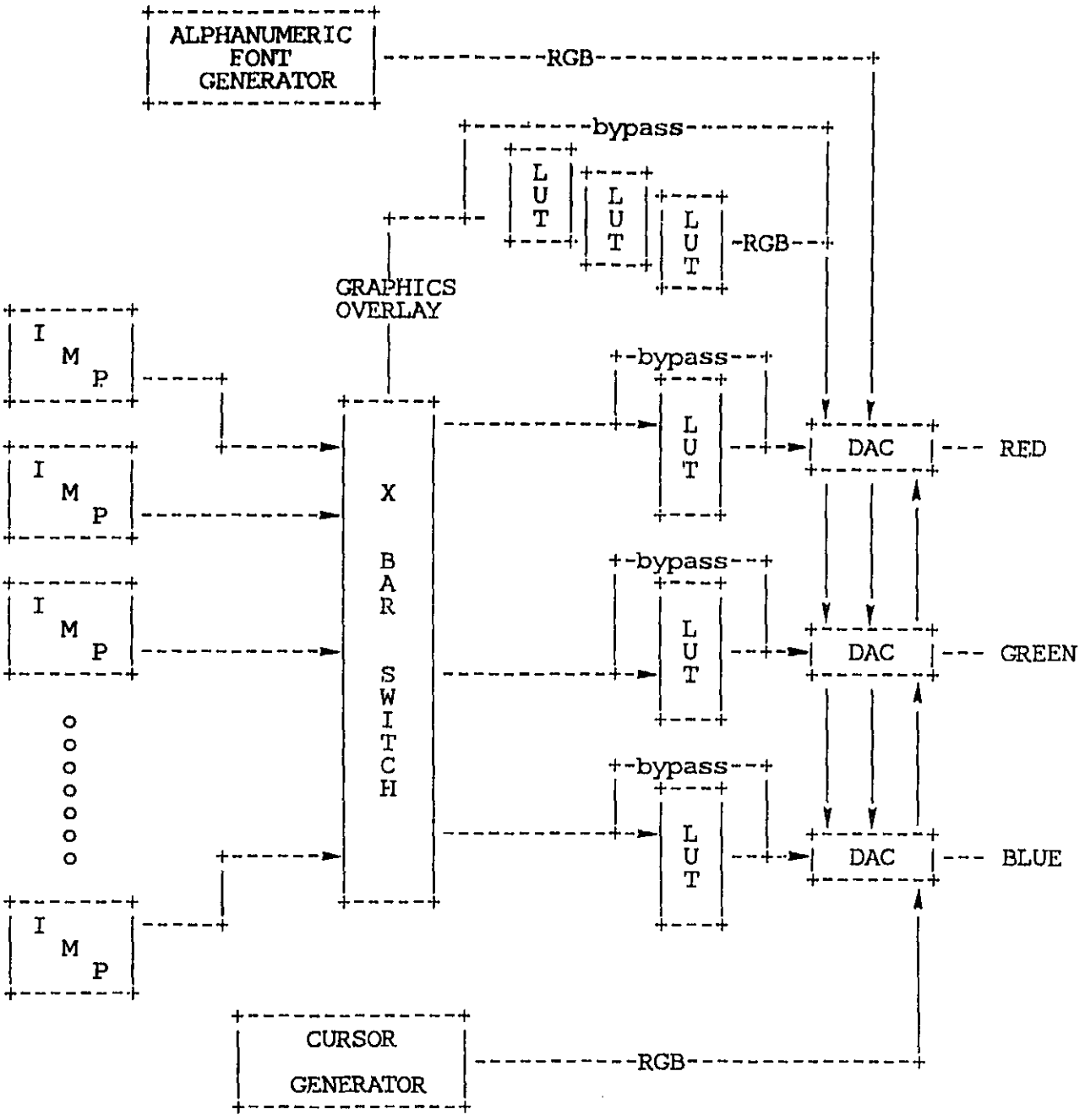

Fig. 2-1. Virtual Frame Buffer

8. Aspect ratios of 1 x 1 and 4 x 3 are available.

9. The location of the display window may be read and controlled by software. The display window may be moved under software control so that every pixel in an image memory plane can be seen.

10. Interactive I/O devices may be connected to any frame buffer. The cursor or the display window may be set to automatically track one of the interactive I/O devices.

11. Capabilities may be reduced by limitations in the actual hardware.

CHAPTER 3

VIRTUAL FRAME BUFFER SUBROUTINES

The virtual frame buffer routines may be treated as subroutines, logical functions, or integer functions by the application program (see Section 2.10 for more information).

It is important to note that currently the unit number "U" is ignored by all virtual frame buffer subroutines. A default value of 1 should be used.

The subroutines described in this chapter are in alphabetical order within major usage categories. For example, all of the routines that access image memory planes are grouped together alphabetically. The description of each subroutine consists of the subroutine name, some descriptive FORTRAN code, text description and return code listings. Other special information may also be included.

3.1 ALPHANUMERIC FONT GENERATOR

3.1.1 XDACLEAR

```
LOGICAL FUNCTION XDACLEAR (U,X,Y,N)
INTEGER*2 U,X,Y,N
```

Clear the text in the Alphanumeric Font Generator (AFG) of display unit U. Characters starting at location (X,Y) in AFG coordinates (line and position) are cleared. If N is zero or negative, all of the characters from (X,Y) to the end of the AFG memory are cleared. Clearing allows the image to be visible.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	AFG not available

3.1.2 XDAOFF

```
LOGICAL FUNCTION XDAOFF (U)
INTEGER*2 U
```

Turn off the display of the characters in the AFG of display unit U.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	AFG is not available

3.1.3 XDAON

```
LOGICAL FUNCTION XDAON (U,T)
INTEGER*2 U,T
```

Turn on the display of characters in the AFG of display unit U. The type T is a limited set of combinations of character color and background color. If T is zero, an implementation defined default value for T is used.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal T value

3.1.4 XDATEXT

```
LOGICAL FUNCTION XDATEXT (U,X,Y,N,STRING)
INTEGER*2  U,X,Y,N
BYTE       STRING(N)
```

Write text into the AFG of display unit U. Text is displayed starting at location (X,Y) in AFG coordinates (line and position).

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated
6	Illegal coordinates
8	AFG memory overflow (too many characters)

3.2 CURSOR

The hardware cursor generator may generate more than one cursor. The available cursors are numbered starting with one (1). Cursor number zero (0) is the default cursor, which is implementation dependent. The cursor may be available in several forms (see Appendix E).

3.2.1 XDCAUTRACK

```
LOGICAL FUNCTION XDCAUTOTRACK (U,C,X,F)
INTEGER*2 U,C,X
LOGICAL*2 F
```

Turn on or off the automatic tracking of the interactive I/O device X by the cursor C on unit U. F is TRUE for auto-tracking and FALSE for no auto-tracking. Auto-tracking means that the display unit automatically moves the cursor in response to the interactive I/O device X without utilizing the system CPU. In the no auto-tracking mode the cursor can be moved only on command from the application program. Parameter X may be the implementation defined default device zero (0).

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal cursor number
8	Illegal interactive I/O device
10	Auto-tracking not available

3.2.2 XDCLOCATION

```
LOGICAL FUNCTION XDCLOCATION (U,C,X,Y)
INTEGER*2 U,C,X,Y
```

Return the location (line,sample) of cursor C of display unit U in (X,Y).

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal cursor number

3.2.3 XDCOFF

```
LOGICAL FUNCTION XDCOFF (U,C)
INTEGER*2 U,C
```

Turn off cursor C of display unit U.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal cursor number

3.2.4 XDCON

```
LOGICAL FUNCTION XDCON (U,C,F,B)
INTEGER*2 U,C,F,B
```

Turn on cursor number C of display unit U. Set the form (implementation defined) of the cursor to F. Set the blink rate to B (implementation defined).

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal cursor number
8	Illegal cursor form
10	Illegal blink rate

3.2.5 XDCSET

```
LOGICAL FUNCTION XDCSET (U,C,X,Y)
INTEGER*2 U,C,X,Y
```

Move cursor C of display unit U to location (X,Y) (line,sample).

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal cursor number
8	Illegal coordinates

3.3 DEVICE CONFIGURATION

3.3.1 XDDACTIVATE

```
LOGICAL XDDACTIVATE (U,F)
INTEGER*2 U
LOGICAL   F
```

Turn on or off the modification of the display unit U. If the logical flag F is TRUE, modification of display unit U is allowed. If it is FALSE, modification is not allowed.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated

3.3.2 XDDALLOCATE

```
LOGICAL FUNCTION XDDALLOCATE (U,D)
INTEGER*2 U,D(1)
```

Allocate resources from a pool of display devices, image memory planes, interactive I/O devices, monitors, etc. Array D is a description of the resources to be allocated. The collection of requested resources defines a virtual display device and is given the user defined unit number U. All access to the virtual display device is through the unit number U, which must be a value between 1 and 8.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	No display device available
6	Unit number is already in use
8	Illegal request

NOTE: Currently, this routine takes no action. Values in array "D" are ignored. The routine is a planned future enhancement.

3.3.3 XDDCONFIGURE

```
LOGICAL FUNCTION XDDCONFIGURE (U,D)
INTEGER*2 U,D(4)
```

Set display unit U to a standard configuration defined by array D. The following define the possible standard configurations:

D(1) = 0	Default
D(1) = 1	Full Color Display
D(1) = 2	Pseudocolor Display
D(1) = 3	Monochrome Display
D(2) = 0	Default
D(2) = 1	512 x 512 Image Memory Planes (approx.)
D(2) = 2	1024 x 1024 Image Memory Planes (approx.)
D(2) = 3	2048 x 2048 Image Memory Planes (approx.)
D(2) = 4	4096 x 4096 Image Memory Planes (approx.)
D(3) = 0	Default
D(3) = 1	512 x 512 Video Output (approx.)
D(3) = 2	1024 x 1024 Video Output (approx.)
D(4) = 0	Default
D(4) = 1	1 x 1 Pixel Spacing Aspect Ratio
D(4) = 2	4 x 3 Pixel Spacing Aspect Ratio

If XDDCONFIGURE is not used, the configuration of the display device is undefined. The program must then set the display device configuration and internal tables. Not all frame buffers devices can be configured in all the above ways.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	D(1) is illegal for this device
8	D(2) is illegal for this device
10	D(3) is illegal for this device
12	D(4) is illegal for this device
14	Illegal combination of D values for this device
16	IMPs must be as big as the display window

3.3.4 XDDFREE

```
LOGICAL FUNCTION XDDFREE (U)
INTEGER*2 U
```

Free (deallocate) display unit U. After the display unit is freed, it is available for use use by others.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated

NOTE: Currently, this routine only marks the device as not available to the user. Frame buffer resources must be freed (deallocated) outside of the program.

3.3.5 XDDINFO

```
LOGICAL FUNCTION XDDINFO (U,S,N,A)
INTEGER*2 U,S,N,A(N)
```

Return the characteristics of display unit U in array A. N words of the internal device configuration data starting at position S within the configuration data array will be returned in array A. Appendix A contains a description of the data returned in array A.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated
6	Illegal data request

3.3.6 XDDNAME

```
LOGICAL FUNCTION XDDNAME (U,F,N,L,R)
INTEGER*2 U,F,L,R
BYTE      N(1)
```

Return the physical or generic name of the currently allocated display device U. The flag F determines which name is returned in the byte array N. L is the length of the byte array N, and R is the number of characters returned. The following define the possible values of F:

F = 1	Return physical device name
F = 2	Return generic device name

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal selection code

3.3.7 XDDOPEN

```
LOGICAL FUNCTION XDDOPEN (U)
INTEGER*2 U
```

Execute this routine before any I/O to display device U. On the VAX system, an I/O channel to the display device is opened.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated

3.4 GRAPHICS OVERLAY

Using graphics overlay is logically the same as using any other image memory plane for video output. The pixel values output by the graphics overlay hardware replace pixels output from the standard outputs when graphics overlay is on. The graphics overlay LUT logically consists of three standard size LUT tables, one for red, green and blue (color only). Many devices, however, do not support full-size overlay LUTs. In this case the software does the best it can.

3.4.1 XDGCONNECT

```
LOGICAL FUNCTION XDGCONNECT (U,I,S,BYPASS)
INTEGER*2 U,I,S
LOGICAL*2 BYPASS
```

Connect image memory plane I of display unit U to the graphics overlay plane LUT L section S. This has the effect of making the image memory plane I the graphics overlay plane. If BYPASS is TRUE, pixels will not be converted by the LUT before output. If BYPASS is FALSE, pixels will be converted by the LUT before output.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device is not allocated or active
6	Image memory plane does not exist
8	LUT section does not exist
10	BYPASS not allowed
12	Graphics overlay not available

3.4.2 XDGLCONSTANT

```
LOGICAL FUNCTION XDGLCONSTANT (U,S,R,G,B)
INTEGER*2 U,S,R,G,B
```

Set the graphics overlay LUT section S to constant R (red), G (green) and B (blue) values.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Section does not exist
8	Graphics overlay not available

3.4.3 XDGLREAD

```
LOGICAL FUNCTION XDGLREAD (U,S,R,G,B)
INTEGER*2 U,S,R(256),G(256),B(256)
```

Copy graphics overlay LUT section S values into the arrays R (red), G (green) and B (blue).

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Section does not exist
8	Graphics overlay not available

3.4.4 XDGLWRITE

```
LOGICAL FUNCTION XDGLWRITE (U,S,R,G,B)
INTEGER*2 U,S,R(256),G(256),B(256)
```

Copy the arrays R (red), G (green) and B (blue) into the graphics overlay LUT section S in display unit U.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Section does not exist
8	Graphics overlay not available

3.4.5 XDGOFF

```
LOGICAL FUNCTION XDGOFF (U)
INTEGER*2 U
```

Turn off the display of the graphics overlay plane in display unit U.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Graphics overlay not available

3.4.6 XDGON

```
LOGICAL FUNCTION XDGON (U)
INTEGER*2 U
```

Turn on the display of the graphics overlay plane in display unit U.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Graphics overlay not available

3.5 IMAGE MEMORY PLANE

3.5.1 XDIAREAFILL

```
LOGICAL FUNCTION XDIAREAFILL (U,I,BOUNDARY,FILL)
INTEGER*2 U,I
BYTE      BOUNDARY,FILL
```

Fill a bounded closed area in image memory plane I of unit U with the value FILL. The bounded area is determined by the value BOUNDARY. Only pixels within the access window will be changed.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist

NOTE: Currently, this subroutine is not implemented.

3.5.2 XDIAWLOCATION

```
LOGICAL FUNCTION XDIAWLOCATION (U,I,LEFT,TOP,RIGHT,BOTTOM)
INTEGER*2 U,I,LEFT,TOP,RIGHT,BOTTOM
```

Return the location of the access window of image memory plane I of display unit U. The coordinates are image memory plane coordinates.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device is not allocated or active
6	Image memory plane does not exist

3.5.3 XDIAWREAD

```
LOGICAL FUNCTION XDIAWREAD (U,I,N,A)
INTEGER*2 U,I
INTEGER*4 N
BYTE      A(N)
```

Read N pixels from the access window of image memory plane I of display unit U into the array A. The read starts in the upper left corner of the access window and continues one line at a time from top to bottom.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist
8	Pixel count (N) less than or equal to zero

3.5.4 XDIAWSET

```
LOGICAL FUNCTION XDIAWSET (U,I,LEFT,TOP,RIGHT,BOTTOM)
INTEGER*2 U,I,LEFT,TOP,RIGHT,BOTTOM
```

Set the location of the access window of image memory plane I of display unit U. The access window must be the same as or within the image memory plane. LEFT must be less than or equal to RIGHT, and TOP must be less than or equal to BOTTOM.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device is not allocated or active
6	Image memory plane does not exist
8	Part of AW is not on image memory plane
10	Coordinates are out of order

3.5.5 XDIAWWRITE

```
LOGICAL FUNCTION XDIAWWRITE (U,I,N,A)
INTEGER*2 U,I
INTEGER*4 N
BYTE      A(N)
```

Write N pixels into the access window of image memory plane I of unit U for array A. The write starts in the upper left corner of the access window and continues one line at a time from top to bottom.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist
8	Pixel count (N) less than or equal to zero

3.5.6 XDICIRCLE

```
LOGICAL FUNCTION XDICIRCLE (U,I,X,Y,R,V)
INTEGER*2 U,I,X,Y,R
BYTE      V
```

Draw a circle of pixel value V in image memory plane I of display unit U of radius R one pixel wide. Any part of the circle outside the access window will not be drawn. Anti-aliasing is implementation dependent.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist

3.5.7 XDIDWLOCATION

```
LOGICAL FUNCTION XDIDWLOCATION (U,I,LEFT,TOP)
INTEGER*2 U,I,LEFT,TOP
```

Return the upper left-hand corner coordinates (LEFT,TOP) of the display window of image memory plane I to display unit U.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist

3.5.8 XDIDWSET

```
LOGICAL FUNCTION XDIDWSET (U,I,LEFT,TOP)
INTEGER*2 U,I,LEFT,TOP
```

Set the upper left corner of the display window of image memory plane I of display unit U to coordinates (LEFT,TOP).

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exits
8	DW location is illegal

3.5.9 XDIFILL

```
LOGICAL FUNCTION XDIFILL (U,I,V)
INTEGER*2 U,I
BYTE      V
```

Fill all the pixels in the access window of image memory plane I in display unit U with the value V.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist

3.5.10 XDIHISTOGRAM

```
LOGICAL FUNCTION XDIHISTOGRAM (U,I,M,A)
INTEGER*2 U,I,M
INTEGER*4 A(256)
```

Return a histogram of the pixels in image memory plane I of display unit U in array A. Parameter M is a mask image memory plane number. If M is zero, all pixels within the access window of I will be used to calculate the histogram. If M is greater than zero, only pixels with a nonzero counterpart in the M image memory plane will be used to calculate the histogram. The pixel must also be within both access windows to be used in the calculations.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane I does not exist
8	Image memory plane M does not exist

3.5.11 XDIIARITHMETIC

```
LOGICAL FUNCTION XDIIARITHMETIC  (U,OP,I1,I2,I3)
INTEGER*2 U,OP,I1,I2,I3
```

Perform the arithmetic operation (OP) between image memory plane I1 and I2 of display unit U. The results will be stored in I3. The operation will be performed on pixels that are within the access windows of I1, I2 and I3. If the display unit I1, I2 or I3 does not exist, FALSE will be returned.

OP Codes	Description
0	Add
1	Subtract

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist
8	Illegal operation

Note: The operation performed is I3 = I1 OP I2.

3.5.12 XDIICOPY

```
LOGICAL FUNCTION XDIICOPY (U,I1,I2)
INTEGER*2 U,I1,I2
```

Copy the contents of the access window of image memory plane I1 into the access window of image memory plane I2. The access windows must be the same size but not necessarily in the same location.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory 1 plane does not exist
8	Image memory 2 plane does not exist
10	Access windows are not the same size

3.5.13 XDIILOGICAL

```
LOGICAL FUNCTION XDIILCGICAL (U,OP,I1,I2,I3)
INTEGER*2 U,OP,I1,I2,I3
```

Perform a bit-wise logical operation between image memory plane I1 and I2 of display unit U. The results will be stored in I3. OP indicates the operation to be performed on pixels that are within the access windows of I1, I2 and I3.

OP Codes	Description
0	AND
1	OR
2	Exclusive OR
3	NOT AND
4	NOT OR
5	NOT Exclusive OR

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane I1 does not exist
8	Image memory plane I2 does not exist
10	Image memory plane I3 does not exist
12	Illegal operation

3.5.14 XDDISHIFT

```
LOGICAL FUNCTION XDIISHIFT (U,K,I1,I2,WRAP)
INTEGER*2 U,K,I1,I2
LOGICAL*2 WRAP
```

Perform a K-bit logical shift operation on pixels in image memory plane I1 of display unit U. The results will be stored in I2. If K is positive, the bits will be shifted left. If K is negative, the bits will be shifted right. The operation will only be performed on pixels that are within the access windows. WRAP is a logical flag indicating bits will be wrapped around or zero filled. TRUE (1) is wrapped and FALSE (0) is zero filled.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane I1 does not exist
8	Image memory plane I2 does not exist

3.5.15 XDILINEREAD

```
LOGICAL FUNCTION XDILINEREAD(U,I,X,Y,N,A)
INTEGER*2 U,I,X,Y,N
BYTE      A(N)
```

Read a line of pixels from image plane I in display unit U into array A. The read starts at location (X,Y) in the image plane. N pixel values will be read. Only pixels within the access window will be read.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist

3.5.16 XDILINEWRITE

```
LOGICAL FUNCTION XDILINEWRITE (U,I,X,Y,N,A)
INTEGER*2 U,I,X,Y,N
BYTE      A(N)
```

Write a line of pixels into image plane I of display unit U from the array A. N pixels will be written starting at location (X,Y) in the image plane. If a pixel is outside the access window, the values will remain unchanged.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist

3.5.17 XDIMAWWRITE

```
LOGICAL FUNCTION XDIMAWWRITE (U,I,M,N,A)
INTEGER*2 U,I,M,A(*)
```

This is the same subroutine as XDIAWWRITE except that the write mask M determines which bits in the image memory plane may be modified. A one indicates that the bit may be modified, and a zero indicates that the bit may not be changed. For example, if bit 3 of the mask is on (1), bit 3 of the pixel being written will replace bit 3 in the image memory plane. For return codes and other information, see XDIAWWRITE.

3.5.18 XDIMFILL

```
LOGICAL FUNCTION XDIMFILL (U,I,M,V)
INTEGER*2 U,I,M,V
```

This is the same subroutine as XDIFILL except that the write mask M determines which bits in the image memory plane may be modified. A one indicates that the bit may be modified, and a zero indicates that the bit may not be changed. For example, if bit 3 of the mask is on (1), bit 3 of the pixel being written will replace bit 3 in the image memory plane. For return codes and other information, see XDIFILL.

3.5.19 XDIMLINEWRITE

```
LOGICAL FUNCTION XDIMLINEWRITE (U,I,X,Y,M,N,A)
INTEGER*2 U,I,X,Y,M,N,A(*)
```

This is the same subroutine as XDILINEWRITE except that the write mask M determines which bits in the image memory plane may be modified. A one indicates that the bit may be modified, and a zero indicates that the bit may not be changed. For example, if bit 3 of the mask is on (1), bit 3 of the pixel being written will replace bit 3 in the image memory plane. For return codes and other information, see XDILINEWRITE.

3.5.20 XDIMPIXELWRITE

```
LOGICAL FUNCTION XDIMPIXELWRITE (U,I,X,Y,M,V)
INTEGER*2 U,I,X,Y,M,V
```

This is the same subroutine as XDIPIXELWRITE except that the write mask M determines which bits in the image memory plane may be modified. A one indicates that the bit may be modified, and a zero indicates that the bit may not be changed. For example, if bit 3 of the mask is on (1), bit 3 of the pixel being written will replace bit 3 in the image memory plane. For return codes and other information, see XDIPIXELWRITE.

3.5.21 XDIPIXELREAD

```
LOGICAL FICTION XDIPIXELREAD (U,I,X,Y,V)
INTEGER*2 U,I,X,Y
BYTE      V
```

Read the pixel value at image memory coordinates (X,Y) in image memory plane I of display unit U. If the pixel is within the access window, the pixel value will be returned in V; otherwise V will remain unchanged.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist
8	Coordinates are outside access window

3.5.22 XDIPIXELWRITE

```
LOGICAL FUNCTION XDIPIXLEWRITE (U,I,X,Y,V)
INTEGER*2 U,I,X,Y
BYTE      V
```

Set the pixel at coordinates (X,Y) in image plane I of display unit U to the value V. If the coordinates are outside the access window, the pixel will remain unchanged.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist
8	Coordinates are outside access window

3.5.23 XDIPOLYLINE

```
LOGICAL FUNCTION XDIPOLYLINE (U,I,V,N,X,Y)
INTEGER*2 U,I,N,X(N),Y(N)
BYTE      V
```

Draw a series of vectors of value V in image memory plane I of unit U. N is the number of vector coordinate pairs in the arrays X and Y. N must be greater than 1. The first coordinates are the starting position for drawing the vectors. A vector is drawn from this point to the second pair of coordinates; from the second to third; etc. Any part of a vector falling outside the access window will not be drawn. Anti-aliasing may or may not take place.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist
8	Number of coordinate pairs (X,Y) is less than 2

3.5.24 XDIROTATE

```
LOGICAL FUNCTION XDIROTATE (U,I,ANGLE)
INTEGER*2 U,I,ANGLE
```

Rotate the the pixels within the access window of image memory plane I of unit U. ANGLE is a flag indicating a rotation angle of 90 or 180 degrees. The access window must be square for this operation.

Rotation Flag	Description
1	-180 degrees
2	-90 degrees
3	90 degrees
4	180 degrees

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist
8	Illegal rotation angle
10	Access window is not square

NOTE: Currently this subroutine is not implemented.

3.6 LOOK UP TABLE

3.6.1 XDLCONNECT

```
LOGICAL FUNCTION XDLCONNECT (U,I,L,S,BYPASS)
INTEGER*2 U,I,L,S
LOGICAL   BYPASS
```

Connect the image memory plane I to the output LUT L section S of display unit U. If BYPASS is true, pixels will not be converted by the LUT before output. If BYPASS is false, pixels will be converted by the LUT before output.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist
8	LUT does not exist
10	LUT section does not exist
12	Bypass of LUT is not allowed

3.6.2 XDLRAMP

```
LOGICAL FUNCTION XDLRAMP (U,L,S)
INTEGER*2 U,L,S
```

Set the LUT L section S in display unit U to a linear ramp. Eight-bit LUTs will be set to values 0 to 255. LUTs with larger outputs will be set to a ramp that has the largest possible value in the last position.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	LUT does not exist
8	LUT section does not exist

3.6.3 XDLREAD

```
LOGICAL FUNCTION XDLREAD (U,L,S,A)
INTEGER*2 U,L,S,A(256)
```

Copy LUT L section S in display unit U into array A.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	LUT does not exist
8	LUT section does not exist

3.6.4 XDLWRITE

```
LOGICAL FUNCTION XDLWRITE (U,L,S,A)
INTEGER*2 U,L,S,A(256)
```

Copy array A into LUT L section S in display unit U.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	LUT does not exist
8	LUT section does not exist

3.6.5 XDLZOOM

```
LOGICAL FUNCTION XDLZOOM (U,L,Z)
INTEGER*2 U,L,Z
```

Set the zoom factor of LUT L in display unit U to Z.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	LUT does not exist
8	Illegal zoom factor

3.7 TEXT GENERATION

3.7.1 XDTCOLOR

```
LOGICAL FUNCTION XDTCOLOR (C,P)
INTEGER*2 C,P
```

Set the color value C and the precision P used to write text strings. The precision is an implementation defined value indicating how text is to be written, i.e. anti-aliasing, etc. The default value zero (0) for the precision should generally be used. The color value is an integer between 0 and 255.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Illegal precision

NOTE: Currently precision is not implemented. A default value zero (0) should be used.

3.7.2 XDTFONT

```
LOGICAL FUNCTION XDTFONT (F)
INTEGER*2 F
```

Read a text font description file into the internal font table. F indicates which implementation defined font type is to be read. F is one (1) or greater. Zero (0) is the default font type.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Illegal font type
4	Internal font table too small
6	Error opening font file
8	Error reading character count
10	Error reading ASCII character code, character vector count or character width
12	Error reading MOVE/DRAW flag, X coordinate or Y coordinate
14	Premature end-of-file of font file
16	Illegal ASCII character code
18	Illegal vector count

NOTE: If error code 4 through 18 occurs, please notify the system engineer.

3.7.3 XDTLENGTH

```
LOGICAL FUNCTION XDTLENGTH (L,N,A)
INTEGER*2 L,N
BYTE      A(N)
```

Return the length L (in pixels) of the string of N characters found in array A.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Illegal character count

3.7.4 XDTROTATE

```
LOGICAL FUNCTION XDTROTATE (A)
REAL A
```

Set the angle above or below the X (horizontal) axis where text is to be written. R is an angle and must be between +180 and -180 degrees.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Illegal angle

3.7.5 XDTSIZE

```
LOGICAL FUNCTION XDTSIZE (H,S)
INTEGER*2 H
REAL      S
```

Set the height H of the text in pixel coordinates (how many pixels tall is the text to be). Also set the horizontal scale factor to be applied to the text. The fonts are defined in a machine independent coordinate system that must be scaled to fit the display.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Illegal height
4	Illegal horizontal scale factor

3.7.6 XDITEXT

```
LOGICAL FUNCTION XDITEXT (U,I,X,Y,LOC,N,A)
INTEGER*2 U,I,X,Y,LOC,N
BYTE      A
```

Write a text string into image memory plane I of display unit U. LOC is a flag indicating where the text string is to be written relative to the (X,Y) in the image memory plane. N is the number of characters in array A to be written. The coordinates (X,Y) must be inside the access window. Any part of the text string that falls outside the access window will not be written into the image memory plane.

LOC Value	Description
1	(X,Y) is lower left corner of text string
2	(X,Y) is bottom center of text string
3	(X,Y) is lower right corner of text string

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Image memory plane does not exist
8	Illegal LOC value
10	Illegal character count (N)

3.8 INTERACTIVE I/O DEVICE

In order to generalize the interface to interactive I/O devices, the interface accesses generic devices such as knobs, tablets and switches. All interactive I/O devices are made to fall under one or more of these generic descriptions. For example, a joystick may appear to the system as a tablet with the pen always in proximity to the tablet and the pen down.

Devices that return coordinates fall into two categories. One returns absolute coordinates, such as a tablet. The second returns relative coordinates, such as a joystick (relative indicates a delta value). To determine what type of device is available and if it returns absolute or relative coordinates, use "XDDINFO". See Appendix A for more information.

Some 2-D and 3-D devices do not have a pen, for example, joysticks, pucks, etc. In many cases these devices have switches that could be substituted for a pen. If a switch is used to simulate a pen, the number of the switch used can be obtained by XDDINFO (see Appendix A.) If a switch is used to simulate the pen, the switch value may also be obtained from XDXSWITCH. The switch number used to simulate the pen is made availabe to the application program (through XDDINFO) so that the user may be informed about which switch is to be used.

Many interactive I/O devices return coordinates as well as switch values. In this case the device may be accessed by more that one interface routine, for example, XDX2D and XDXSWITCH.

The following is a description of the information that may be obtained about each device with XDDINFO (see Appendix A for more information):

ARRAY POSITION	DESCRIPTION
1	Device Type 0 - No device available 1 - 1D device (knob) 2 - 2D device (tablet and pen) 3 - 3D device (tablet and pen) 4 - Switch device
2	Type of Coordinates Returned 0 - Device does not return coordinates 1 - Returns absolute coordinates 2 - Returns relative coordinates

3 Switch Simulating Pen

O - No switch simulating a pen
N - Number of switch used to simulate pen

4 Switches Available

Number of switches available on device

3.8.1 XDX1D

```
LOGICAL FUNCTION XDX1D (U,D,K,X)
REAL        X
INTEGER*2 U,D,K
```

Return a value from interactive I/O device D connected to display unit U. Device D returns a single value X. The model for this type of device is a set of knobs. The returned value X has been normalized to the range -1.0 to +1.0. K indicates from which knob values are returned. K is 1 or greater.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal interactive I/O device
8	Illegal K value

3.8.2 XDX2D

```
LOGICAL FUNCTION XDX2D (U,D,X,Y,PROX,PEN)
REAL        X,Y
INTEGER*2 U,D,PROX,PEN
```

Return values from interactive I/O device D connected to display unit U. Device D returns two-dimensional coordinates (X,Y). The model for this type of device is a tablet and pen. The returned values X and Y have been normalized to the range -1.0 to +1.0. PROX is a flag that indicates proximity (0-out of proximity,1-in proximity). If the device has no proximity detection, the returned value is always 1. PEN is a flag indicating if the pen is down or up (0-up,1-down). If the device has no pen or switch to simulate a pen, the value returned is always 1.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal interactive I/O device

3.8.3 XDX3D

```
LOGICAL FUNCTION XDX3D (U,D,X,Y,Z,PROX,PEN)
REAL       X,Y,Z
INTEGER*2 U,D,PROX,PEN
```

Return values from interactive I/O device D connected to display unit U. Device D returns three-dimensional coordinates (X,Y,Z). The model for this type of device is a tablet and pen. The returned values X, Y and Z have been normalized to the range -1.0 to +1.0. PROX is a flag that indicates proximity (0-out of proximity,1-in proximity). If the device has no proximity detection, the returned value is always 1. PEN is a flag indicating if the pen is down or up (0-up,1-down). If the device has no pen or switch to simulate a pen, the value returned is always 1.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal interactive I/O device

3.8.4 XDXSWITCH

```
LOGICAL FUNCTION XDXSWITCH (U,D,S,V)
INTEGER*2 U,D,S,V
```

Return values from interactive I/O device D connected to display unit U. Device D returns a single value V. The model for this type of device is a switch box. The returned value V indicates if the switch is off (0) or on (1). S indicates which switch to test. S is one or greater.

Returned Status	Description
0	Function not implemented
1	Normal return (logical TRUE)
2	Unit number out of range
4	Display device not allocated or active
6	Illegal interactive I/O device
8	Illegal S value

APPENDIX A

DATA RETURNED BY XDDINFO

The XDDINFO subroutine returns information about the frame buffer currently being used. The data returned are any or all of an integer*2 array. The following defines the data in that array that may be returned by XDDINFO:

Array Index	Description
(1)	physical device number
(2)	display active (0-no,1-yes)
(3)	number of video outputs (LUT/DACs)
(4)	number of image memory planes
(5)	number of lines in image memory plane (IMP)
(6)	number of samples per line
(7)	hardware configurations available
	bit 0 - 512 x 512 image memory planes available
	bit 1 - 1024 x 1024 image memory planes available
	bit 2 - 2048 x 2048 image memory planes available
	bit 3 - 4096 x 4096 image memory planes available
	bit 4 - reserved
	bit 5 - reserved
	bit 6 - reserved
	bit 7 - reserved
	bit 8 - 512 x 512 video output (low res.)
	bit 9 - 1024 x 1024 video output (high res.)
	bit 10 - reserved
	bit 11 - reserved
	bit 12 - 1 x 1 aspect ratio available
	bit 13 - 3 x 4 aspect ratio available
	bit 14 - reserved
	bit 15 - reserved
(8)	reserved
(9)	reserved

(10) current D(1) configuration value from XDDCONFIGURE
  1 - full color
  2 - pseudo color
  3 - monochrome

(11) current D(2) configuration value from XDDCONFIGURE
  1 - 512 x 512 IMPs
  2 - 1024 x 1024 IMPs
  3 - 2048 x 2048 IMPs
  4 - 4096 x 4096 IMPs

(12) current D(3) configuration value from XDDCONFIGURE
  1 - 512 x 512 video output (low res.)
  2 - 1024 x 1024 video output (high res.)

(13) current D(4) configuration value from XDDCONFIGURE
  1 - 1 x 1 aspect ratio
  2 - 4 x 3 aspect ratio

(14) reserved

(15) reserved

(16) reserved

(17) reserved

(18) reserved

(19) reserved

(20) each IMP has a separate display window (0-no,1-yes)

(21) IMPs may be connected to different LUTs
  0 - no (IMP/LUT connection is hardwired, can not be changed)
  1 - Yes (IMPs can be connected to R,G,B LUTs)

(22) reserved

(23) reserved

(24) number of sections in image LUTs

(25) LUT may be bypassed (0-no,1-yes)

(26) largest value in LUT (8-bit DACs output 255)

(27) each LUT has a separate zoom (0-no,1-yes)

(28) reserved

(29) reserved

(30) display device has graphics overlay (0-no,1-yes)

(31) display graphics overlay on/off (0-off,1-on)

(32) IMPs can be connected to overlay LUT (0-no,1-yes)

(33) graphics overlay LUT may be bypassed (0-no,1-yes)

(34) IMP currently connected to graphics overlay LUT

(35) graphics LUT section used (0-bypass,>0-LUT section)

(36) graphics overlay LUT characteristics code
  1 - has a separate graphics overlay LUT
  2 - graphics overlay LUT is the same as image LUTs

(37) number of sections in graphics LUT

(38) largest pixel value in graphics overlay plane (graphics overlay plane may have less that 8 bits)

(39) reserved

(40) reserved

(41) display device has an AEG (0-no,1-yes)
(42) AEG on/off (0-off,1-on)
(43) number of lines of text in AEG
(44) number of characters per line in AEG
(45) number of AEG display types
(46) reserved
(47) reserved
(48) number of cursors
(49) number of cursor types
(50) number of cursor blink rates
(51) reserved
(52) reserved
(53) reserved
(54) reserved
(55) reserved
(56) reserved
(57) reserved
(58) reserved
(59) reserved
(59) reserved
(60) number of interactive I/O devices
(61) I/O device 1 - device type
(62) I/O device 1 - coordinates returned
(63) I/O device 1 - pen simulation
(64) I/O device 1 - number of switches
(65) reserved
(66) reserved
(67) reserved
(68) reserved
(69) reserved
(70) reserved
(71) I/O device 2 - device type
(72) I/O device 2 - coordinates returned
(73) I/O device 2 - pen simulation
(74) I/O device 2 - number of switches
(75) reserved
(76) reserved
(77) reserved
(78) reserved
(79) reserved
(80) reserved

APPENDIX B

TEXT FONTS

The following is a list and description of the fonts currently available on the MIPL system. The Standard font files contain 7-bit ASCII codes. The special and combination fonts contain multiple fonts and special characters and do not conform to any ASCII standard. A complete description of the special fonts currently available is provided in this appendix. Several utilities are also available that will display a font on a frame buffer or plot it on a Printronix printer.

Table B-1. Standard Fonts Available

Font Number	Font File Name	Description
0	000.FON	Default Font
1	001.FON	Simplex
2	002.FON	Duplex
3	003.FON	Roman
4	004.FON	Standard
5	005.FON	Standard 2
6	006.FON	Standard Italics
7	007.FON	Script
8	008.FON	Hollow
9	009.FON	Cartographic
10	010.FON	Greek
11	011.FON	English Gothic
12	012.FON	German Gothic
13	013.FON	Italian Gothic
14	014.FON	Cyrillic
30	030.FON	Font from plot package

Table B-2. Special and Combination Fonts Available

Font Number	Font file Name	Description
102	102.FON	Cartographic Special Characters
103	103.FON	Upper Case Simplex Roman, and Greek
104	104.FON	Lower Case Simplex Roman, and Greek
105	105.FON	Simplex Roman Special Characters, Geometry, Cards and Weather Symbols
106	106.FON	Circuit and Map Symbols
107	107.FON	Circles and Highway Symbols
111	111.FON	Math Symbols (normal size)
112	112.FON	Upper Case Complex Roman, Greek and Italic
113	113.FON	Lower Case Complex Roman, Greek and Italic
114	114.FON	Complex Roman Special Characters and Astrology Symbols
115	115.FON	Zodiac and Music Symbols
116	116.FON	Math Symbols (large size)
117	117.FON	Upper Case Duplex Roman and Complex Script
118	118.FON	Lower Case Duplex Roman and Complex Script
119	119.FON	Duplex Roman and Complex Script Special Characters
120	120.FON	Complex Cyrillic (Upper Case)
121	121.FON	Complex Cyrillic (Lower Case)
122	122.FON	Upper Case Triplex Roman and Italic
123	123.FON	Lower Case Triplex Roman and Italic
124	124.FON	Triplex Roman and Triplex Italic Special Characters
125	125.FON	German Gothic (Upper Case)
126	126.FON	German Gothic (Lower Case)
127	127.FON	English Gothic (Upper Case)
128	128.FON	English Gothic (Lower Case)
129	129.FON	Gothic Special Characters
130	130.FON	Italian Gothic (Upper Case)
131	131.FON	Italian Gothic (Lower Case)

Table B-3. FONT 102, Cartographic Special Characters

Character Value	Description
0 - 9	Numerals
10	Period
11	Comma
12	Colon
13	Exclamation
14	Prime
15	Interrogation
16	Prime
17	Second,Quote
18	Degree
19	Dollar
20	Solidus
21	Left Parenthesis
22	Right Parenthesis
23	Bar
24	Minus
25	Sum
26	Equality
27	Cross
28	Asterisk
29	Dot
30	Left Quotation
31	Right Quotation
32	Arrow
33	Number
34	Ampersand
35	Lozenge

Table B-4. FONT 105, Simplex Roman Special Characters, Geometry, Card and Weather Symbols

Character Value	Description
0 - 9	Numerals
10	Period
11	Comma
12	Colon
13	Exclamation
14	Prime
15	Interrogation
16	Prime
17	Second, Quote
18	Degree
19	Dollar
20	Solidus
21	Left Parenthesis
22	Right Parenthesis
23	Bar
24	Minus
25	Sum
26	Equality
27	Cross
28	Asterisk
29	Dot
30	Left Quotation
31	Right Quotation
32	Arrow
33	Number
34	Ampersand
35	Lozenge
Geometry Symbols	
37	Parallel
38	Perpendicular
39	Angle
40	Conclusion
Card Symbols	
41	Spade
42	Heart
43	Diamond
44	Club
Misc. Symbols	
45	Shamrock
46	Fleur De Lis
Weather Symbols	
50	Drizzle
51	Rain
52	Snow
53	Surface Cold Front
54	Surface Warm Front
55	50 Knot Flag
56	Upper Cold Front
57	Upper Warm Front
58	Cumulo
59	Alto
60	Alto
61	Cirro
62	Left Cirrostrato
63	Right Cirrostrato
64	Sand
65	Glaze
66	Haze
67	Thunderstorm
68	Hurricane
Circuit Symbols	
96	Horizontal
97	45-Oblique
99	Space (blank)

Table B-5. FONT 106, Circuit and Map Symbols

Character Value	Description
Circuit Symbols	
00	Horizontal
01	30-Oblique
02	60-Oblique
03	Vertical
04	120-Oblique
05	150-Oblique
06	Horizontal
07	45-Oblique
08	Vertical
09	135-Oblique
10	Upper Left Quadrant
11	Lower Left Quadrant
12	Lower Right Quadrant
13	Upper Right Quadrant
14	Lower Quadrant
15	Left Quadrant
16	Right Quadrant
17	Upper Quadrant
18	Vertical Zigzag
19	Horizontal ZigZag
20	30-Zigzag
21	45-Zigzag
22	Upper Loop
23	Left Loop
24	Lower Loop
25	Right Loop
26	30-Loop
27	45-Loop
28	Junction
29	Jumper
30	Grid
31	Shield
32	Filament
33	Ground
34	Antenna
Map Symbols	
40	Circle
41	Square
42	Triangle
43	Diamond
44	Star
45	Mark
46	Cross (line)
47	Asterisk
50	Circle (solid)
51	Square (solid)
52	Up Vertex (solid)
53	Left Vertex (solid)
54	Down Vertex (solid)
55	Right Vertex (solid)
56	Star (solid)
57	Flag (solid)
60	Anchorage
61	Aerodrome
62	Mine
63	Derrick
64	Lightship
65	Wreck
66	Cross (hollow)
67	Crescent
68	Star (hollow)
69	Bell
70	Palm
71	Pine
72	Oak
73	Willow
74	Grass
99	Space (blank)

Table B-6. FONT 107, Circles and Highway Signs

Character Value	Description
00	2-Circle
01	4-Circle
02	5-Circle
03	7-circle
04	11-Circle
05	17-Circle
06	22-circle
07	41-Circle
08	US Highway Sign
09	IS Highway Sign

Table B-7. FONT 111, Math Symbols (Normal Size)

Character Value	Description
01	Pi
02	Sigma
03	Left Parenthesis
04	Right Parenthesis
05	Left Bracket
06	Right Bracket
07	Left Brace
08	Right Brace
09	Upper Half Brace
10	Lower Half Brace
11	Radical
12	Integral

Table B-8. FONT 114, Complex Roman Special Characters and Astrology Symbols

Character Value	Description
Roman Special Characters	
0 - 9	Numerals
10	Period
11	Comma
12	Colon
13	Semicolon
14	Exclamation
15	Interrogation
16	Prime
17	Second
18	Degree
19	Asterisk
20	Solidus
21	Left Parenthesis
22	Right Parenthesis
23	Left Bracket
24	Right Bracket
25	left Brace
26	Right Brace
27	Left Elbow
28	Right Elbow
29	Bar
30	Double Bar
31	Minus
32	Plus
33	Plus or Minus
34	Minus or Plus
35	Cross Product
36	Dot Product
37	Quotient
38	Equality
39	Inequality
40	Identity
41	Less
42	More
43	Equal or Less
44	Equal or More
45	Variation
46	Approximation
47	Caret
48	Acute Angle
49	Grave Accent
50	Breve
51	Right Quotation
52	Left Quotation
53	Normal Aspirate
54	Inverted Aspirate
55	Radical
56	Right Hook
57	Up Hook
58	Left Hook
59	Down Hook
60	Element
61	Right Arrow
62	Up Arrow
63	Left Arrow
64	Down Arrow
65	Delta
66	Nabla
67	Radical
68	Integral
69	Circuit Integral
70	Infinity
71	Percent
72	Ampersand
73	At
74	Dollar
75	Number
76	Paragraph
77	Dagger
78	Double Dagger
79	Existence
Astrology Symbols	
81	Sun
82	Mercury
83	Venus
84	Earth
85	Mars
86	Jupiter
87	Saturn
88	Uranus
89	Neptune
90	Pluto
91	Moon
92	Comet
93	Star
94	Ascending Node
95	Descending Node

Table B-9. FONT 115, Zodiac and Music Symbols

Character Value	Description
Zodiac Symbols	
01	Aries
02	Taurus
03	Gemini
04	Cancer
05	Leo
06	Virgo
07	Libra
08	Scorpio
09	Sagittarius
10	Capricorn
11	Aquarius
12	Pisces
Music Symbols (light)	
17	Dot
18	Upper Flag
19	Lower Flag
20	Whole Note
21	Half Note
22	Quarter Note
23	Sharp
24	Natural
25	Flat
26	Whole Rest
27	Half Rest
28	Quarter Rest
29	Eighth Rest
30	G Clef
31	F Clef
32	C clef

Character Value	Description
Music Sysbols (heavy)	
67	Dot
68	Upper Flag
69	Lower Flag
70	Whole Note
71	Half Note
72	Quarter Note
73	Sharp
74	Natural
75	Flat
76	Whole Rest
77	Half Rest
78	Quarter Rest
79	Eighth Rest
80	G Clef
81	F Clef
82	C Clef

Table B-10. FONT 116, Math Symbols (Large Size)

Character Value	Description
01	Pi
02	Sigma
03	Left Parenthesis
04	Right Parenthesis
05	Left Bracket
06	Right Bracket
07	Left Brace
08	Right Brace
09	Upper Half Brace
10	Lower Half Brace
11	Radical
12	Integral

Table B-11. FONT 119, Duplex Roman and Complex Script Special characters

Character Value	Description	Character Value	Description
Duplex Roman		Complex Script	
0 - 9	Numerals	50 - 59	Numerals
10	Period	60	Period
11	Comma	61	Comma
12	Colon	62	Colon
13	Semicolon	63	Semicolon
14	Exclamation	64	Exclamation
15	Interrogation	65	Interrogation
16	Left Quote	66	Left Quote
17	Right quote	67	Right Quote
18	Ampersand	68	Ampersand
19	Dollar	69	Dollar
20	Solidus	70	Solidus
21	Left Parenthesis	71	left Parenthesis
22	Right Parenthesis	72	Right Parenthesis
23	Asterisk	73	Asterisk
24	Subtraction	74	Subtraction
25	Addition	75	Addition
26	Equality	76	Equality
27	Prime	77	Prime
28	Second	78	Second
29	Degree	79	Degree

Table B-12. FONT 124, Triplex Roman and Triplex Italic Special Characters

Character Value	Description	Character Value	Description
Duplex Roman		Complex Script	
0 - 9	Numerals	50 - 59	Numerals
10	Period	60	Period
11	Comma	61	Comma
12	Colon	62	Colon
13	Semicolon	63	Semicolon
14	Exclamation	64	Exclamation
15	Interrogation	65	Interrogation
16	Left Quote	66	Left Quote
17	Right quote	67	Right Quote
18	Ampersand	68	Ampersand
19	Dollar	69	Dollar
20	Solidus	70	Solidus
21	Left Parenthesis	71	left Parenthesis
22	Right Parenthesis	72	Right Parenthesis
23	Asterisk	73	Asterisk
24	Subtraction	74	Subtraction
25	Addition	75	Addition
26	Equality	76	Equality
27	Prime	77	Prime
28	Second	78	Second
29	Degree	79	Degree

Table B-13. FONT 129, Gothic Special Characters

Character Value	Description
0 - 9	Numerals
10	Period
11	Comma
12	Colon
13	Semicolon
14	Exclamation
15	Interrogation
16	Left Quote
17	Right Quote
18	Ampersand
19	Dollar
20	Solidus
21	Left Parenthesis
22	Right Parenthesis
23	Asterisk
24	Subtraction
25	Addition
26	Equality
27	Prime
28	Second
29	Degree

APPENDIX C

CREATING USER DEFINED FONTS

C.1 HISTORY

The Hershey character fonts are the basis for the fonts provided by the XD text subroutines. The Hershey fonts were originally obtained in a complicated bit saving format. This format was changed and expanded to an ASCII text file for ease of use and modification. The Hershey character fonts are widely used and provide a large number of standard and special character fonts.

C.2 USER DEFINED FONTS

The XD text subroutines are written so that users may define their own text fonts. Font files are named XXX.FON where XXX is a three digit number between 000 and 999. Font files are ASCII text files and may be created using a text editor or a program. "XDFONT:" is a symbol pointing to the directory containing the converted Hershey font files. Users may redirect this symbol to any directory they wish. The following is an example redirecting XDFONT to a different directory:

```
$ assign  mydisk:[home.myfonts]  xdfont
```

C.3 DEFINING CHARACTERS

Characters within a font are defined by a series of vectors that are drawn on an image memory plane. The time needed to draw a character is directly related to the number of vectors making up the character. Very simple to very complex fonts have been provided. Special characters and symbols have also been provided (see Appendix B for more information).

Vectors making up a character have an origin in the lower left hand corner of a box that defines the nominal size of a character (see Fig. C-1). Vectors making up a character are not required to stay within this box, however, for example, lower case letters with descenders.

The height of this box has been normalized to 1.0 (this is usually the size of capital letters). The width of the box is font dependent. Users may define fonts with fixed or variable width characters. Users may also define fonts with white space included on either side of a character or on top and bottom. The Hershey character fonts currently provided have variable width characters and usually have white space on either side of the character. They, however, do not have white space on the top or bottom of capital letters.

The width of characters is used to determine the length of character strings. When a string is written characters are drawn with no space between them. That is why white space is usually included in the width of each character.

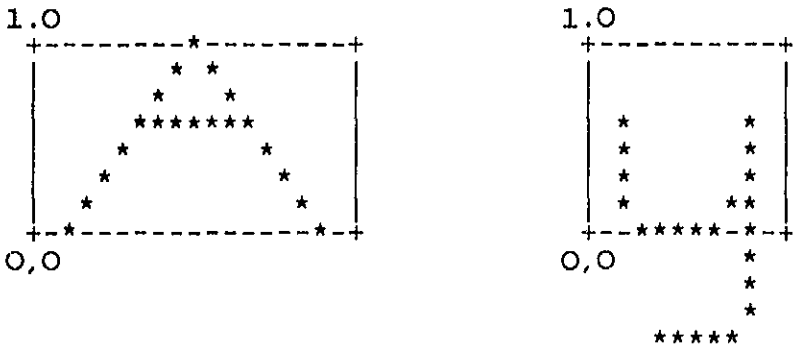

Fig. C-1. Examples of Variable Width Characters

Characters may have an ASCII value of 0 to 127. Characters not defined in a font file have a width of zero (0). A character that has a width but no vectors defines a blank or space.

C.4 FONT FILE DESCRIPTION

The basic file format is a record containing the numbers of characters in the font file followed by descriptions of each character.

A character description starts with a record containing the ASCII code value for the character, the number of coordinate pairs, and the nominal width of the character. This record is followed by zero or more coordinate pair records. Each coordinate pair record contains a move/draw flag and (X,Y) coordinates. The move/draw flag is a zero (0) for move and a one (1) for draw.

Character descriptions do not need to be in numerical order in the font file. If the file contains duplicate descriptions (the same ASCII code value), the last one encountered will be used and the others ignored.

C.5 PROGRAM TO READ FONT FILES

The following is a program fragment in FORTRAN that reads a font file. From this fragment the format of the font file can be easily determined.

```
      integer i,j
      integer N,CODE,MOVEDRAW,COORDPAIRS
      real    WIDTH,XCOORD,YCOORD

c     Open the font file

      open (1,file='OOO.FON',status='old',readonly)

c     Read the number of characters defined by the font file

      read (1,100) N

c     Loop - read the character descriptions in the font file

      do i = 1,N

c        Read character code, Number of coordinate pairs
c        and the nominal character width

         read (1,101) CODE,COORDPAIRS,WIDTH

c        Loop - read the coordinate pairs defining vectors

         do j = 1,COORDPAIRS

c           Read move/draw flag (0-move,1-draw), X coordinate
c           and Y coordinate

            read (1,102) MOVDRAW,XCOORD,YCOORD

         end do

      end do

100   format (I4)
101   format (I4,2X,I4,2X,E12.7)
102   format (I5,2X,F12.7,2X,E12.7)
      end
```

Fig. C-2. FORTRAN Program that Reads Font Files

APPENDIX D

SAMPLE PROGRAMS AND CODE FRAGMENTS

All of the sample programs and code fragments (see Figs. D-1 through D-10) in this appendix are from running programs. They have been reduced and rewritten for this document. Errors or omissions could have occurred in the translation.

Sample High Level Interactive Program

```
      integer*2  d,unit,config(4),device(4)
      integer    xddopen,xddactivate,xddconfigure,xddfree

      data       config/0,0,0,0/,device/0,0,0,0/

      parameter  (unit = 1)

c  Open, activate and configure virtual frame buffer.
c  Stop if there is a problem.

      if (.not.XDDOPEN(Unit)) then
         goto 10
      else if (.not.XDDACTIVATE(unit,.TRUE.)) then
         goto 10
      else if (.not.XDDCONFIGURE(unit,config)) then
         goto 10
      end if


c--- Main loop of display program.  Interactive programs
c--- obtains a command, execute the command and then loop
c--- back to obtain a new command.


c  Deallocate and free the display device

      call XDDACTIVATE(unit,.FALSE.)
10    call XDDFREE(unit)

      end
```

Fig. D-1. Sample Interactive Program

Change the configuration of display unit U from a full color configuration to a pseudocolor configuration. Image memory plane 3 contains the image. Do not bypass the LUTs. Use LUT section 1.

```
integer xdlconnect

if (.not.XDLCONNECT(unit,3,1,1,.FALSE.)) then
  type *,'Can not connect IMP 3 to LUT 1'
end if

if (.not.XDLCONNECT(unit,3,2,1,.FALSE.)) then
  type *,'Can not connect IMP 3 to LUT 2'
end if

if (.not.XDLCONNECT(unit,3,3,1,.FALSE.)) then
  type *,'Can not connect IMP 3 to LUT 3'
end if
```

Fig. D-2. Converting from Full Color to Pseudocolor Configuration

Draw a triangle in image memory plane 3 with a pixel value of 200.

```
integer*2 x(4),y(4)
integer   status,xdipolyline

x(1) = 100
y(1) = 100
x(2) = 100
y(2) = 200
x(3) = 200
y(3) = 200
x(4) = 100
y(4) = 100

status = XDIPOLYLINE(unit,3,200,4,x,y)

if (.not.status) type *,' Error Return Code ',status
```

Fig. D-3. Drawing a Triangle With Vectors

Draw the text "ABC" in image memory plane 1 using text font 4. Rotate text 30 degrees above the horizontal, make the height of the characters 30 pixels and horizontal scale factor 1.5. Write the text with a pixel value of 230. Draw the text at location x=200,y=200 and location flag = 1.

```
      byte str(3)
      data str /'A','B','C'/
      call xdtfont(4)
      call xdtcolor(230,0)
      call xdtrotate(30.0)
      call xdtsize(30,1.5)
      call xdtttext(unit,1,200,200,1,3,str)
```

Fig. D-4. Writing Text Into an Image Memory Plane

Return TRUE if two points are within a given range of each other; otherwise return FALSE. This function is used by several of the following code samples.

```
      logical function proximity (x1,y1,x2,y2)
      integer*2 x1,y1,x2,y2,delta
      parameter (delta = 4)
      proximity = .FALSE.
      if (abs(x1-x2).gt.delta) return
      if (abs(y1-y2).gt.delta) return
      proximity = .TRUE.
      return
      end
```

Fig. D-5. Testing Two Points for Proximity

```
Wait until a given switch changes state.
   code = 1       wait until switch goes from on to off
   code = 2       wait until switch goes from off to on
   code = 3       wait until switch is off
   code = 4       wait until switch is on
```

```
      subroutine switchwait (unit,device,switch,code)

      integer*2 unit,device,switch,code,value

      if (code.eq.1) then

10       call xdxswitch(unit,device,switch,value)
         if (.not.value) goto 10
11       call xdxswitch(unit,device,switch,value)
         if (value) goto 11

      else if (code.eq.2) then

20       call xdxswitch(unit,device,switch,value)
         if (value) goto 20
21       call xdxswitch(unit,device,switch,value)
         if (.not.value) goto 21

      else if (code.eq.3) then

30       call xdxswitch(unit,device,switch,value)
         if (value) goto 30

      else if (code.eq.4) then

40       call xdxswitch(unit,device,switch,value)
         if (.not.value) goto 40

      end if

      return
      end
```

Fig. D-6. Waiting for a Switch to Change States

Draw a series of connected vectors based on the position of cursor number 1. The cursor is in autotrack mode and switch 1 is used to set a vertex. Write the vectors in image memory plane 1 with a pixel value of 255. To terminate place the last vertex on top of the next-to-last vertex. Save the vertex list.

```
      integer*2 n,nn,v,d(4),x(100),y(100)
      integer   status,xdopen,xddactivate,xddconfigure
      integer   xdifill,xdcset,xdcon,xdcautotrack
      logical   loop,proximity
      data      d /0,0,0,0/

      type *,'  XDDOPEN         ',xddopen(unit)
      type *,'  XDDACTIVATE     ',xddactivate(unit)
      type *,'  XDDCONFIGURE    ',xddconfigure(unit,d)
      type *,'  XDIFILL         ',xdifill(unit,1,0)
      type *,'  XDCSET          ',xdcset(unit,1,255,255)
      type *,'  XDCON           ',xdcon(unit,1,1,0)
      type *,'  XDCAUTOTRACK    ',xdcautotrack(unit,1,1,1)

      call xdxswitch(unit,1,1,v)    ! clear switch setting

      n = 0
      loop = .TRUE.
      do while (loop)
         n = n + 1
         call switchwait(unit,1,1,1)
         call xdclocation(unit,1,x(n),y(n))
         if (n.gt.1) then
            nn = n - 1
            if (proximity(x(n),y(n),x(nn),y(nn)) then
               n = n - 1
               loop = .FALSE.
            else
               call xdipolyline(unit,1,255,n,x,y)
            end if
         end if
      end do
```

Fig. D-7. Using the Cursor to Draw Vectors

Given: Two arrays (X,Y) containing N vertex points that have been drawn with XDIPOLYLINE into image plane 1 with a value of 255 (see previous code example). Let the user pick a vertex and move it. Draw the new line(s). Use device 1, switch 1 and cursor 1.

```
      subroutine movevertex (n,x,y)

      integer*2 i,n,xx,yy,x(100),y(100)
      logical   proximity

      if (n.lt.1) return

      call xdxswitch(unit,1,1,v)      ! clear switch setting

      call switchwait(unit,1,1,1)     ! wait for switch
      call xdclocation(unit,1,xx,yy)  ! get cursor location

      do i = 1,n
         if (proximity(xx,yy,x(i),y(i))) then
            call switchwait(unit,1,1,1)
            call xdclocation(unit,1,xx,yy)
            call xdipolyline(unit,1,0,n,x,y)
            x(i) = xx
            y(i) = yy
            call xdipolyline(unit,1,255,n,x,y)
            return
         end if
      end do

      type *,'cursor is not over a vertex'

      return
      end
```

Fig. D-8. Using the Cursor to Pick a Vertex and Move It

Draw a test pattern in a full color frame buffer with graphics overlay. Assume a standard full color configuration.

```
      integer*2 maxvalue
      byte      red(3),green(5),blue(4)
      data  red   /'R','e','d'/
      data  green /'G','r','e','e','n'/
      data  blue  /'B','l','u','e'/

      call xdifill(unit 1,0)                    ! zero image
      call xdifill(unit 2,0)                    !   memories
      call xdifill(unit 3,0)
      call xdifill(unit 4,0)

      call xdlramp(unit,1,0)                    ! load linear LUT
      call xdlramp(unit,2,0)                    !   ramps
      call xdlramp(unit,3,0)

      call xdlconstant(unit,1,255,255,255)      ! load graphics
                                                !   LUT
      call xdiawset(unit,1,51,51,300,300)       ! draw red box
      call xdifill (unit,1,255)

      call xdiawset(unit,2,199,51,450,300)      ! draw green box
      call xdifill (unit,2,255)

      call xdiawset(unit,3,101,199,400,450)     ! draw blue box
      call xdifill (unit,3,255)

      call xdiawset(unit,1,1,1,512,512)         ! reset access
      call xdiawset(unit,2,1,1,512,512)         !   windows
      call xdiawset(unit,3,1,1,512,512)

      call xddinfo(unit,38,1,maxvalue)          ! label boxes
      call xdtcolor(maxvalue,0)
      call xtdsize(20,0.0)
      call xdtrotate(0)
      call xdtfont(0)
      call xdttext(unit,4,100,100,1,3,red)
      call xdttext(unit,4,325,100,1,5,green)
      call xdttext(unit,4,256,400,2,4,blue)
```

Fig. D-9. Drawing a Full Color Test Pattern

Draw a test pattern (same as previous example) in a pseudo color frame buffer with one image memory plane and no graphics overlay. Assume a standard pseudocolor configuration.

```
      integer*2 rlut(256),blut(256),glut(256)

      data  rlut /255,  0,  0,255,  0,255,255,249*0/
      data  glut /  0,255,  0,255,255,  0,255,249*0/
      data  blut /  0,  0,255,  0,255,255,255,249*0/

      call xdifill(unit,1,0)                    ! zero image
                                                !   memory
      call xdlwrite(unit,1,1,rlut)              ! load LUTs
      call xdlwrite(unit,2,1,glut)
      call xdlwrite(unit,3,1,blut)

      call xdiawset(unit,1,51,51,300,300)       ! draw red box
      call xdifill (unit,1,1)

      call xdiawset(unit,1,199,51,450,300)      ! draw green box
      call xdifill (unit,1,2)

      call xdiawset(unit,1,101,199,400,450)     ! draw blue box
      call xdifill (unit,1,3)

      call xdiawset(unit,1,199,51,300,300)      ! draw red/green
      call xdifill (unit,1,4)                   !   box

      call xdiawset(unit,1,199,300,400,300)     ! draw green/blue
      call xdifill (unit,1,5)                   !   box

      call xdiawset(unit,1,101,199,300,300)     ! draw red/blue
      call xdifill (unit,1,6)                   !   box

      call xdiawset(unit,1,199,199,300,300)     ! draw red/green/
      call xdifill (unit,1,7)                   !   blue box

      call xdiawset(unit,1,1,1,512,512)         ! reset access
                                                !   window
      call xdtcolor(7,0)                        ! lable boxes
      call xtdsize(20.0,0)
      call xdtrotate(0)
      call xdtfont(0)
      call xdttext(unit,4,100,100,1,3,red)
      call xdttext(unit,4,325,100,1,5,green)
      call xdttext(unit,4,256,400,2,4,blue)
```

Fig. D-10. Drawing a Pseudocolor Test Pattern

APPENDIX E

SAMPLE CURSOR SHAPES

Many frame buffers allow users to define the cursor shapes, usually with a bit mask. This appendix contains some of the shapes found useful by the MIPL users. In the following diagrams the symbol "*" indicates a bright white pixel and the symbol "#" indicates a black pixel. The rest of the cursor is transparent and allows the image to be seen. The size of the cursor is fixed and allows the user to estimate the sizes of objects in the image.

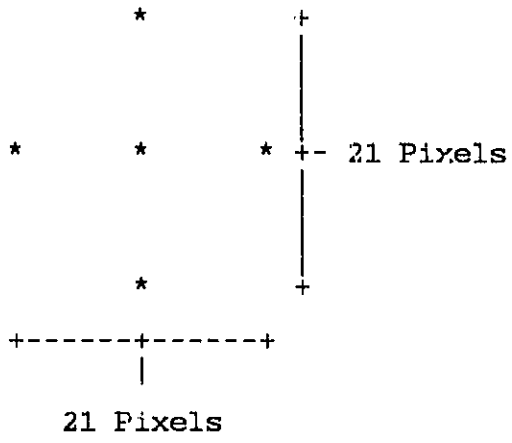

Fig. E-1. Cursor A

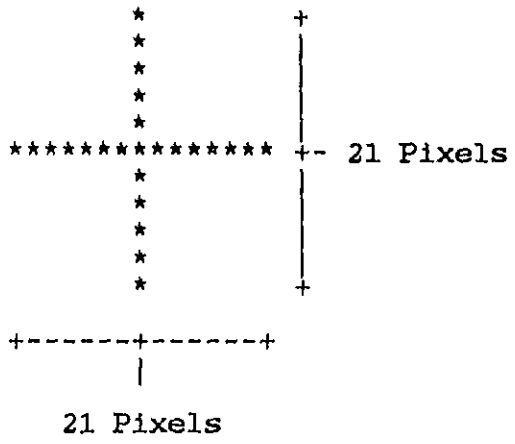

Fig. E-2. Cursor B

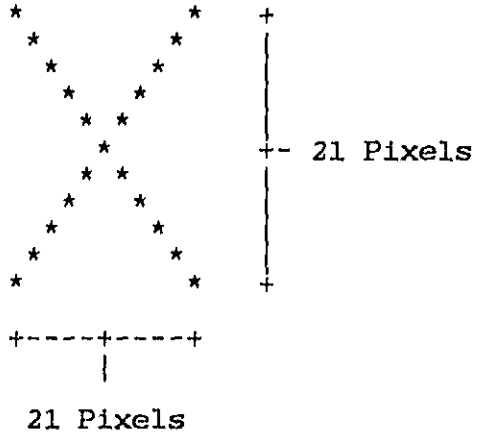

Fig. E-3. Cursor C

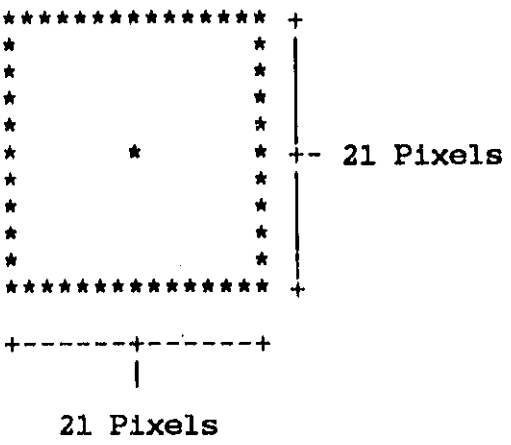

Fig. E-4. Cursor D

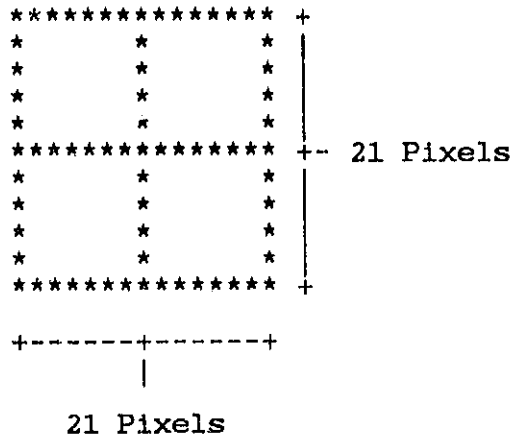

Fig. E-5. Cursor E

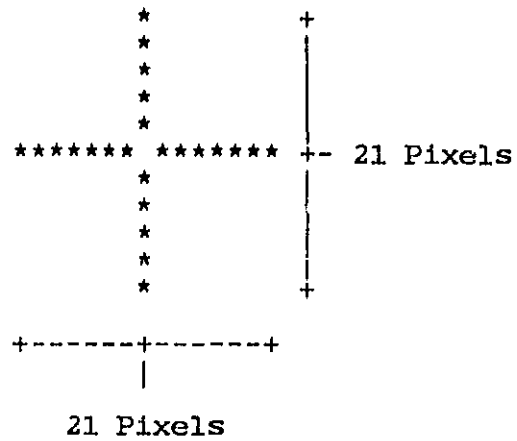

Fig. E-6. Cursor F

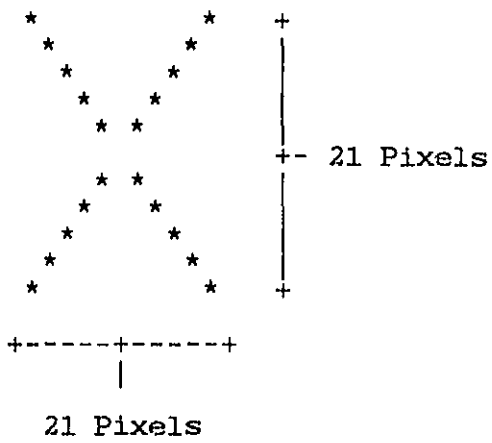

Fig. E-7. Cursor G

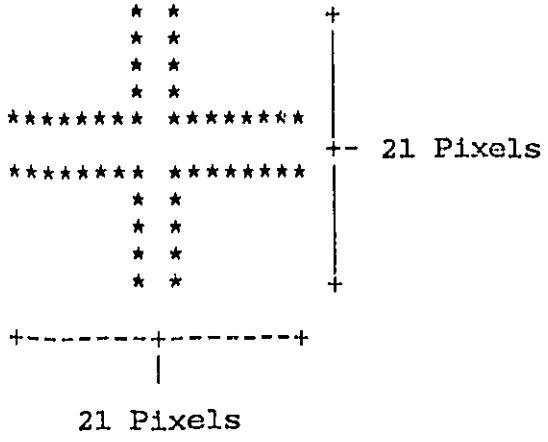

Fig. E-8. Cursor H

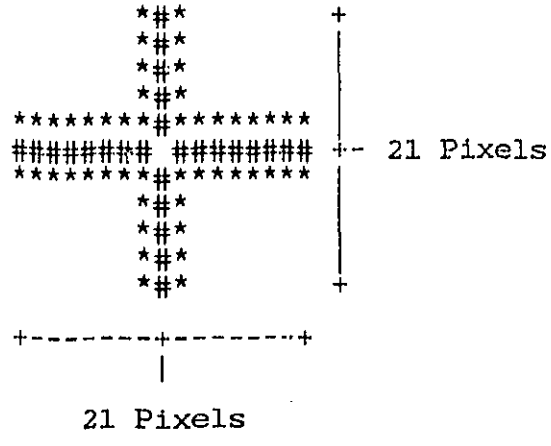

Fig. E-9. Cursor I

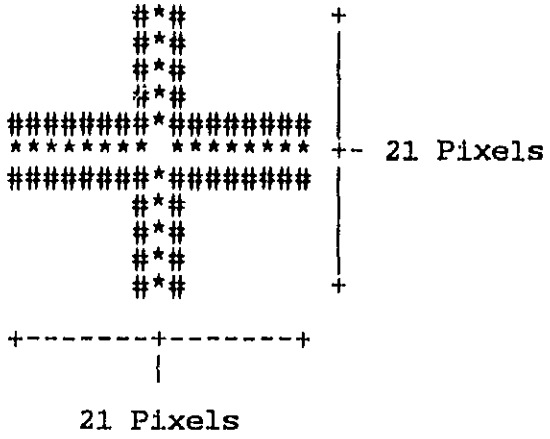

Fig. E-10. Cursor J

Index

Index