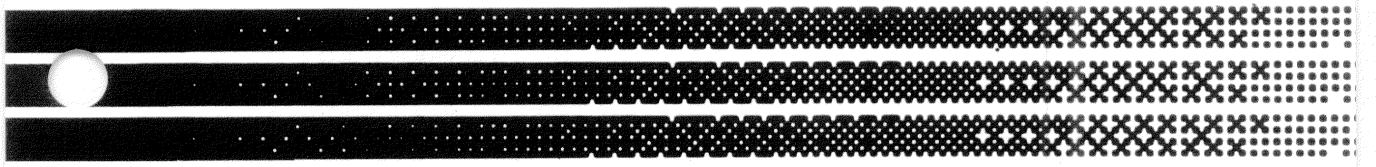
EK-104AA-TM-001

VCB02
Video Subsystem
Technical Manual

digital™

Technical Manual

EK-104AA-TM-001



VCB02
Video Subsystem
Technical Manual

Prepared by Educational Services
of Digital Equipment Corporation

First Edition, February, 1987

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

FCC Notice: This equipment generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference in which case the user at his own expense may be required to take measures to correct the interference.

The manuscript for this book was created on a VAX-11/780 system running WPS-PLUS. The book was produced by Educational Services Development and Publishing in Marlboro, MA.

The following are trademarks of Digital Equipment Corporation:

digital™	Letterprinter	PDT
DEC	Letterwriter	RSTS
DECnet	LSI-11	RSX
DECsystem-10	MASSBUS	TOPS-10
DECSYSTEM-20	MicroPDP-11	TOPS-20
DECUS	MicroVAX	ULTRIX
DECwriter	MicroVMS	UNIBUS
DIBOL	MINC	VAX
EduSystem	OMNIBUS	VMS
IAS	OS/8	VT
LA	PDP	VAXstation

FIGURES

TABLES

This manual provides detailed technical information on the features, specifications, configurations, installation, functions, logic, programming, and diagnostics of the VCB02 video subsystem as installed in a VAXstation II/GPX workstation system environment. It is intended for use by original equipment manufacturers (OEMs) and internal/external corporate end-users.

The manual has five chapters, four appendices, and a glossary.

- **Chapter 1, General Description,** provides an overview of the VCB02 video subsystem's features and specifications.
- **Chapter 2, Configurations and Installation,** describes the VCB02 video subsystem implementation in a MicroVAX color graphics workstation configuration, and includes an installation section.
- **Chapter 3, Functional Description,** describes the functional characteristics of major logic elements in the VCB02 video subsystem - the address processor and video processor chips, timing generator, bitmap memory, DMA gate array (CMOS), template RAM, keyboard/mouse/tablet input circuitry, Q22-Bus interface, and video output path.
- **Chapter 4, Programming Information,** describes how common drawing operations (for example, text, graphics, functions, and windowing) are implemented on the VCB02 video subsystem. Examples are included wherever needed. VCB02 drawing modes, DMA use, and program input (mouse/tablet support) are also examined.
- **Chapter 5, Diagnostics,** describes the diagnostic programs that apply to VCB02 video subsystem fault isolation, and includes a summary of error types, indicator locations, display messages, and suggested remedial actions.
- **Appendix A, Q22-Bus Specification,** profiles the Q22-Bus, which interfaces the VCB02 video subsystem.
- **Appendix B, LK201 Keyboard Specification,** describes the device's characteristics, and its programmable functions.

- **Appendix C, Mouse Specification,** describes the characteristics of the supplied 3-button circular mouse.

- **Appendix D, Digitizing Tablet Specification,** describes the characteristics of the optional, 11-inch square, digitizing tablet, which comes with a 2-button stylus and 4-button puck.

- The **Glossary** defines many important terms used in this manual.

RELATED DOCUMENTATION

For a list of related documentation, see Appendix 2 of either the VAXstation II/GPX Owner's Manual, BA123 Enclosure (EK-105AA-OM), or the VAXstation II/GPX Owner's Manual, BA23 Enclosure (EK-106AA-OM).

CHAPTER 1
GENERAL DESCRIPTION

1.1 INTRODUCTION

The VCB02 video subsystem consists of a base module and one or two 4-plane modules. These provide a high-performance, high-resolution, full-page DMA color video subsystem based on the Q22-Bus. This raster scan video subsystem is capable of 4- or 8-plane color video memory display on a MicroVAX workstation. Both MicroVMS and ULTRIX-32w operating system support is provided.

The MicroVAX workstation is a single-user, stand-alone, 32-bit workstation. A standard system includes three megabytes of memory, a 19-inch color or monochrome monitor, a mouse, a keyboard, a loading device, and a fixed disk drive. Each of the two versions of the MicroVAX workstation (BA23 and BA123 enclosures) includes a VCB02 video subsystem of either 2 modules (4-plane) or 3 modules (8-plane). The 8-plane subsystem provides 1024 (horizontal) X 864 (vertical) pixel resolution of displayed graphics and text. The 2-module, 4-plane VCB02 video subsystem allows simultaneous displays of up to 16 colors or shades of gray; the 3-module, 8-plane subsystem allows up to 256.

The base module and 4-plane module communicate through a cabling scheme to provide a basic configuration of 4-planes (16 colors from a palette of 16.7 million), with an upgrade path to 8-planes (256 colors from a palette of 16.7 million).

1.2 BASE MODULE

The base module (M7169) shown in Figure 1-1 provides the hardware for the system interface, the user I/O interface, and the full-page color video support for 4- or 8-planes of video memory.

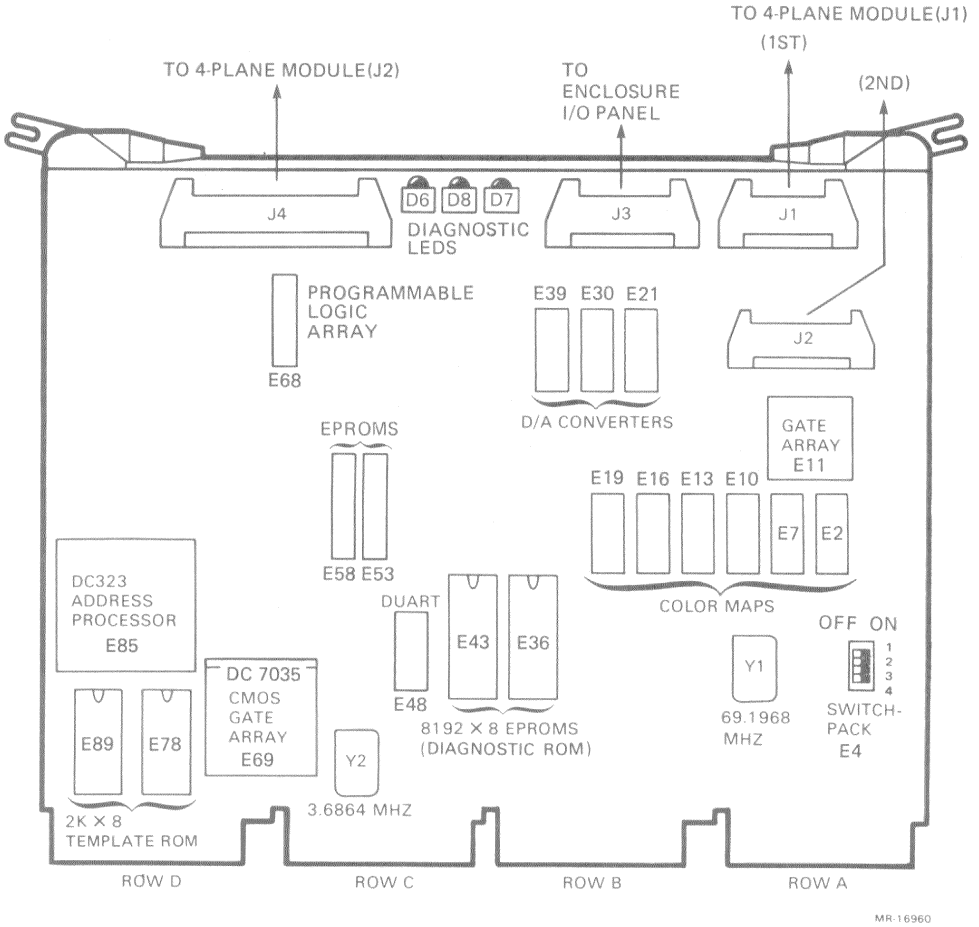


Figure 1-1 VCB02 Base Module (M7169)

Figure 1-2 is a block diagram of the base module's major hardware components:

1. Address processor (DC323) chip
2. DMA gate array
3. Template RAM for display list and cursor support
4. Q22-Bus interface transceivers
5. Serial I/O bus and address interface
6. Console emulation/diagnostic ROM
7. Subsystem timing generator
8. Color bitmap 8-plane video output path
9. Digital-to-analog (D/A) converters
10. Video shifter and cursor multiplexer
11. Cable connector for I/O interface and memory upgrade
12. Communications input (keyboard/mouse/tablet)

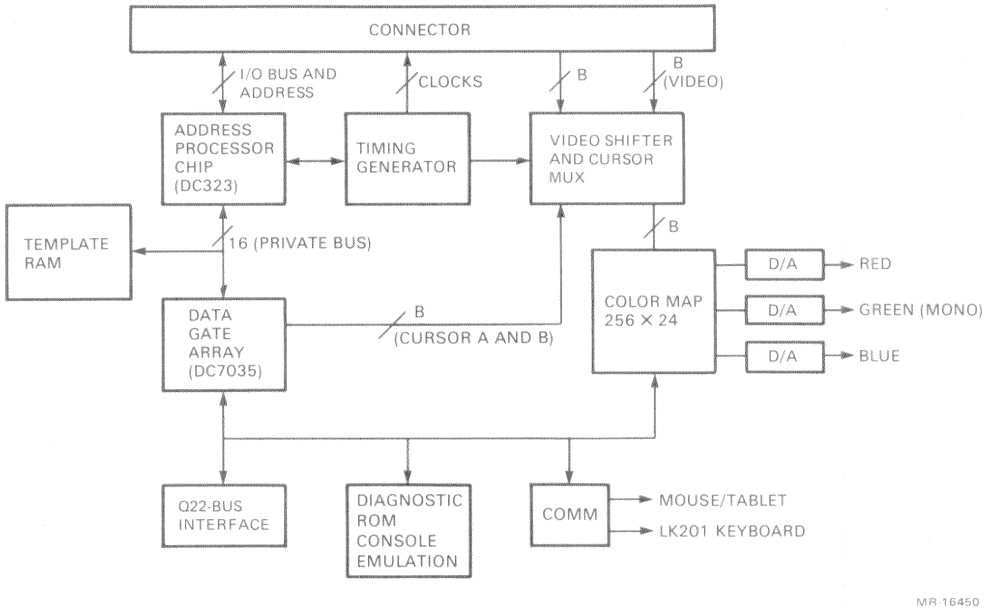


Figure 1-2 VCB02 Base Module, Block Diagram

1.3 FOUR-PLANE MODULE

The 4-plane module (M7168) shown in Figure 1-3 provides the hardware for 4-planes of full-page video memory, which is connected to the base module by a 50-pin ribbon cable. Two versions of this cable allow the base module to be connected to one or two 4-plane modules for a 4-plane or 8-plane color MicroVAX

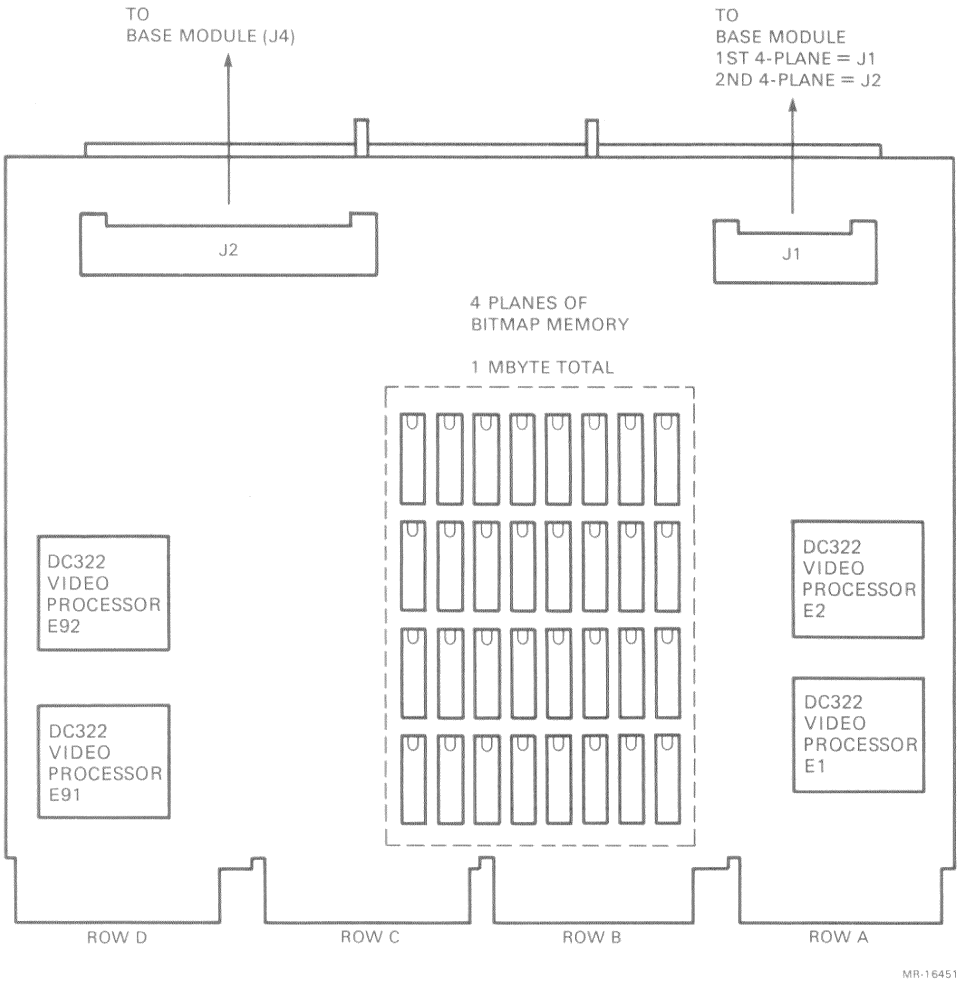


Figure 1-3 VCB02 4-Plane Module (M7168)

workstation. Figure 1-4 is a block diagram of the 4-plane module's major hardware components:

1. Four video processor (DC322) chips
2. 4-planes of video memory (2 pages per plane)
3. Subsystem support logic
4. Video shifter
5. Cable connector for base module interface

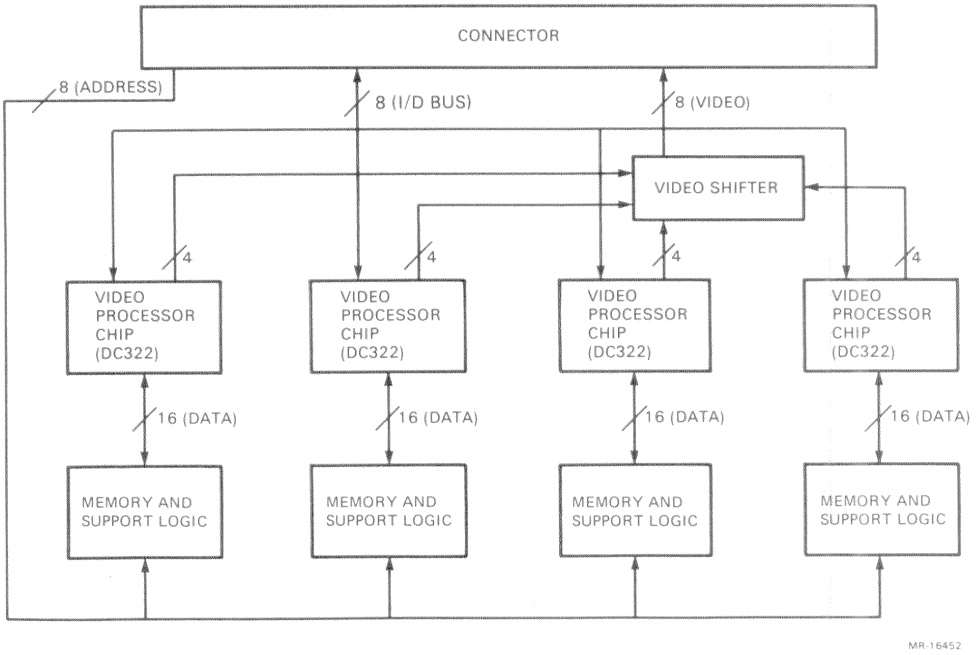


Figure 1-4 VCB02 4-Plane Module, Block Diagram

1.4 PRINCIPAL FEATURES OF THE VCB02 VIDEO SUBSYSTEM

The following are the principal features of the VCB02 video subsystem.

1. A bitmap display provides variable character size and positioning, with inherent graphics capability.

2. The display provides approximately 850,000 pixels, refreshed at 60 Hz to avoid flicker and smear of interlaced displays.

3. Arbitrary rectangular regions support multiple viewports on the screen. Each region can be smooth scrolled both vertically and horizontally at various speeds. Incoming data is clipped to the region boundaries. New data can be added to a region even while scrolling is in progress.

4. Return of clipping results can be used by the local processor to assist editing, windowing, and selection of algorithms.

5. The interface with the local processor supports either direct local processor access to the video system or passing of commands through a DMA controller.

6. Multiple bitmap planes (4 or 8) support applications requiring gray scale, color, or control planes. All planes can be manipulated simultaneously.

7. Z-axis addressing allows an alternate form of memory access that transfers the bits from each plane (or subplane) that correspond to one pixel, instead of adjacent X bits from one plane. This can also be used to easily program all planes for a color to be written.

8. Text attributes may be extended with additions like: arbitrary character size, arbitrary angle, italics, sub/superscript, and variable pitch fonts.

9. Rasterops that transform a rectangular source to a parallelogram destination of any size or orientation are provided. Halftone tiles or another image may be combined with any destination. Scaled or rotated rasterops (single source) operate at about 0.5 million pixels/s; normal rasterops operate at about 8 million pixels/s.

10. Rasterop mode allows creation of two-dimensional linear patterns of any cell size. Linear patterns orient to follow the drawing path.

11. Rasterop mode also allows the area between a series of vector pairs to be filled with a solid color, tile pattern, or an image at about 8 million pixels/s.

1.5 OPERATIONAL DESCRIPTION

The VCB02 video subsystem contains the following operational elements, shown in Figure 1-5.

- **Address Processor Chip (DC323)** — Responsible for functions common to all planes, such as: local processor interaction, all rasterop computations, bitmap address generation, clipping, screen refresh, scroll control, and monitor synchronization generation. The address processor chip controls the bitmap address bus (providing all addresses for screen refresh, scrolling, and updates), and the instruction/data (I/D) interconnect to configure and control the video processor chips.

- **Video Processor Chip (DC322)** — Provides the data path and control elements unique to each plane, such as: data FIFOs for refresh and scrolling, a barrel shifter for bit alignment, a logic unit with data and mask registers for memory modification, Z-axis address logic, and a control store RAM to define video processor chip operations during rasterops. The video processor chips are controlled by the I/D interconnect, exchange data with the bitmap memories on the memory bus, and provide screen refresh data on the video output bus.

- **Bitmap Memory** — Each plane consists of 1 K X 1 K X 2 pixels. This allows both on-screen and off-screen drawing to occur.

- **Video Output** — Provide a color map, video shift circuits, registers, and digital-to-analog converters for driving the monitor.

- **Timing Circuits** — Provide required system and memory clocks.

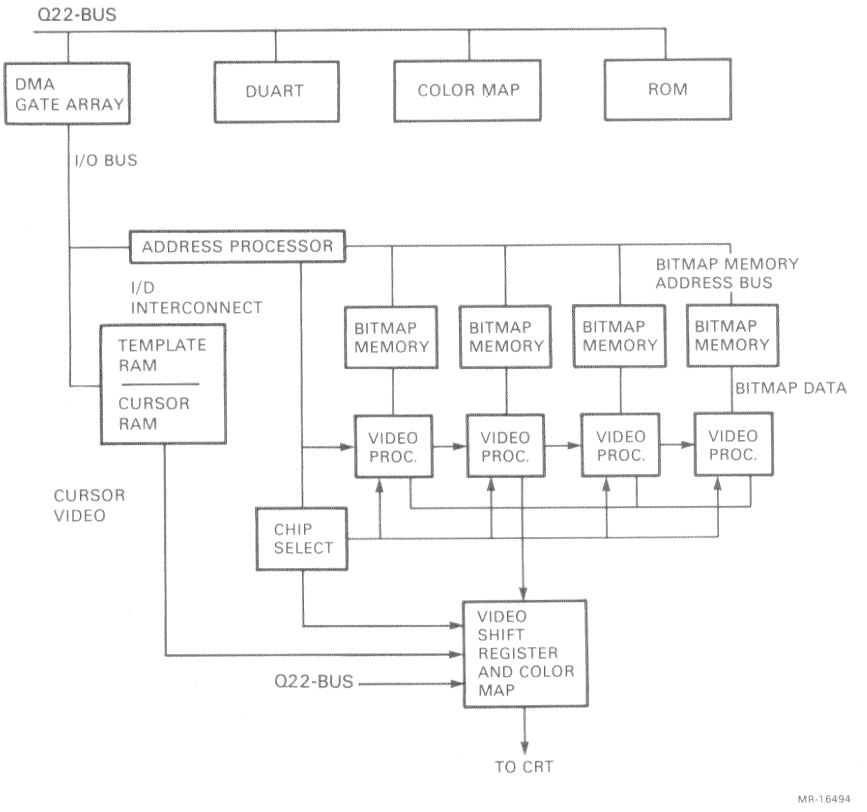


Figure 1-5 VCB02 Video Subsystem, Block Diagram

1.5.1 Multiplane Support

The VCB02 video subsystem uses the multiple video processor chips and bitmap memories to provide multiple memory planes for color, without any performance penalty for additional planes. It also has several other provisions to support manipulation of multiple planes.

1. A chip select mechanism controls which video processor chips (planes) will participate in bitmap update operations; a related mechanism controls participation in scrolling.

2. Z-axis (color variation) data transfers are provided to allow data to be exchanged between the MicroVAX CPU and the bitmap on the basis of a color value for each pixel, rather than exchanging the data for each plane separately. This form of data transfer can also be used to simultaneously load the color registers in all video processor chips. Z-axis transfers accommodate the use of low-resolution planes.

1.5.2 Viewport Support

The VCB02 video subsystem provides scrolling and clipping support for rectangular viewports. Scrolling is coupled to screen refresh, and moves all the data in an arbitrary rectangle (limited to 4 pixel boundaries, horizontally) up, down, left, or right by 1 to 15 pixels (or more) in a single frame time. The incoming edge of the scrolled region is filled with a programmable color; or, a mode is available that clears the entire region to a color.

Scrolling uses memory cycles that would otherwise be available for updating the bitmap. Therefore, the speed of rasterops drops to about one-third of its nonscrolling value when the whole screen is being scrolled. Down-scrolling actually moves the entire nonscrolling image (outside the scrolling rectangle) up, offsetting the entire screen to compensate. When down-scrolling of any region is in progress, rasterops are slowed to one-third speed, as if the whole screen were scrolling.

Clipping prevents updates from writing outside an arbitrary rectangle (same horizontal limitation as above, pending pass 3 address processor results). This rectangle may be the same as, or different from, the scrolling rectangle. Updates may occur concurrently with scrolling. This is facilitated by the addition of an index value to all update addresses. Indexes are changed with scrolling to represent where data has moved in the bitmap, which allows new updates to follow the scrolling image already in the bitmap.

1.5.3 Rasterops

All updates to the bitmap memory are performed by basic rasterops or various special rasterop modes. During rasterops, the address processor chip provides all memory address calculation, and the video processor chips provide all data modification. If no scaling, rotation, or reflection of the horizontal axis of the source is required in a rasterop, the VCB02 video subsystem automatically operates in fast mode. In fast mode, multiple pixels along the X-axis are operated on simultaneously. Otherwise, the VCB02 video subsystem operates in slow mode, processing only one pixel at a time. Fast mode operates 16 times faster than slow mode.

1.5.3.1 **Programmable Modes** -- Processor/bitmap transfers (not actually modes, but commands) transfer images between the MicroVAX CPU bus and the bitmap. These transfers work very much like rasterops, except that there is either a source or a destination in the bitmap, not both. Also, the data transferred to/from the MicroVAX CPU bus is a stream of words through a read/write I/O register the MicroVAX CPU or DMA may access. There are two transfer modes:

1. X-mode transfers each word (size depends on the memory configuration) to/from a single plane, with the adjacent bits in a word representing adjacent pixels in the X-axis. This type of transfer always operates in fast mode. Full indexing and shifting are available when transferring from the bitmap. But, when data is transferred to the bitmap from the MicroVAX CPU bus, the data must already be aligned with bitmap memory words because there is no shifter in the address processor chip.

2. Z-mode (color variation) transfers each word containing the bits representing the color of each pixel to/from the bitmap. This type of transfer always operates in slow mode, but writes all planes at the same time. Indexing is always available in this mode.

1.5.3.2 **Video Processor Data Manipulation** -- The video processor chips contain the data path for each plane of the bitmap. During source reads from the bitmap, a shifter aligns the data with the pixels in the destination. This shifter is controlled automatically by the address processor chip. Data that is read during source reads may be broadcast to other video processor chips or the address processor chip on the I/D interconnect (one video processor may broadcast), or may be retained in the video processor chip.

Data retained in the video processor chip or received from the I/D interconnect may be stored in any of three registers for use during a destination cycle. The source register provides 1 input to a 16-function logic unit; the other input is the old destination data. The output of the logic unit selects, on a pixel-for-pixel basis, a foreground or background color to form new destination data. The new destination data is masked by the

combined outputs of the two mask registers, which, like the source register, are loaded with a constant or during a source read. Masked bits return the old destination data instead of the new data to the bitmap. Either of the masks can be complemented, and the combined outputs adjusted so that only complete groups of bits are changed in low-resolution planes. A complementer and resolution logic are available for the source register.

Data manipulations are controlled by two sets of indirect registers that are selectable from the address processor chip. The control store RAM specifies the data transfers that occur on each source or destination cycle. Four logic function registers specify the logic and mask operation to be performed during the destination cycle. Having four function registers allows two orthogonal functions to be preprogrammed in the video processor chips.

1.5.3.3 **Performance** -- Destination-only and single-source/destination rasterops are compute-bound at 1 cycle every 2 us. A cycle writes 1 pixel in slow mode, or 1 word (16 bits) in fast mode. Therefore, 500 K pixels/s are written in slow mode and 8 M pixels/s are written in fast mode.

1.5.4 Private Memory

To achieve rasterop performance, parallel multiplane operation, and to support scrolling, the bit-mapped memory operated upon by the VCB02 video subsystem must be on a private bus accessible to the processor only through the chip set. In the VCB02 video subsystem address space, as much off-screen memory as displayable memory is provided in each plane to support occluding viewports. With DMA-assisted processor-to-bitmap and bitmap-to-processor transfers in either X- or Z-mode, the need for direct processor access to the bitmap is diminished.

1.5.5 MicroVAX CPU-to-Address/Video Processor Chip Interface

The MicroVAX CPU controls the address/video processor chips through the MicroVAX CPU bus on the address processor chip. This bus consists of 16 bidirectional data lines, 6 address lines, and some control lines. The address processor chip has 58 registers (mostly write-only) that can be accessed directly by the MicroVAX CPU. Alternatively, the registers can be loaded through an indirect register address counter. Address processor chip registers are manipulated directly. The following operations are also provided.

1. Video processor chip registers are loaded with explicit I/D interconnect instructions. The chip select register must be loaded to select the video processor chips that are to receive the data; this register is the I/D interconnect.

2. To invoke a rasterop, the MicroVAX CPU loads the required video processor and address processor chip registers (unless unchanged by previous use), and starts the update with a rasterop command.

3. Processor/bitmap transfers are similar to rasterops, except that the data is sent to or accepted from the address processor chip.

4. The cancel command terminates any update operation in progress and clears all rasterop status.

5. Scrolling is started by loading the address processor chip scroll constant register. All address processor and video processor scroll parameter registers must have been previously loaded. Separate I/D command, data and chip select registers are provided for loading scroll parameters into the video processor chips without disturbing any update that might be in progress.

A status register reports the progress of various processes in the address processor chip. Any or all of the status bits can be enabled to generate a request on either of two pins. (One pin is reserved for DMA requests, the second is used for interrupts.) Scroll status provides a top-of-frame flag to load scroll parameters, a programmable flag that will be set when the refresh/scroll process passes any point on the screen to synchronize region updates, and a bottom-of-frame flag to start any vertical blank activities. Rasterop and I/D status indicates when various parameters and data may be loaded to minimize waiting. Clipping status reports when some rasterops are clipped or not clipped.

1.5.5.1 **Address Processor Chip Registers** -- The following are control, scroll, rasterop, and configuration registers of the address processor chip.

CONTROL REGISTERS

Address Counter	Indirect access to address processor registers used during DMA
Status	Readable scroll, rasterop, I/D, and clipping status
Request Enable Interrupt Enable	Status bit masks for interrupt outputs
I/D Data	Data transfer register
Command	Update command register

SCROLL REGISTERS

Most of the scroll parameter registers are double-buffered to allow a whole frame time to load new scroll data.

Scroll X Min Scroll X Max Scroll Y Min Scroll Y Max	Scroll region boundaries
Pause	Programmable frame position status return
Y Scroll Constant Y Offset	Vertical scroll control
I/D Scroll Data I/D Scroll Command	I/D path for scroll control

RASTEROP REGISTERS

Pending X Index Pending Y Index New X Index New Y Index Old X Index Old Y Index	Index values for update control while scrolling
Clip X Min Clip X Max Clip Y Min Clip Y Max	Update region boundaries
Mode	Rasterop mode control
Fast Source 1 DX Slow Source 1 DY	Source 1 rectangular extent
Source 1 X Origin Source 1 Y Origin	Source 1 origin
Destination X Origin Destination Y Origin	Destination origin
Fast Destination DX Fast Destination DY Slow Destination DX Slow Destination DY	Destination parallelogram extent
Fast Scale Slow Scale	Source 1 destination scale factors
Source 2 X Origin Source 2 Y Origin	Source 2 origin
Source 2 Height and Width	Source 2 rectangular extent (powers of 2 only)
Error 1 Error 2	Optional error control for vector generators

CONFIGURATION REGISTERS

X Scan Configuration	8-frame timing and memory configuration
Y Scan Configuration	4-frame timing and memory configuration
X Limit	Limit of visible (scrollable) screen
Y Limit	

1.5.5.2 **Video Processor Chip Registers** -- The following are scroll, rasterop, and configuration registers of the video processor chip.

SCROLL REGISTERS

All the scroll parameter registers are double-buffered to allow a whole frame time to load new scroll data.

Scroll Constant	Horizontal scroll control
Fill	Background fill color for scrolling
Left Scroll Boundary	Subword scroll edges
Right Scroll Boundary	

RASTEROP REGISTERS

Logic Unit Function	(4) Indirect selection of logic function
Control Store RAM	(6) Indirect selection of data transfers
Mask 1	Rasterop data masks
Mask 2	
Source	Data input to logic unit
Foreground Color	Writing colors selected by logic function unit
Background Color	

CONFIGURATION REGISTERS

Bus Width	Memory bus width (screen size)
Plane Address	Z-mode bit address
Resolution Mode	Subplane mode

1.6 APPLICATIONS

The following applications provide examples of implementing common drawing operations using the VCB02 video subsystem.

1.6.1 Text

Normal rasterops are used to write characters. Common text procedures are described below.

1.6.1.1 Font Storage and Access -- Fonts are stored in an undisplayed portion of the bitmap. Ordinarily, a font is stored only in one plane and is transmitted from that plane to itself and others when a character is written. This results in characters that appear in one (foreground) color written on a background of another color, as is normally desired. If multicolored or gray-scale characters (for anti-aliasing) are desired, the fonts can be stored across many planes. Fonts are normally stored in the off-screen portion of the VCB02 bitmap.

1.6.1.2 Normal Text -- A display of unrotated, unscaled, fixed-pitch text might use the following sample settings in the address processor and video processor chips. Only those settings actually changing between characters need to be updated after initialization.

1. The update region is initialized with clipping limits, indexes, and resolution mode.

2. The video processor chip control store RAMs are set to enable broadcast of the character data from the plane containing the font to all other planes.

3. The address processor chip mode register and the video processor chip logic functions, foreground, and background are set for some of the desired character attributes.

4. The source 1 origin (X and Y) is set to the start of the first character to be transferred.

5. The destination origin (X and Y) is set to the desired screen location for the first character. The destination edge vectors are set parallel to the X and Y axes.

6. The scale registers are set for scale factors of 1.

The following operations can now be performed to write each character in a string.

1. The MicroVAX CPU loads the command register in the address processor chip to start a source 1/destination rasterop.

2. After waiting for the address processor chip to finish the initialization phase of the rasterop (but while the character is still being written), the local processor can reload the source and destination origins and the command so that the next character can start immediately upon completion of the first.

3. This process is repeated until the text is exhausted or an attribute change is required.

Adjusting the destination position (and possibly width) produces variable-pitch text (proportional spacing).

1.6.1.3 **Character Attributes** -- To set the character attributes, appropriate address processor and video processor chip registers are loaded. Only the registers for those attributes that actually change between strings need to be loaded. The following list describes the intended implementation of attributes.

1. Reverse -- Reverse is a specific case of a more general class of attributes that modifies the logic function being performed on the destination data. Two bits are provided in the address processor chip command word to address one of the four logic unit function registers in the video processor chips.

2. Underline -- Underlines can be drawn as a vector of the desired thickness, and linear pattern mode can be used if a dashed or other patterned underline is desired.

3. Overstrike -- Overstrike may be accomplished by loading (or not loading) the destination origin and the use of OR logic functions.

4. Invisible -- The pen up bit in the address processor chip mode register may be used to disable writing; otherwise, the MicroVAX CPU omits text strings that are invisible. If erasure of the invisible area is desired, this must be performed explicitly.

5. Intensity, Color, and Blink -- These are Z-axis (color variation) parameters implemented through the color map. To enable writing the appropriate color in the bitmap planes, a Z-axis load of the video processor chip foreground and/or background registers would be performed. Source data would then be passed through a video processor chip mask register to accomplish overstrike writing, or through the source register to accomplish replace mode writing. Otherwise, blink is performed by periodic writing to the bitmap.

6. Subscript and Superscript -- These character offsets are obtained by appropriate loading of the destination origin registers.

7. Size, Italics, and Rotation -- Any size, italic slope, or rotation is available by setting the destination edge vectors and the scale factors, and providing appropriate destination origin update.

1.6.2 Graphics

Most graphics functions use basic rasterops. Linear patterns, polygon fill and flood use the additional rasterop modes.

1.6.2.1 **Vectors** -- Solid color vectors are plotted as specific parallelograms, without any source operations. The foreground/background registers in the video processor chips are loaded with constants according to the desired color. The start point is loaded into the destination origin registers; one edge vector is set to the DX and DY of the desired vector; the other edge vector is set to the width of the desired vector (and perpendicular to the first edge); and the rasterop command is loaded. Curves must be formed from a string of straight vectors or points.

Vectors may be shaded in one of two ways:

1. Inclusion of a first source set for linear pattern mode provides linear patterns (like dashed lines, bullets, or stripes) that align themselves with the direction of a vector.

2. Inclusion of a second source set for tile mode provides tile patterns that are "uncovered" under the path of the vector.

1.6.2.2 **Fill Mode - Polygons** -- In general, a fill operation creates a solid or textured object whose shape is described mathematically, as by a list of vertices.

Various fill modes are used to fill polygons. Polygons are filled with solid colors (destination only), tile patterns, or from off-screen images (source 2 and destination). A whole polygon is filled in the following way.

1. The polygon may need to be subdivided into two parts intersected by a line parallel to the fill axis. (This method is more flexible than simply requiring a convex polygon.)

2. The address processor and video processor chips are configured for fill mode with the appropriate source and destination operations.

3. The vertices of the polygon are divided into two "sides" (such as "left" and "right", from the uppermost point to the lowermost point).

4. The rasterop origin registers are loaded with the top of the polygon.

5. The first pair of edge vectors (starting at the first vertex from each side) is loaded into the rasterop extent registers.

6. The address processor chip command register is then loaded with a rasterop command, including a source 2 if needed.

The space between the two vectors is filled parallel to the X (or Y) axis as the vectors are advanced synchronously in Y (or X).

When the address processor chip signals for more data, the MicroVAX CPU loads the vector associated with the next vertex in the list of vertices. It then loads the address processor chip command register again, continuing with the previous step until all required vectors are plotted.

For convenience, a simplified fill mode is provided. This mode fills from a single vector to either a horizontal or vertical baseline. Only a single vector path is required. The path is allowed to reverse the direction of its component that is parallel to the baseline. (This is not allowed with 2-line mode.) However, the space between two lines that converge on a point from the same quadrant is difficult to fill in this mode.

1.6.2.3 **Polygon Flood** -- In contrast to a fill operation, a flood operation colors or textures the interior of an area. The address processor chip provides no explicit support for a polygon flood function. This function requires the path of writing in the bitmap to be conditional on the previous contents of the bitmap. In the hardware, the path of writing is controlled by the address processor chip, but only the video processor chips accept the memory data. A flood operation is implemented in the following way.

1. The local processor/bitmap transfer facility reads a scan of the bitmap.

2. Each returned pixel value is checked for a match against a set of boundary colors, and a count is maintained of the pixels received.

3. Each time a match is found, the processor/bitmap transfer is terminated by a cancel command to the address processor chip.

4. A vector writes the desired pixels on the scan with a color or pattern.

5. The flood algorithm then steps to a point on the next scan or continues at a previous seed point.

1.6.2.4 **Objects** -- Ordinary rasterops can be used to move objects or windows on the screen. In a multiplane environment, rectangles can be copied by moving the pixels in all planes, in parallel, to new locations in the same planes.

To move a simple object, that is, to alter only the destination pixels within the boundary of the object, the hardware distinguishes between the object and the background. This can be done by defining the foreground of the object in one plane. This area is then transmitted to the other planes during the rasterop as a mask that prevents the alteration of background pixels. Otherwise, the second source (with fill mode, if desired) can be used to copy the object from off screen, with clipping to the destination shape.

It is also possible to move monochromatic objects (ones specified only by a foreground shape), and assign a color to them at the time they are moved. This type of operation is similar to that for the writing of text.

2.1 GENERAL

This chapter describes the configuration and installation of the VCB02 video subsystem in a MicroVAX workstation. With the VCB02 subsystem, the workstation can display graphics and text with 1024 (horizontal) X 864 (vertical) pixel resolution on a 19-inch monitor in color or shades of gray. The VCB02 video subsystem consists of two modules: the base module and the 4-plane module, shown interconnected in Figure 2-1. The open-end modular design of the VCB02 subsystem allows expansion to an 8-plane video subsystem with the addition of another 4-plane module, as shown in Figure 2-2.

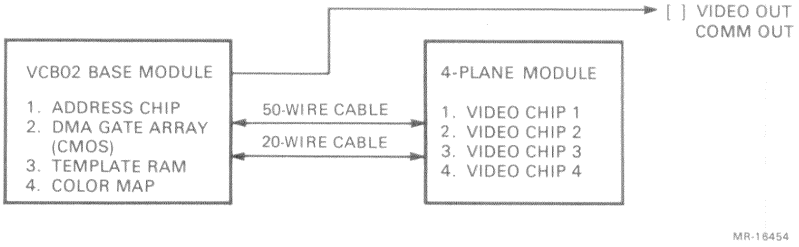


Figure 2-1 VCB02 Video Subsystem 4-Plane Configuration

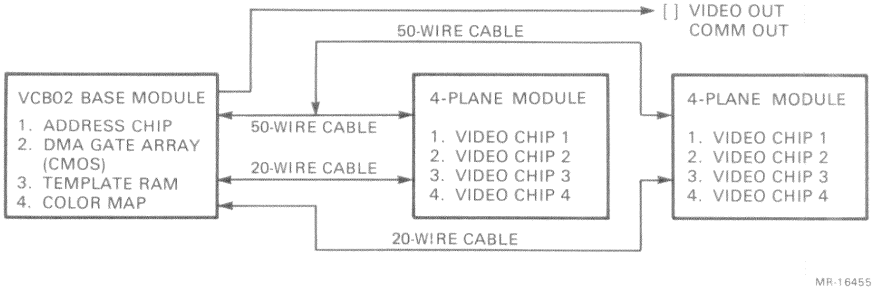


Figure 2-2 VCB02 Video Subsystem 8-Plane Configuration

2.2 SYSTEM CONFIGURATION

The VCB02 video subsystem is installed in a MicroVAX graphics workstation contained in either a BA23 or BA123 enclosure. The two versions of the MicroVAX workstation have some common features. Both use the MicroVAX II CPU module that includes the floating point unit (FPU) and one megabyte of memory. Each system uses the MS630 series of memory modules, and allows up to 9 megabytes of memory to be configured. The RQDX3 controller module interfaces various diskette and fixed disk storage devices. A VR260 monochrome monitor, or a 19-inch VR290 color monitor is supplied with either workstation system that includes the LK201 keyboard and VSXXX-AA serial mouse.

The BA123 enclosure contains one of the two versions of the VCB02 video subsystem: the 4-plane VCB02-B, or the 8-plane VCB02-C. The BA23 enclosure offers an 8-slot backplane which accepts only the 4-plane VCB02-B version of the subsystem because of power limitations. The BA123 enclosure offers a 12-slot backplane that allows for additional system options. It has space for four 5.25 inch disk drives.

Table 2-1 compares the configurations of the BA23 and BA123 enclosure color graphics workstation systems. (Refer to this manual's Preface for the appropriate system technical manual for complete configuration rules and procedures.)

Table 2-1 BA23/BA123 Enclosure Workstation Configurations

Description	BA23	BA123
MicroVAX II FPU/1 MB	KA630-AA	KA630-AA
Memory Expansion Module	MS630-BA(2 MB)	MS630-BA (2 MB) Supplied MS630-CA (4 MB) Optional
VCB02 Base Module	M7169	M7169 (see Note 1)
VCB02 4-Plane Module	M7168	M7168
VCB02 4-Plane Module	Unavailable	M7168
Ethernet Interface	DEQNA-KP	DEQNA-KP
Disk Drive Controller	RQDX3-AA	RQDX3-BA
Diskette Drive	RX50Q-AA	RX50Q-BA
Fixed Disk Drive	RD53Q-AA	RD53Q-BA
Serial Mouse	VSXXX-AA	VSXXX-AA
Keyboard	LK201-AA	LK201-AA
Monitor	VR260/VR290	VR290 (see Note 2)
Monitor Cable	BC18Z-10/ BC18P-10	BC18Z-10 (see Note 2)

NOTES

1. Both systems have color video capability. The VS265 monitor requires a 4-plane VCB02-B option, while the VS270 monitor requires an 8-plane VCB02-C option.

2. The VR260 monitor's cable can be substituted with a BC18P-10 cable. When used with the 4-plane or 8-plane VCB02 options, the VR260 monitor will display shades of gray.

2.2.1 System Components

Table 2-2 lists the backplane slot assignments for the system modules of both the BA23 and BA123 enclosures.

Table 2-2 Backplane Slot Assignments

Slot	Module	Description	Rows	Type
1	M7606-AA	MicroVAX II CPU (KA630-AA)	A-D	Quad
2	M7608-AA	2 Mbyte Memory (M5630-BA)	A-D	Quad
3	M9047	Grant card	A-B	Dual
4	M7504	DEQNA Ethernet interface	A-B	Dual
5	M7169	VCB02 base module (see Note 3)	A-D	Quad
6	M7168	VCB02 4-plane module (see Note 3)	A-D	Quad
7	M7168	VCB02 4-plane module (see Note 3)	A-D	Quad
8	M7513	RQDX3 disk controller	C-D	Dual
8-12		Option slots		Dual/Quad
13	M9058	Signal distribution board Reserved for future use	C-D A-B	Dual Dual

NOTES

These configuration notes apply equally to the BA23 and BA123 enclosures.

1. KA630-AA -- The quad-height KA630-AA (MicroVAX II with FPU and 1 Mbyte of memory) module must be installed in the first slot of the backplane. The A-B rows of this slot are connected to the Q22-Bus, its C-D rows are connected to the CD interconnect.

2. MS630-BA -- The quad-height MS630-BA (2 Mbyte memory) module requires a full slot in the backplane. The slot must have the CD interconnect available and be adjacent to the MicroVAX CPU, which limits the memory module to slot 2.

3. VCB02 -- The video subsystem modules can be installed in any of slots 5 through 8, but occupy adjacent slots because of intermodule cable length.

4. RQDX3 -- The dual-height RQDX3 interface module must be installed in a Q22-Bus slot in the backplane.

2.2.1.1 **BA23 Enclosure** -- The BA23 enclosure supports the 4-plane monochrome color graphics MicroVAX workstation. It has the following components.

- 230 W power supply
- 8-slot backplane
- 1 or 2 5.25-inch storage units
- Rear I/O distribution panel
- Front control panel and assembly

The 8-slot BA23 backplane is configured as shown in Figure 2-3. (See also the notes to Table 2-2.)

2.2.1.2 **BA123 Enclosure** -- The BA123 enclosure supports the 4- or 8-plane monochrome/color graphics MicroVAX workstation. It has the following components.

- 460 W power supply
- 12-slot backplane
- Up to 4 5.25-inch storage units
- Rear I/O distribution panel
- Front control panel and assembly

The 12-slot BA123 backplane is configured as shown in Figure 2-4. (See also the notes to Table 2-2.)

2.2.1.3 **KA630-AA CPU Module** -- The quad-height KA630-AA processor module has the following components and functions.

- MicroVAX II processor chip
- Floating Point Unit chip
- Q22-Bus interface (See Appendix A.)
- Q22-Bus map for DMA transfers
- 1 Mbyte of on-board memory
- Addresses up to 16 megabytes of physical memory
- 1 serial line unit
- 64-Kbyte boot PROM
- 10 ms interval timer interrupts enabled by the internal processor register

The KA630-AA module supports the following subset of the VAX data types: byte, word, longword, quadword, character string, and variable-length bit field. The floating point unit supports ffloating, dfloating, and gfloating. Support for the remaining VAX data types can be provided via macrocode emulation.

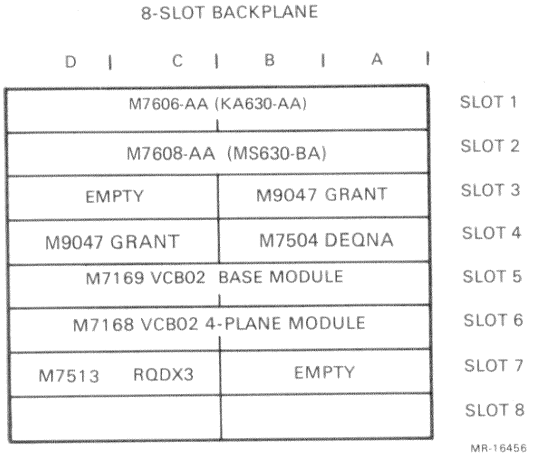


Figure 2-3 BA23 Backplane Configuration

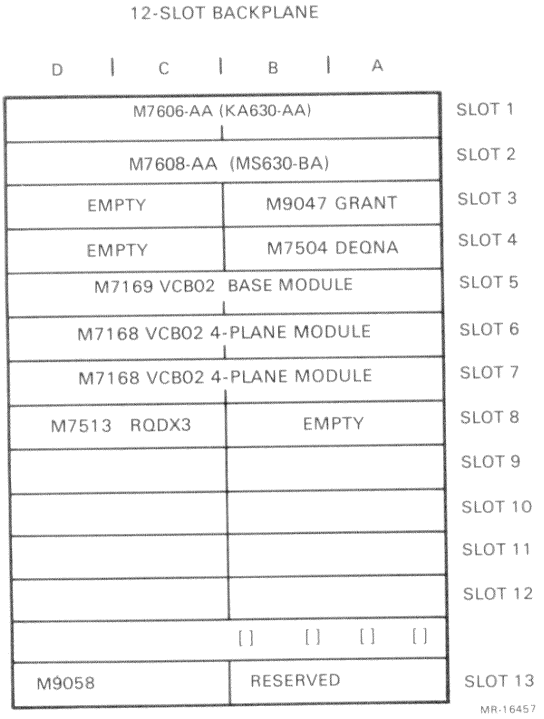


Figure 2-4 BA123 Backplane Configuration

The KA630-AA module implements the following subset of the VAX instruction set: integer and logical, address, variable-length bit field, control, procedure call, miscellaneous, queue, MOVC3/MOVC5, and operating system support. Floating point is implemented with an optional floating point chip. The remaining VAX instructions are implemented via macrocode emulation. (The MicroVAX CPU chip provides microcode assists for the emulation of the character string, decimal string, EDITPC and CRC instructions.) The KA630-AA module also provides a demand paged memory management unit that is fully compatible with VAX memory management.

2.2.1.4 **MS630 Memory Expansion Module** -- The MS630 series of memory modules use the local memory interconnect, which consists of the CD interconnect and a cable, to create a direct path with the MicroVAX II CPU. This allows much faster memory access because the memory responds directly to the MicroVAX CPU instead of routing through the Q22-Bus.

The MS630 modules contain parity control and self-identification logic. A maximum of two MS630 modules are allowed. If only one MS630 module is installed, it must occupy the next slot after the MicroVAX CPU (CD interconnect required). A second MS630 module would occupy the A-B rows of this slot. For detailed information on these modules, refer to the VAXstation II/GPX Technical Manual, BA123 Enclosure (AZ-GNFAA-MN).

Three variations of the MS630 module are used in various combinations in each system.

Version	Storage	Module Height	Module No.
MS630-AA	1 Mbyte	Dual	M7607-AA
MS630-BA	2 Mbyte	Quad	M7608-AA
MS630-CA	4 Mbyte	Quad	M7608-BA

2.2.1.5 **VCB02 Video Subsystem** -- The VCB02 video subsystem is available in two versions: 4-plane and 8-plane. Both versions are capable of monochrome and color operation. The BA23 enclosure workstation system supports the 4-plane VCB02 video subsystem, while the BA123 enclosure workstation system supports both the 4- and 8-plane versions. Each video subsystem supports two RS232 serial ports. The LK201 keyboard uses one port, and either the VSXXX-AA mouse or the optional VSXXX-AB digitizing tablet uses the other.

The VCB02-B module set option comprises a base module and a 4-plane module (two quad-height modules total). The BC18Z-10 cable, used for color operation, connects the enclosure to the VR290 color monitor. The BC18P-10 cable, used for gray-shade operation, connects the enclosure to the VR260 monochrome monitor.

The VCB02-C module set option comprises a base module and two 4-plane modules (three quad-height modules total). The BC18Z-10 cable, used for color operation, connects the enclosure to the VR290 color monitor. The BC18P-10 cable, used for gray-shade operation, connects the enclosure to the VR260 monochrome monitor.

VCB02 Video Interfaces
The VCB02 subsystems are Q22-Bus options. Performance is enhanced by the use of the VCB02 DMA gate array, which allows data to be transferred quickly between the host system and the VCB02 bitmap.

- Base Module — The VCB02 base module supports the system/user interface and either 4 or 8 planes of full-page video memory.

- 4-Plane Module — The VCB02 4-plane module supports 4 planes of video memory connected to the base module through a 50-pin ribbon cable. Two versions of this cable allow the base module to be connected to one or two 4-plane modules for 4- or 8-plane color systems. A separate 20-pin cable is used for control signals between the 4-plane module and the base module.

2.2.1.6 **Mass Storage** -- The following mass storage devices can be used in configuring the color graphics workstation system.

- RQDX3 Controller — The dual-height RQDX3 controller module is the interface to the RX diskette and RD Winchester disk drives. It is a direct memory access (DMA) interface and uses mass storage control protocol (MSCP).

- RX50-M Diskette Drive — The RX50-M dual diskette drive uses 5.25-inch diskettes. Each diskette can store 400 Kbytes of data. For more information, refer to the VAXstation II/GPX Technical Manual, BA123 Enclosure (AZ-GNFAA-MN).

- RD53 Fixed Disk Drive — The RD53 Winchester fixed disk drive contains a 5.25-inch disk with a 71 Mbyte formatted capacity. For more information, refer to the VAXstation II/GPX Technical Manual, BA123 Enclosure (AZ-GNFAA-MN).

- TK50 Tape Drive — The TK50 streaming tape subsystem uses tape cartridges, each with a formatted capacity of approximately 100 Mbytes.

2.2.1.7 **Ethernet Controller (DEQNA) Module** -- The dual-height DEQNA module is the interface between the print station and the Ethernet network. It allows data packets (ranging from 46 to 1500 bytes) to be transferred over the Ethernet network at a rate of 10 Mbytes/s.

2.2.1.8 **Monitors** -- Two types of system monitors are offered with MicroVAX workstations:

- VR260 Monochrome Monitor — The VR260 monitor measures 19 inches diagonally. It is capable of displaying 1024 (horizontal) X 864 (vertical) pixels on its screen. The monochrome video BNC input is connected to the BA23 enclosure by the BC18P-10 monochrome cable. The monitor is a 100-120/220-240 V, 50/60 Hz ac unit, and uses less than 65 W of power.

- VR290 Color Monitor — The VR290 monitor measures 19 inches diagonally. It is capable of displaying 1024 (horizontal) X 864 (vertical) pixels on its screen. The red, green, and blue BNC inputs are connected to the BA123 enclosure by the BC18Z-10 color cable. The monitor is a 100-120/220-240 V, 50/60 Hz ac unit, and uses less than 150 W of power.

2.2.1.9 **Monitor Cables** -- Two cables are used in configuring the two versions of the MicroVAX workstations:

- BA23 Enclosure -- The BC18P-10 monochrome video monitor cable, 3m (10 ft) long, connects the enclosure to the VR260 monochrome monitor.

- BA123 Enclosure -- The BC18Z-10 color video monitor cable, 3m (10 ft) long, connects the enclosure to the VR290 color monitor.

2.2.1.10 **LK201 Keyboard** -- The LK201 keyboard uses the full-duplex RS232 serial port on the VCB02 option. The keyboard's 105 keys are divided into four groupings: normal typing keys, editing keys, a numeric keypad, and special function keys. The keyboard also has a small speaker, four light-emitting diodes (LEDs), and associated electronic circuitry. A 1.9 m (6 ft) coiled cable connects the keyboard to a dedicated 4-pin MICRO-DIN connector on the monitor's rear panel. Refer to Appendix B for keyboard specifications.

2.2.1.11 **Mouse** -- The VSXXX-AA 3-button circular mouse is a hand-held, ergonomically designed input device connected to the monitor by a 1.5 m (5 ft) cable. The mouse's signals are passed to the RS232 serial port on the VCB02 base module for display on the monitor's screen. Refer to Appendix C for mouse specifications.

2.2.2 Options

2.2.2.1 **Additional Memory** -- You can add additional memory to the system by using a combination of the MS630 series of memory option modules. The MS630-AA, -BA, and -BB modules offer 1 Mbyte, 2 Mbytes, and 4 Mbytes of memory, respectively.

2.2.2.2 **Optional Tape Storage** -- The TK50 streaming tape drive allows use of tape cartridges, each with a capacity of 100 Mbytes, to be used in MicroVAX workstation systems. In the BA23 enclosure workstation, the TK50 tape drive can be substituted for the RX50 diskette drive, with Digital's software/diagnostics available to support the system. The BA123 enclosure workstation accepts the TK50 tape drive in addition to the RX50 diskette drive, allowing the system to use both methods of removable media.

2.2.2.3 **Digitizing Tablet** -- The VSXXX-AB digitizing tablet option consists of a 27.5 cm (11 in) square tablet, a 4-button puck, and a 2-button stylus. The tablet is connected by its 1.5 m (5 ft) power/signal cable to the monitor's RS232 serial port, replacing the mouse's cable connection. Refer to Appendix D for tablet specifications.

2.2.2.4 **Printers** -- Three printers are offered as options for MicroVAX workstations:

- LA50 -- 50 character/s impact printer
- LA210 -- 100 character/s impact printer
- LN03-AA -- 333 character/s (8 page/min) laser printer.

Refer to the VAXstation II/GPX Technical Manual, BA123 Enclosure (AZ-GNFAA-MN) for configuration/interfacing procedures for these printers.

2.2.2.5 **Communication Devices** --

• DZQ11 Asynchronous Multiplexer	This dual-height module has four asynchronous serial lines that conform to RS232 and RS423-A interface standards. The module can be used to support printer options and full-duplex modem operations.
• DMV11 Synchronous Controller	This quad-height module supports full- and half-duplex operations, point-to-point, multipoint communications, and DMA.

2.3 SYSTEM SPECIFICATIONS

2.3.1 BA23 Enclosure - Electrical

Input voltage	88 Vac - 128 Vac 176 Vac - 256 Vac
Frequency	47 Hz - 63 Hz
Power consumption	320 W (max.) 4.4 A @ 120 Vac 2.2 A @ 240 Vac

2.3.2 BA123 Enclosure - Electrical

Input voltage	87 Vac - 128 Vac 174 Vac - 256 Vac
Frequency	47 Hz - 63 Hz
Power consumption	640 W (max.) 8.8 A @ 120 Vac 4.2 A @ 240 Vac

2.3.3 Environmental

Temperature range	15°C - 32°C (59°F - 90°F)
Humidity	20% - 80% noncondensing with a maximum wet bulb of 25°C (77°F) and a minimum dewpoint of -26°C (6°F). NOTE The recommended operating temperature range is 18°C - 24°C (65°F - 75°F), 40% - 60% relative humidity.
Altitude	0 - 3000 m (0 - 10,000 ft)
EMI	Radiation limits are within FCC regulations for class "A" devices.

2.3.4 VCB02 Power Requirements

	Supply voltage:	Typical operating current:
Base Module	5 V (+/- 5%) 12 V (+/- 10%)	5.8 A 0.7 A
4-Plane Module	5 V (+/- 5%)	3.4 A

2.4 VCB02 VIDEO SUBSYSTEM INSTALLATION

The VCB02 video subsystem is shipped as an integral part of the assembled MicroVAX workstation system, and therefore, requires no customer installation. However, if the VCB02 subsystem is an add-on to an existing workstation, or requires servicing or replacement, you will find its removal/replacement procedures in the appropriate BA23 or BA123 Enclosure Maintenance Guide. The VCB02 connections, backplane slot insertion, and switch-pack on the VCB02 base module (M7169) settings are given here to supplement the procedures in the Maintenance Guides.

Figure 2-5 shows the cable connections for the VCB02-B video subsystem (4-planes).

Figure 2-6 shows the cable connections for the VCB02-C video subsystem (8-planes).

Figure 2-7 shows the connections of an 8-plane configured system installed in the backplane of a BA123 enclosure.

NOTE

Consult the removal/replacement procedures in the appropriate System Maintenance Guide before attempting to service or replace the VCB02 video subsystem.

Table 2-3 lists the VCB02 base module switch-pack settings. (See Figure 1-1.)

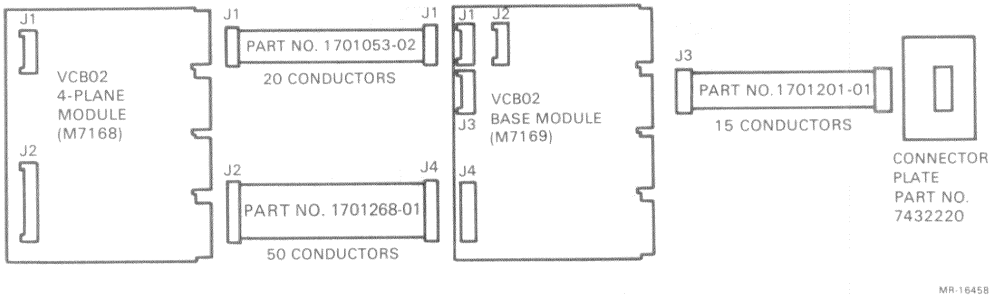


Figure 2-5 VCB02-B Cable Connections

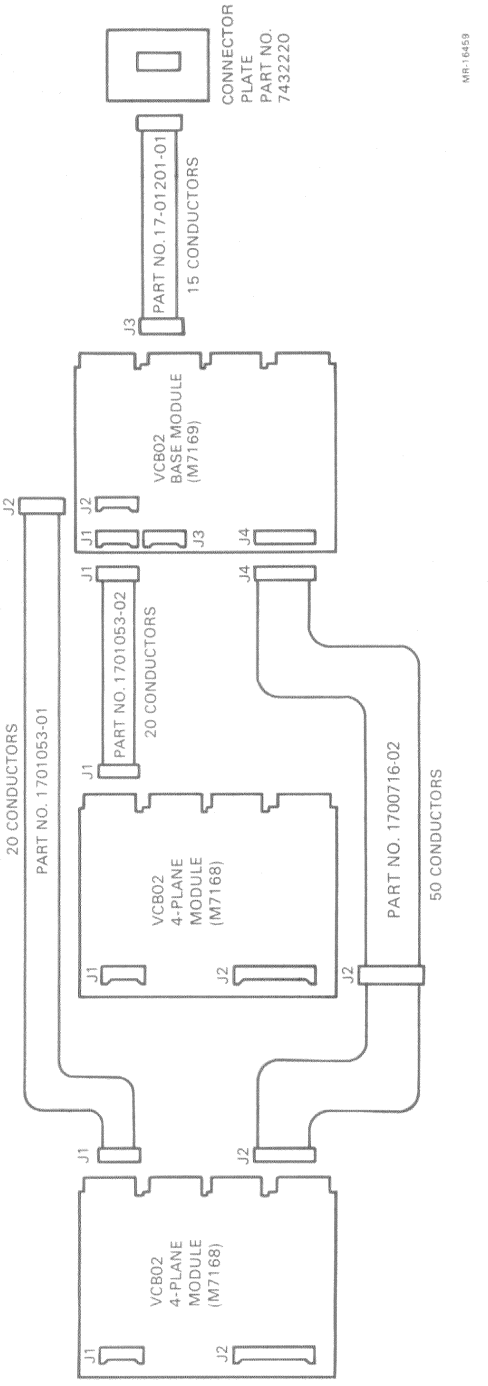


Figure 2-6 VCB02-C Cable Connections

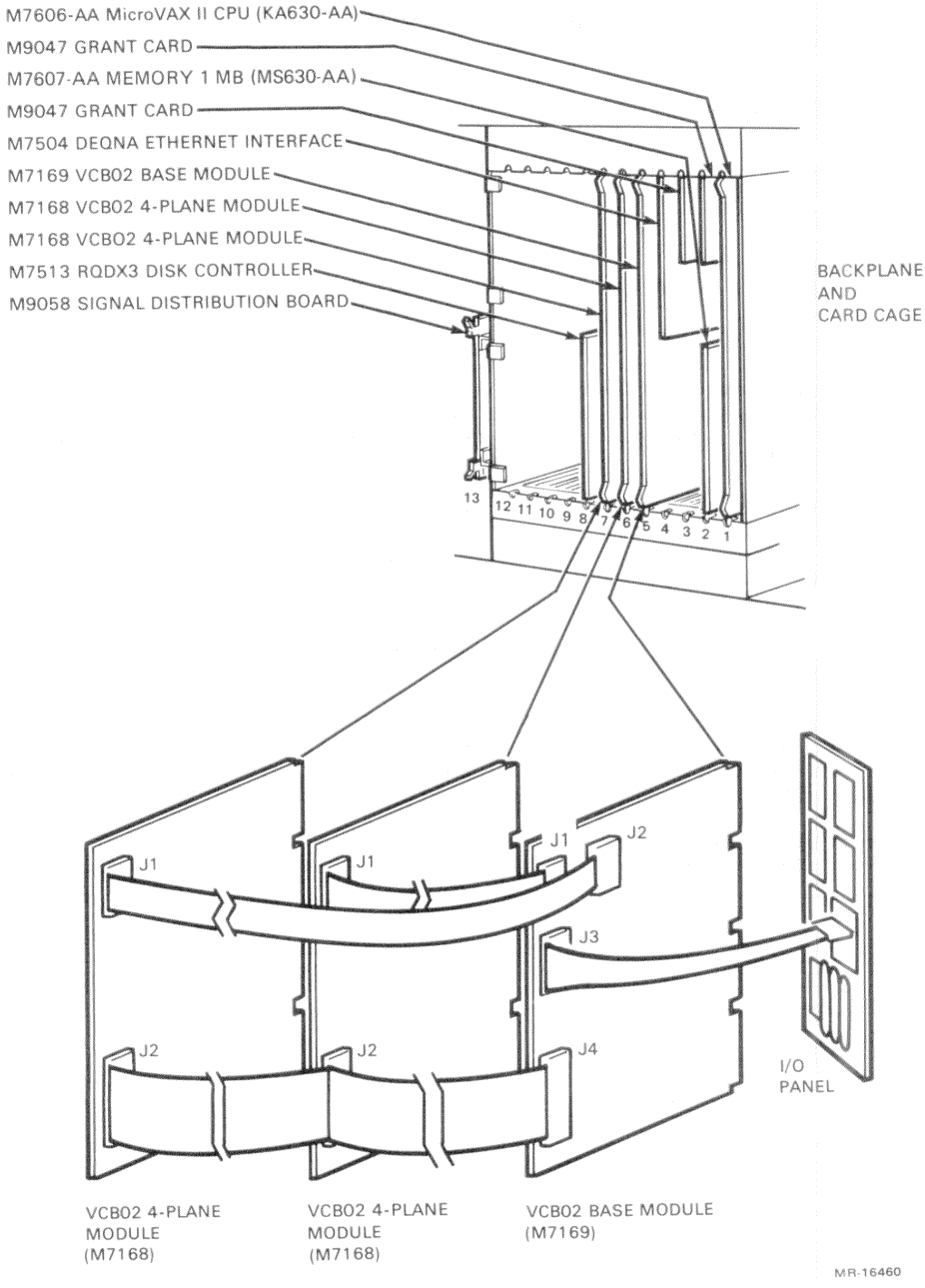


Figure 2-7 VCB02 Connections in a BA123 Enclosure

Table 2-3 Base Module Switch-Pack Settings

Switch	Function	Settings
S1	Selects address line 3	0 = On 1 = Off
S2	Selects address line 2	0 = On 1 = Off
S3	Selects address line 1	0 = On 1 = Off
S4	Unused	

NOTE

S1 through S3 selects address lines for the I/O page CSR.

2.4.1 Intermodule Connections

The VCB02 base module and the VCB02 4-plane module are connected together by cables. The interconnect signals are of two categories:

1. Interconnect signals common to both 4-plane modules in an 8-plane system

2. Interconnect signals unique to each of the 4-plane modules in an 8-plane system.

2.4.1.1 Common Intermodule Connections -- The interconnect signal subsets are listed in Table 2-4.

Electrical Connections

Two versions of a 50-signal ribbon cable with ground plane are used: one to support 4-plane systems, the other to support 8-plane systems with daisy-chaining capability between the three modules.

Mechanical Connections

A 50-pin Berg connecter is mounted on the edge of each quad-height module.

Table 2-4 Common Intermodule Connections

Memory Signals	
RAS	1 - Memory row address strobe
CAS	1 - Memory column address strobe
MEMOE	1 - Memory output enable
LATCH CNTROL	1 - External memory latch controller
WSTB	1 - Memory write from timing generator
MWE0:MWE3	4 - Memory write enables from address processor
MAD0:MAD7	8 - Memory addresses from address processor
MAD9	1 - Memory address from address processor
ADDCLK	1 - Address pipeline strobe
	---
	19 Total
System Clocks	
PHI1	1 - VCB02 video subsystem clock
PHI2	1 - VCB02 video subsystem clock
ALPHA	1 - Video output clock
SYNC	1 - System synchronization
SHIFT30	1 - Video synchronizer
CLK30	1 - Control synchronizer
	---
	6 Total
Video Processor Control	
128/-16	1 - Cycle control
RD/-WR	1 - Cycle control
LTCLK	1 - Video processor latch clock
ID0:ID7	8 - Instruction/data bus
SCROLL	1 - Scroll control
FORCE	1 - Scroll control
IDCTL	1 - Chip select control
WRSCR	1 - Scroll chip select write strobe
WRUCS	1 - Update chip select write strobe
	---
	16 Total

2.4.1.2 **Unique Intermodule Connections** -- Table 2-5 lists the signals between the VCB02 base module and the first VCB02 4-plane module. Table 2-6 lists the signals that pass between the base module and the second 4-plane module.

Electrical Connections
Two different lengths of a 20-signal ribbon cable with alternating signals and grounds are provided. The shorter cable connects to the low-order 4-plane module, which is configured adjacent to the base module, but on the opposite side of the Q22-Bus arbitrator (MicroVAX CPU module). The longer cable connects to the high-order 4-plane module, which is configured in the Q22-Bus backplane adjacent to the low-order 4-plane module). This scheme makes the two identical 4-plane modules appear electrically unique.

Mechanical Connections
The base module has one 15-conductor Berg connector on its upper edge, linked to the enclosure's connector plate. The 4-plane module has two 20-conductor Berg connectors, each aligned with connectors on the base module.

2.4.2 **I/O Connections**
An I/O connection on the base module routes the user I/O interface signals to a bulkhead on the enclosure. A cable connects to the base module and terminates at the enclosure bulkhead. Refer to the technical manual for the specific enclosure to determine the length of the I/O cable required. Table 2-7 lists the I/O signals.

2.4.2.1 **Mechanical Connections** -- The 15-pin Berg connector is located at the upper edge of the base module.

Table 2-5 **Intermodule Connections (First 4-Plane Module)**

Video Signals	
P1VID0, P1VID1	2
P2VID0, P2VID1	2
P3VID0, P3VID1	2
P4VID0, P4VID1	2
Status	
OPT1 PRES	1 - Option 1 present
SELECT	1 - Defines 4-plane module video processors to be low-order
	---
	10 Total

Table 2-6 Intermodule Connections (Second 4-Plane Module)

Video Signals	
P5VID0, P5VID1	2
P6VID0, P6VID1	2
P7VID0, P7VID1	2
P8VID0, P8VID1	2
Status	
OPT2 PRES	1 - Option 2 present
SELECT	1 - Defines 4-plane module video processors to be high-order
	---
	10 Total

Table 2-7 Input/Output Interconnect Signals

Keyboard Port 1	
Transmit	1
Receive	1
Power	1
Ground	1
Auxiliary Port 2	
Transmit	1
Receive	1
Power	1
Ground	1
-12 V	1
Video Signals	
Red video	1
Red ground	1
Green video	1
Green ground	1
Blue video	1
Blue ground	1
	---
	15 Total

3.1 SUBSYSTEM OVERVIEW

This chapter describes the hardware functions of the VCB02 video subsystem. The VCB02 video subsystem is a Q22-Bus-compatible, full-page bit-mapped, graphics video subsystem designed to support either the VR260 monochrome or VR290 color monitor. A DMA gate array chip (DC7035) provides the VCB02 subsystem with a specialized DMA engine that interfaces the Q22-Bus with the subsystem's video logic. The video logic is based on the address processor (DC323) and video processor (DC322) chip set. The address processor chip is responsible for all bitmap memory address calculations, CRT control, and video data path control. The video processor chip is responsible for all data manipulations to and from bitmap memory, and data output to the video refresh stream.

Each VCB02 video subsystem has one address processor chip and one DMA gate array. One video processor chip is used for each plane of bitmap memory in the VCB02 video subsystem. The subsystem is designed to support either 4 or 8 planes of bitmap memory. With a color monitor, a 4-plane system can display 16 colors (shades of gray for a black and white monochrome monitor) from a palette of 16.7 million colors. An 8-plane VCB02 video subsystem can display 256 colors (or shades of gray) from a palette of 16.7 million colors. Both the 4- and 8-plane versions of the VCB02 video subsystem provide a screen resolution of 1024 (horizontal) X 864 (vertical) pixels.

Each VCB02 video subsystem has a base module that provides the Q22-Bus interface, diagnostic ROM, two asynchronous communication ports, the address processor, color maps, and video timing and output logic. A 4-plane VCB02 video subsystem also has one 4-plane module that provides four video processor chips, four planes of bitmap memory, and chip select registers for the video processor chips. In addition to the base module, an 8-plane VCB02 subsystem has two 4-plane modules, for a total of eight planes of bitmap memory and eight video processor chips.

An 8-plane VCB02 video subsystem is a 3-module set, while a 4-plane VCB02 video subsystem is a 2-module set. The base module communicates with the 4-plane module(s) through cables. A 4-plane VCB02 video subsystem can be upgraded to an 8-plane system by adding a second 4-plane module and changing the module communication cables.

3.2 VCBO2 MODULES

3.2.1 Base Module

The base module (M7169) has the hardware that: processes address calculations, interfaces the DMA to the Q22-Bus, accesses display lists using the template RAM, controls cursor movement, accepts serial I/O bus input from attached keyboard/mouse/tablet devices, self-tests/diagnoses malfunctioning module components using LED indicators, emulates consoles, generates subsystem timing, and provides an 8-plane video output path for its color bitmaps.

In Chapter 1, the base module is shown in Figure 1-1 and Figure 1-2 is a block diagram of the module. Chapter 2 gives additional information on configuring the base module and 4-plane module in a workstation enclosure, a tabulation of each module's power requirements, the various intermodule connections, and instructions on module installation/servicing.

3.2.1.1 Instruction/Data Interconnect -- The I/D interconnect is a byte-wide synchronous interconnect between the address processor chip, video processor chip, and chip select registers. The I/D interconnect is synchronized to the system PHI1 and PHI2 clocks, which are further divided into PHI1A, PHI1B, PHI2A, and PHI2B. Instructions are transferred on the A clocks, data is transferred on the B clocks.

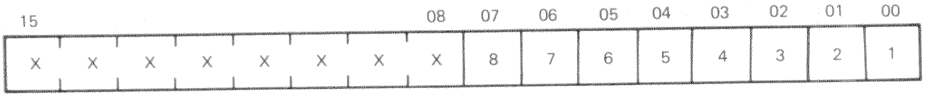
Chip Select Decoding

The base module generates either a scroll or update chip select strobe when a chip select instruction is decoded. The video processor chip selects are decoded from the I/D interconnect. To accomplish this, A and B clocks and decode logic provided on the base module, and either a scroll chip select write strobe (WRSCR) or an update chip select write strobe (WRUCS) are generated if a chip select instruction is decoded. These two signals are cabled to the 4-plane module through the module interconnect, and strobe the data into registers on the I/D interconnect. The external chip select load commands are shown in Figures 3-1 through 3-4.

X = DON'T CARE

MR-16981

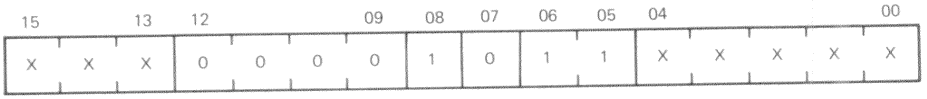
Figure 3-1 Scroll Chip Select Load Format - Command 140H



<8:1> = VIDEO PROCESSOR PLANE THAT WILL BE CHIP SELECTED BY THIS BIT (1 = ASSERTED).
X = DON'T CARE

MR-16962

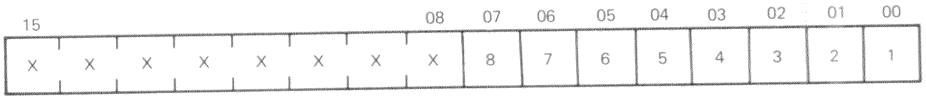
Figure 3-2 Scroll Chip Select Load Format - Data



X = DON'T CARE

MR-16963

Figure 3-3 Update Chip Select Load Format - Command 160H



<8:1> = VIDEO PROCESSOR PLANE THAT WILL BE CHIP SELECTED BY THIS BIT (1 = ASSERTED).
X = DON'T CARE

MR-16964

Figure 3-4 Update Chip Select Load Format - Data

3.2.1.2 **Subsystem Timing Generation** -- The timing needed to run the VCB02 video subsystem falls into three categories: system clocks, memory strobes, and video clocks. The pixel clock (69.1968 MHz) is used as the main divide down frequency. The inputs and outputs of the timing generator are listed below.

INPUTS (all sourced by the address processor)

- Sync Request
- Next Cycle
- Next Rd/-Wr
- Force

SYSTEM CLOCKS

- PHI1
- PHI2
- PHI3
- PHI4
- ALPHA
- 128/-16
- RD/-WR
- SYNC
- LTCLK

MEMORY STROBES

- RAS
- CAS
- WSTB
- MEMOE
- LATCH CNTRL
- ADDCLK

VIDEO

- TTL/ECL 15 H
- TTL/ECL 30 H
- TTL/ECL 30 L
- CONV H
- CONV L

Note that on power-up, memory and system clocking does not begin until POK has been received. This is so the system powers-up in a consistent manner.

3.2.1.3 **Video Output Logic** -- The devices that source data for the video output are the video processors (bitmap data) and the DMA gate array (cursor data). Video data from all sources is given to the video output logic in 4-bit nibbles. The four main functions of the video logic are hardware cursor support, mapping of the video data through a color look-up table (LUT), converting the digital video information into RS-170 voltage levels (not timing) for output to the monitor, and looping back the video output for diagnostic readback and self-test.

The following paragraphs describe the operation of the hardware cursor, the color map loading, and the video loopback register. The digital-to-analog conversion is not accessible to the programmer.

Hardware Cursor
The 2-plane hardware cursor is used to multiplex the video information to the pixel level in the following ways.

Cursor plane A data is 0	The cursor is transparent and bitmap data appears on the screen. Note this is the general case any time the cursor positioning logic is not outputting cursor data.
Cursor plane A data is 1	This can occur only when the positioning logic is outputting cursor data. In this case, cursor plane B data controls data to the screen in the following way.
Cursor plane B data is 1	Location [255] of the color map is accessed. This location is always considered foreground cursor color, but may be accessed by bitmap data as well.
Cursor plane B data is 0	Location [254] of the color map is accessed. This location is always considered background cursor color, but may be accessed by bitmap data as well.

Synchronization and Blanking
Video synchronization and blanking are programmable by the address processor on any rising PHI3 or PHI4 clock (57 ns interval nominal). They must then be synchronized to the video timing and shifted so their pipeline is equivalent to that of the video output path. Video blanking can also be enabled by setting a bit in the memory CSR.

Video Upgrade Path
When only one 4-plane module is installed, the upper four bits of the video data path are disabled.

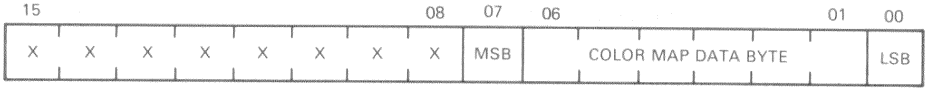


Figure 3-5 Color Map Data Format

Color Map Loading
Color map loading pertains only to the base module. In a 4-plane configuration, only the low 16 addresses need to be loaded in each map, except for locations [254] and [255] (used by the hardware cursor). In an 8-plane configuration, all 256 locations should be loaded. The red, green, and blue color maps are accessed independently (256 locations for each map). Figure 3-5 shows the format for the color map data.

Color maps should be loaded only during vertical blanking time so that no "glitches" are noticed on the screen. If this is not possible, a bit provided in the CSR will force the video to a blanking state while the maps are being loaded.

3.2.1.4 I/O Devices -- The VCB02 video subsystem has two serial EIA RS-232-compatible ports that can be addressed. One port is always reserved for the LK201 keyboard. The other port can be used for any other serial device, probably a mouse or a tablet. For programming information concerning these ports refer to Appendix B, C, and D, respectively.

NOTE
The serial ports are RS-232 voltage-compatible only.

LK201 Keyboard
The LK201 keyboard is the keyboard interface to the VCB02 video subsystem. A double-buffered UART in the Q22-Bus I/O space provides the send/receive port to the 4800 baud serial device. A switch closure on the keyboard causes an interrupt to the Q22-Bus to be serviced by reading the value of the UART receive register at the CSR address specified in the map. The LK201 keyboard is described in Appendix B.

Mouse
The 3-button circular mouse is supplied with the MicroVAX workstation. It is RS-232-compatible, using standard serial input device protocols. The mouse is described in Appendix C.

Tablet
The optional digitizing tablet responds at the address found in the CSR map. It is capable of interrupting the MicroVAX workstation for service of position changes and external events. The tablet is described in Appendix D.

Serial Device Protocols
The protocols for the serial devices are described in the Appendices: Appendix B - LK201 keyboard, Appendix C - 3-button circular mouse, and Appendix D - optional digitizing tablet.

Monitors
The monitors specified for the VCB02 video subsystem are the VR260 for monochrome applications and the VR290 for color applications.

3.2.1.5 **System Support** -- The system support provided by the VCB02 video subsystem includes console emulation, multiple VCB02 terminal support, and full diagnostic support.

VCB02 Address Map
Each VCB02 video subsystem has one register in the Q22-Bus I/O page. See Figure 3-6. Switches on the base module are used to select one of eight possible addresses for this register, as listed below. When read, this register provides an identification code for VCB02. Data written to this register selects the VCB02 memory base address (on a 64-Kbyte boundary). Figure 3-7 shows the Q22-Bus memory space. Register mapping is shown in Figure 3-8. All registers are accessed on word boundaries.

VCB02 Address Mapping Scheme:

QBUS I/O PAGE BASE + 1F00H*
QBUS I/O PAGE BASE + 1F02H
QBUS I/O PAGE BASE + 1F04H
QBUS I/O PAGE BASE + 1F06H
QBUS I/O PAGE BASE + 1F08H
QBUS I/O PAGE BASE + 1F0AH
QBUS I/O PAGE BASE + 1F0CH
QBUS I/O PAGE BASE + 1F0EH

Figure 3-6 Q22-Bus I/O Page

* Alternate Console: A VCB02 module assigned this address is the system alternate console device.

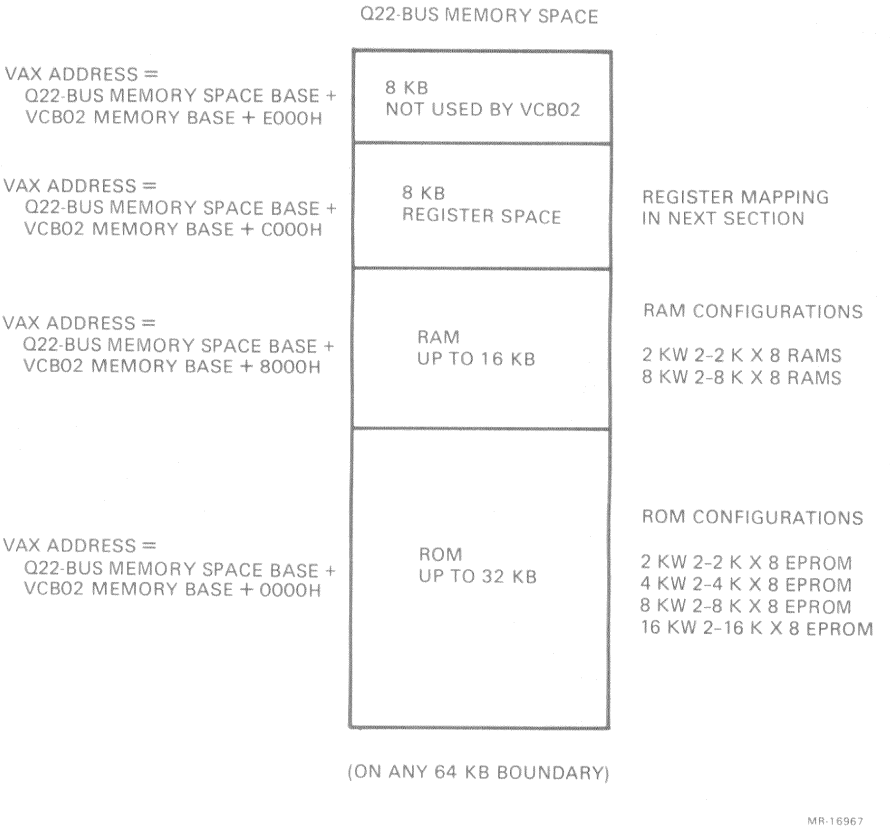


Figure 3-7 Q22-Bus Memory Space

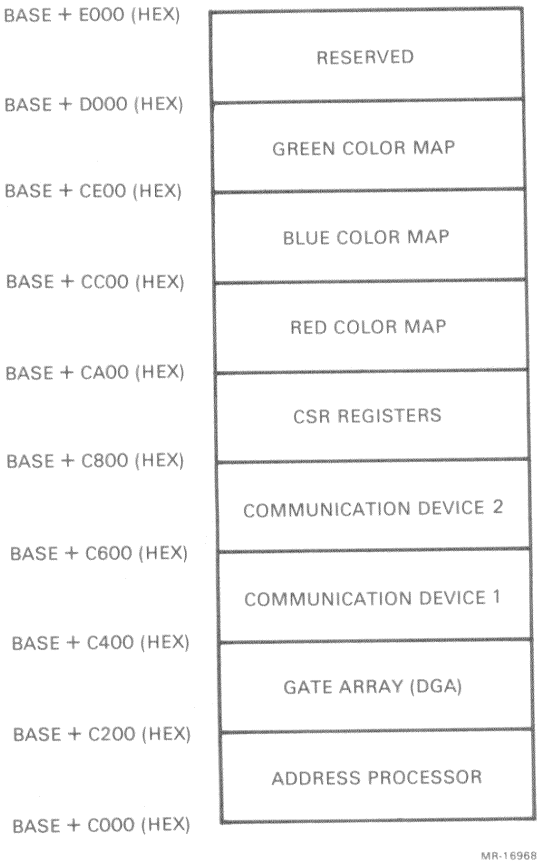


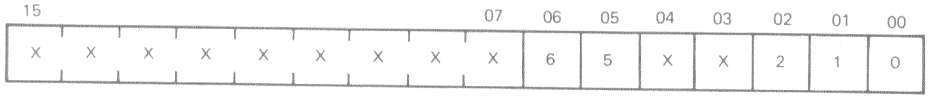
Figure 3-8 Register Mapping

3.2.1.6 **Control and Status Registers (CSRs)** -- The VCB02 video subsystem has two control and status registers. One is located in the Q22-Bus I/O page, the other in Q22-Bus memory space. Both registers are read/write.

I/O Page CSR
Figure 3-9 shows the format for the I/O page CSR read VCB02 identification code.

The value written into the memory base register selects the starting address (base address) of a 56-Kbyte block of addresses that will be decoded by the VCB02 DMA gate array. The base address is always on a 64-Kbyte boundary in the Q22-Bus memory address space, as selected by bits 00 through 05 of the memory base register. Bits 05 through 00 are compared with signal pins A(21:16) H during the address portion of Q22-Bus memory cycles. Figure 3-10 shows the write memory base register format.

Memory CSR
Figure 3-11 shows the memory CSR read video readback register format. Figure 3-12 shows the memory CSR write format for the control write register.



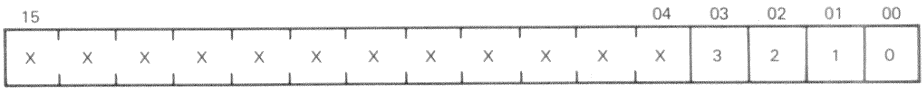
BIT 6 : OPTION 2 PRESENT (0 = ASSERTED)
BIT 5 : OPTION 1 PRESENT (0 = ASSERTED)
BIT 2 : FULL-PAGE SYSTEM CONFIGURATION (1 = ASSERTED)
BIT 1 : HALF-PAGE SYSTEM CONFIGURATION (1 = ASSERTED)
BIT 0 : QUARTER-PAGE SYSTEM CONFIGURATION (1 = ASSERTED)
X : NOT DEFINED

MR-16969

Figure 3-9 I/O Page CSR Read VCB02 ID Code Format

MR-16970

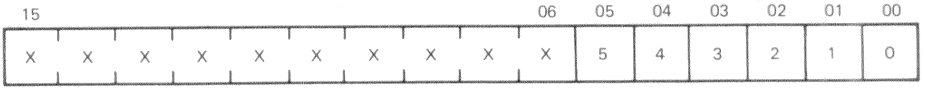
Figure 3-10 Write Memory Base Register Format



BIT 3 : SYNC DETECT (0 = ASSERTED)
BIT 2 : BLUE > RED (0 = ASSERTED)
BIT 1 : GREEN > BLUE (0 = ASSERTED)
BIT 0 : RED > GREEN (0 = ASSERTED)

MR-16971

Figure 3-11 Memory CSR Read Video Readback Register Format



BIT 5 : CSR BLANK (0 = ASSERTED)
BIT 4 : CSR VIDEO READBACK (1 = ASSERTED)
BIT 3 : CSR SYNC ENABLE (1 = ASSERTED)
BIT 2 : LED FOR HIGH-ORDER MEMORY UPGRADE FAILURE (0 = ASSERTED)
BIT 1 : LED FOR LOW-ORDER MEMORY UPGRADE FAILURE (0 = ASSERTED)
BIT 0 : LED FOR BASE SUBSYSTEM FAILURE (0 = ASSERTED)

MR-16972

Figure 3-12 Memory CSR Write Format for the Control Write Register

3.2.1.7 **Console Emulation** -- The VCB02 video subsystem supports console emulation through interactions of the MicroVAX CPU, the MicroVAX boot ROM, and the VCB02 ROM, which are mapped in the VCB02 address space.

The VCB02 video subsystem is the alternate console device, and supports Micro/ODT by providing an ASCII interface to the system when in console mode. The VCB02 ROM is word-addressable to allow direct MicroVAX CPU reads. For a more detailed explanation of the VCB02 diagnostic requirements, refer to Chapter 5.

3.2.1.8 Diagnostic Support --

Diagnostic ROM
The VCB02 ROM provides code for both diagnostics and console emulation. For more information on the VCB02 ROM, refer to Chapter 5.

Bitmap and Template Memory Access
The memory on the VCB02 4-plane module is divided into three segments:

1. Bitmap memory
2. Template RAM
3. Video Look-Up Table (LUT).

All bitmap memory can be accessed through processor-to-bitmap and bitmap-to-processor transfers, which are executed through the address processor and video processor chips. Template RAM memory is directly accessible through Q22-Bus transfers. The LUT, which can be written directly, is read back indirectly through the memory CSR as described below.

Video Readback
The video readback path is accessed through the memory CSR. Three readable bits correspond to comparisons of the outputs of the video digital-to-analog converters. The three outputs are compared and incremented through the different voltage levels, so there is a high degree of confidence in the integrity of the video output data path. The resolution of this comparison is 20 mV.

A fourth CSR bit shows the comparison of the synchronization pulse voltage level on the green gun to a known reference voltage. In the event of an error, this test helps diagnostics determine if the error is on the VCB02 module or a monitor/cabling problem. The reference voltage is set at 0.1 V, and the comparison resolution is 20 mV.

3.2.2 Four-Plane Module

The 4-plane module (M7168) has the hardware that provides the 4-plane, full-page memory to the base module. Refer to Chapter 1 for a list of the 4-plane module's components and Figures 1-1 and 1-3. For configuration, power requirements, intermodule connections, and installation information, refer to Chapter 2.

3.2.2.1 Video Memory Structure -- For each "on-screen" page of memory within a plane, the VCB02 video subsystem provides an equal amount of "off-screen" memory. "On-screen" refers to the memory that is used for screen refresh. The VCB02 video subsystem presents the programmer a block of memory that is 1024 (horizontal) X 2048 (vertical) pixels.

3.3 ADDRESS AND VIDEO PROCESSOR OVERVIEW

This section presents an overview of the features and operation of the address processor and video processor chips as implemented on the VCB02 video subsystem.

The address processor chip (DC323) controls all video operations in the VCB02 video subsystem. The DMA gate array on the VCB02 base module controls the register interface of the address processor using a 16-bit private data bus, a 6-bit address bus, and control signals. Internal to the address processor is a register structure through which all commands to, and information to and from, the video chips must pass. Communication between the address processor and video processor chips occurs on a byte-wide instruction/data (I/D) bus. The address processor chip registers are classified into four categories: control, scroll, rasterop parameter, and configuration registers. Once these registers are properly loaded, the address processor chip uses control signals and the I/D bus to synchronize all internal address calculations, CRT control, and external memory cycles with the video processor chip's data manipulations.

The video processor chip (DC322) is the bitmap data path for all VCB02 video subsystem transactions. The video processor chip is controlled by signals from the address processor chip and by 20 programmable registers that can be loaded from the I/D bus. These registers are classified into three categories: scroll, rasterop, and configuration registers. Once properly loaded, the video processor is capable of: sourcing and receiving data on the I/D bus, logical operations on bitmap data, barrel shifting, and sourcing data to the output video bus.

The I/D bus is a byte-wide synchronous interconnect between the address processor, video processor and chip select registers. The I/D bus is synchronized to the system PHI1 and PHI2 clocks, which are further divided into PHI1A, PHI1B, PHI2A, and PHI2B. Instructions are transferred on the A clocks, and data is transferred on the B clocks.

3.3.1 Hardware Support

The address processor chip is responsible for functions common to all bitmap planes, such as: all rasterop computations, bitmap address generation, clipping, screen refresh, scroll control, and monitor synchronization generation. The video processor chip (DC322) provides the data path and control data FIFOs for refresh and scrolling, a barrel shifter for bit alignment, a logic unit with data and mask registers for memory modification, Z-axis address logic, and a control store RAM to define video processor chip operations during rasterops. Figure 3-13 is a block diagram of the VCB02 video subsystem.

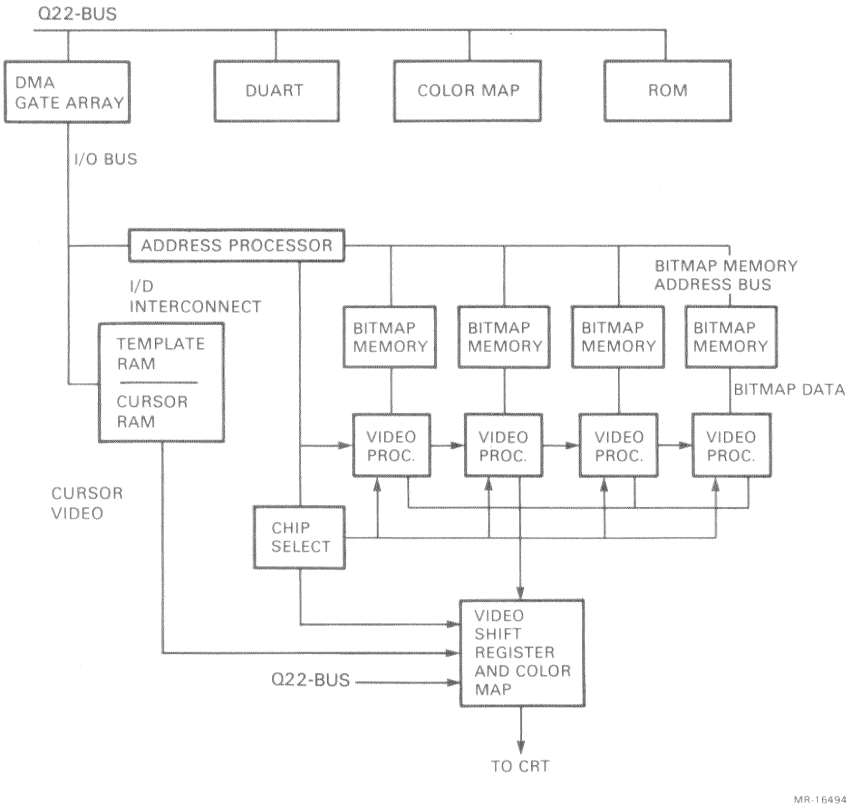


Figure 3-13 VCB02 Video Subsystem, Block Diagram

3.3.1.1 **Buses** -- The address processor chip communicates on three buses:

1. The private data bus is controlled by six control lines and six address lines, and has 16 bidirectional data lines that transfer data to/from internal registers in the address processor chip. These registers may be addressed in two ways:

   a. The six address lines are provided for direct access by the MicroVAX CPU through its normal memory or I/O addressing modes.

   b. An address counter in the address processor chip allows the DMA logic in the DMA gate array to pass data and control words to sequential address processor chip registers via a single register (address).

2. The bitmap address bus provides 11 bits each of multiplexed row and column address to the bitmap RAM. The VCB02 uses 8 row and 9 column addresses to access 128 K words (2 M bits) of bitmap memory. The bus also provides all addresses for screen refresh, scroll writeback, and bitmap read and update.

3. The instruction/data (I/D) bus (interconnect) controls the video processor chips and provides for all data flow between the video processor chips and the MicroVAX CPU (via the address processor chip). Each cycle of the 8-bit I/D bus has four states that provide a 16-bit instruction from the address processor chip to the video processor chips, and 16 bits of data from one I/D bus device to the others. These cycles are used to configure the video processor chip registers, regulate data transfers during rasterops, and load the chip select registers on the I/D interconnect. The chip select registers external to the video processor chips and controlled by the I/D bus are used to enable one or more video processor chips to perform a transfer.

The video processor chips use three buses:

1. The I/D bus (interconnect) described above.

2. The 16-bit memory data bus is used in the transfer of all data to or from the memories during screen refresh, scroll writeback, and memory update.

3. The video output bus supplies the bit stream for each plane as successive 4-bit nibbles. These nibbles are externally processed by the color map, video shift registers, and D/As to provide the video signal to the monitor.

3.3.1.2 **Bitmap Memory** -- Each plane of bitmap memory is configured as eight 64 K X 4 dynamic RAMs. A total of 2 M bits is provided for each plane, with 864 K bits used for screen refresh (1024 horizontal X 864 vertical pixels). The remainder is used for off-screen storage of fonts and bitmap data.

3.3.1.3 **Timing** -- Hardware timing refers to a nominal pixel period of 15 ns (14.45 ns actual). All bus timing, with the exception of the video output bus, and is based on multiples of a 30 ns clock. The 4-bit-wide video bus operates at 60 ns.

The register interface on the address processor chip is asynchronous to the rest of the video hardware because its timing is controlled by the DMA gate array.

There are two kinds of bitmap memory cycles. A major cycle (960 ns nominal) is used to read or write 8 words (128 bits) to or from the video processor chip for screen refresh or scrolling. Any unused major cycle is subdivided into 2 minor cycles (also called update cycles, 480 ns nominal) during any of which either a read or a read-modify-write of a word may occur to accomplish bitmap update. Refresh of the dynamic bitmap memory is accomplished every 593 μs (except during vertical retrace, which is less than 677 μs) by the screen refresh read. (Worst case refresh time = 1.263 ms.)

Figure 3-14 shows an example of a scan divided into 16 major cycles. Seven are used for screen refresh, and seven may be used for scroll writeback, if any of their words are contained in a scrolling region. The remaining two, plus any unused scroll cycles, are divided into minor cycles for updates or NOPs. The address bus provides one row and eight column addresses during a major cycle, and one row and one column address during a minor cycle. The data bus to the video processor chips operates concurrently to transfer 8 words (128 bits) during a major cycle, and to read and return a modified word (16 bits) during a minor cycle.

In the VCB02 subsystem, each scan is programmed to last 10 major cycles (8 are used for screen refresh). The positioning and width of horizontal blanking and synchronization are programmable with respect to memory cycles; they are set as required by the VR260 and VR290 monitors. The number of scans per frame is programmed to 901 (including vertical retrace time); the number of displayable scans is 864. The position and width of vertical blanking and synchronization are programmable; they are set as required by the VR260 and VR290 monitors.

The I/D bus operates continually at the minor cycle rate (480 ns nominal), whether major or minor cycles are in progress. It also operates in synchronization with the memory cycles, during which two instruction bytes and two data bytes are transferred.

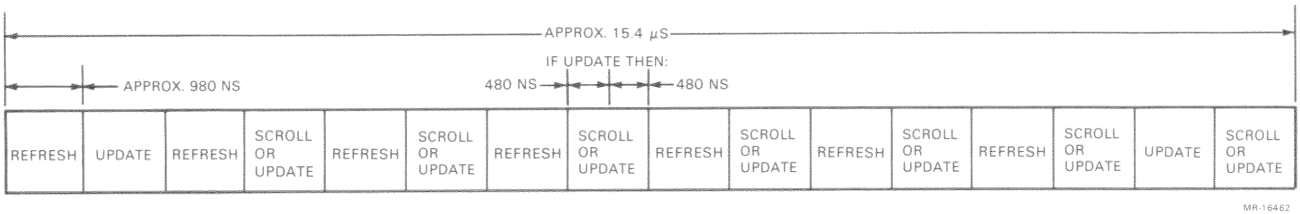


Figure 3-14 Memory Cycles during One Scan

The computation cycle of the address processor chip is two major cycles (1920 ns, nominal). During this time, all source and destination address computations occur for one step of a rasterop. This step may be either one pixel or, if the rasterop is progressing parallel to the X-axis and is not scaling the source, a whole bitmap bus word. Four additional compute cycles are required to initialize a rasterop. Computation can occur with or without available update cycles, and there is a small amount of buffering of results (6 addresses of 3 sources and 3 destinations; 6 destinations, etc.). This allows optimal use of update cycles. No-source and 1-source operations are compute-bound when no scrolling is in progress; any other operations may be memory cycle-bound. In general, the time to load data and start the next rasterop is added to the rasterop execution time (though the rasterop origins may be loaded during rasterop execution) so that sequential text characters may be written to the bitmap without wasting time between characters.

3.3.2 Memory Organization

The address processor chip provides one large rectangular bitmap memory of fixed dimensions for the storage of data. Any address to this memory consists of an X and a Y component. All bit map space must be allocated two-dimensionally.

The bitmap memory is not directly accessible by the MicroVAX CPU, and the address processor chip does not perform rasterop manipulations in the MicroVAX CPU's memory. However, requests may be made for data exchange between processor memory and the bitmap. The DMA gate array can be used to accelerate data to/from the MicroVAX CPU's memory.

The memory is divided into an on-screen and an off-screen portion. The on-screen memory is a rectangle starting at the address [0,0] and ending at (but not including) the address [X limit = 1024, Y limit = 864]. (These limits are programmable in the address processor chip.) This area includes all memory that is read (but not necessarily displayed) by the screen refresh process.

3.3.3 Subsystem Control

The hardware is controlled by the MicroVAX CPU through the loading of command and data registers in the address processor chip. Registers control the configuration of the address processor and video processor chips for screen format, video synchronization, scrolling, and rasterops.

The MicroVAX CPU can load any of the writable registers directly. It can also read several status bits to verify the address processor chip's readiness to accept (or provide) data. These status bits include: rasterop initialization complete, rasterop computation complete, address buffer empty, pause complete (programmable frame synchronization), scroll service required, vertical blanking (fixed frame synchronizations), I/D bus data ready, and various clipping occurrences. Any of these status bits may be enabled to assert an interrupt request pin on the address processor chip (connected to the interrupt logic in the DMA gate array).

An interface to a DMA controller (in the DMA gate array) allows decoupling of the MicroVAX CPU from address processor chip execution. A request pin may be programmed to assert on any of the same conditions that are available as flags or interrupts to the MicroVAX CPU. This pin requests the next data word from the DMA gate array. When loading data and command registers, the DMA gate array writes to a special address in the address processor chip that is associated with an internal address counter. If the MSB of the data word is clear, the counter addresses the address processor chip register to which data is to be transferred. The counter is incremented after each word is transferred. If the MSB of the data word is set, the address counter is loaded with the low six bits of the data word.

Only the I/D bus data registers contain 16 significant bits of data to allow loading of arbitrary data to these addresses. The MSB does not cause the address counter to be loaded if it is pointing to either of the I/D data registers. (The addresses of these registers are preceded by reserved register addresses so that no register load will cause the address counter to be left pointing inadvertently to an I/D data register.) The register addresses are assigned so that common repetitive functions only need to access one group of consecutive registers, in addition to the command register. A group of special instructions have also been implemented in the DMA gate array to further assist in the execution of commands.

3.3.4 Viewport Support

To assist the implementation of multiple viewports on the screen, the VCB02 chips provide clipping to, and scrolling or dragging of, rectangular regions. (The term "region" refers to the implementation of viewports in the hardware.) The top and bottom of a region may be set to any pixel, the left and right edges must lie on a multiple of four pixels from the left side of the screen. Figure 3-15 shows examples of possible region configurations. There is one clipping (update) region and one scrolling region at any instant, but the two are independent of each other. This is so writing and scrolling may be active in different regions at the same time. Any part of the screen not currently contained in either the clipping or scrolling region cannot be modified. It will retain any data placed in it previously. The whole update region may be either scrolling or not, but a region should not be updated if only part of it is scrolling. (For instance, region 11 should not be updated if region 5 is being scrolled.)

When a region with writing disabled is updated, clipping still allows the image outside the region to be computed. The address processor chip provides status to the MicroVAX CPU to indicate whether or not any rasterop was completely or partially clipped while writing (to aid in selecting algorithm). These clipping status bits are accumulated as rasterops progress, and may be read or cleared at any time. However, the results are predictable only if the bits are read or cleared between rasterops.

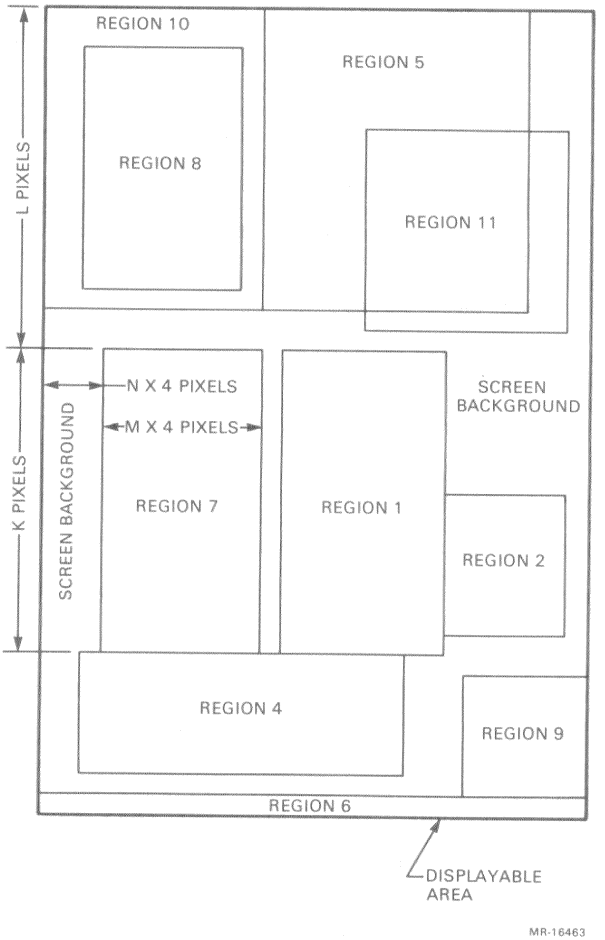


Figure 3-15 Region Configuration Example

The regions are allowed to be independent as much as possible. The scrolling resource must be allocated to only one region per frame time (and an additional frame time is normally required to change regions). All regions must share one color map and agree on resolution mode settings. Unless large amounts of extra bitmap are provided, regions share the off-screen areas used for symbol and data storage. Obviously, overlapping regions cannot be independent of each other.

3.3.4.1 **Scrolling** -- Scrolling movement can be up, down, left, or right (but not diagonally) at a rate of 0 to 15 pixels/frame (or faster for vertical scrolls). The part of the region vacated by the moving image is simultaneously filled with any solid color. Scrolling uses part of the memory time that would otherwise be available for writing; but, even when the whole screen is in motion, 30 to 40 percent of the nonscrolling update time is still available for source/destination operations, depending on screen format. (Fifty to eighty percent of nonscrolling time is available for destination-only or source-only operations because these operations are primarily compute-bound when not scrolling.)

NOTE

Actually, the distance of an upward scroll is limited only by the number of scan times allowed for vertical retrace. This time is used to write the fill color back into the vacated memory if the region extends to the bottom of the screen. The distance of a downward scroll is subject to the memory limitation described below.

Scrolling is accomplished by moving all the scrolling data from its old position in memory to a new position. The normal screen refresh process reads and displays the existing memory data, and additional memory write cycles are used where necessary to return scrolling data to new locations. This is the only technique that allows a region of any size, including the whole screen, to be scrolled in one frame time; but there are two unpleasant side effects. First, only one region can be in motion during a frame time, because two regions might need to write portions of the same word to more than one memory location. Second, scrolling down cannot be accomplished in the obvious way because this would require writing data into memory locations not yet read by the refresh/scrolling process. An attempt to do this would destroy screen data.

Instead, scrolling down moves all the data outside the scrolling region up, and then offsets the memory address at which screen refresh starts reading the bitmap. This creates the illusion that the scroll region moved down. (Proper synchronization is maintained so that the data outside the region does not move.) This creates four bad effects:

1. Some scans of memory must be reserved (and not displayed) to prevent data at the bottom of the screen from being overwritten by data from the top. The number of scans required is equal to the maximum distance, in pixels, to be scrolled down in one frame time.

2. The writing time is consumed as if the whole screen were scrolling up. Scroll writing must occur throughout the frame so that all data can be moved up in planes where scrolling is disabled.

3. Diagonal scrolling is not possible because different data must be moved for the downward and sideway motion of a region.

4. Management of the Y offset adds extra work for the MicroVAX CPU.

To synchronize scrolling commands with screen refresh, most of the scroll-related registers in both the address processor and video processor chips are double-buffered. In the address processor chip, the index registers are buffered only to the extent required for the continuation of scrolling in one region. (See the NOTE below.) The pending half (top level) of any buffered register is addressable by the MicroVAX CPU and is loaded at any time during a frame. Between frames, the address processor chip transfers the data from all pending registers to the active registers, and then sets the flag that requests more scroll data.

A separate I/D command and data path (including chip select control for the video processor chips) is provided for scrolling to prevent interference between the loading of video processor chip registers for scrolling and for updating. This allows a scroll service routine to act without regard to the state of update service. The scroll and update services must avoid using each other's registers.

NOTE

To conserve die area in the address processor chip, scroll registers are buffered only to the extent required to sustain scrolling or dragging of a single region. Extra buffering of the index registers would be needed to scroll different regions in successive frames. This function is not considered important because its only use would be in trying to simulate simultaneous scrolling of two separate regions. However, since a smooth effect cannot be achieved with this technique, it should not matter that the abruptness of jump-scrolling of the two regions is increased. If the MicroVAX CPU were able to service an interrupt request to load 4 registers in 200 to 500 μs, it would be possible to scroll different regions in successive frames without the extra buffering of the index registers. (Normally, the MicroVAX CPU takes an entire frame time to load a buffered register.)

3.3.4.2 **Dragging** -- The scroll function can be used to drag a region smoothly to a different place on the screen. For each increment of motion, the scroll boundary registers are set for the current region location, but with the region size increased in the direction of motion by the distance to be moved. This allows the whole region to be moved, without the normal truncation, and leaves the fill color in the "wake" of the region. Because the region boundaries can be defined only on 4-pixel increments in the horizontal direction, horizontal dragging must stop on 4-pixel boundaries, however; it is possible to move the region at any valid scroll speed during the drag.

If desired, the area to be covered by the moving region may be saved prior to each movement, and the area vacated may be restored after each movement. A region cannot be updated while it is scrolling because there is no way to synchronously change the clipping boundaries. The clipping boundaries are not double-buffered as the scroll boundaries are.

3.3.4.3 **Clearing a Region** -- The address processor chip has a provision to allow a region to be cleared to the fill color in one frame time, using the bulk scrolling hardware. This action is linked to one frame time in the same way that scrolling is. While small regions (less than one-sixth of the screen) can be cleared more quickly with a rasterop, the erase mechanism does the same very cleanly because it is synchronized to occur between two display frames.

3.3.4.4 **Drawing in the Scrolling Region** -- The blank space created by scrolling needs to be filled with new data. Drawing as scrolling continues is desirable because the smooth effect of the scroll is not disturbed. The indexing mechanism of the address processor chip allows data to be drawn to the correct screen location without regard to how scrolling is moving the screen. Typically, the update process draws the display list (or selected portions of it) repeatedly to keep filling the scroll region as more blank space is created.

The clipping region stands still during scrolling to prevent data from being written outside the region. However, this means the clipping region cannot be used as a "drawing function" to control the shape or extent of objects being drawn into a moving region. This is so because the clipping region does not move with the objects as they are drawn. Also, clipping cannot be used to prevent multiple writes of data when the same display list segments are repeated during scrolling. This would require the clipping region to follow a particular set of blank pixels as they move on the screen. Multiple writes will create problems for display lists that contain segments drawn in complement mode or drawn with the painter's algorithm (in which successive elements replace previous elements).

To enable synchronizing the update process with scroll commands (that is, to fill the blank areas created by scrolling), the pause register and the interrupt or request enable bits can be set to interrupt the MicroVAX CPU or request further data from the DMA gate array (during DMA) when a specific point on the screen has been passed by screen refresh. Normally, the pause is set by the scroll service routine to allow the update process to continue after the top or bottom of the scroll region is displayed. This allows a full frame time to fill the top or bottom edge of a region that is scrolling up or down before the blank space would become visible.

Indexing

To allow updating from the MicroVAX CPU or DMA gate array into a region that is scrolling (or has scrolled) without requiring the MicroVAX CPU to change the coordinate values in its display list, index registers in the address processor chip are added to the first source and destination coordinates to match the new position of data that has been moved in the memory by scrolling.

There are six index registers: old X and Y, new X and Y, and pending X and Y. The old values are the indexes that apply to data that has not yet moved during the current frame; the new values are the indexes that apply to data that has already been moved; the pending values become the new values at the start of the next frame. Between frames, the address processor chip transfers the value stored in the new register to the old register, and from the pending register to the new register, thus starting the new frame with the correct indexes. The index values apply to the region that is being updated. Ordinarily, if the update region is also being scrolled, the new and old indexes will differ by the scroll constant (X or Y) for the current frame (loaded in the previous frame). If the update region is not being scrolled, the new and old values are the same.

The index values must be changed by the update process between updates to different regions. Also, if the update region is being scrolled, the scroll process must update the index values between frames so that updates will stay locked to the scroll movement when they extend beyond the end of one frame time. Because both the update and scroll processes modify the index registers, and the update process must use the most recent values provided by the scroll process, a very tight interlock must be maintained between these two processes.

This interlock is maintained by the MicroVAX CPU, if the DMA-driven update process has not changed the update region to one that is scrolling. This is so because a frame may end between the time the MicroVAX CPU index values in the data list for the DMA controller and the time these values are loaded into the address processor chip. It is best not to send region changes via DMA.

When the MicroVAX CPU changes the update region, the following interlock with the scroll process is recommended.

1. The MicroVAX CPU always executes the scroll process and the part of the update process that changes regions (no DMA controller).

2. The update process must be interruptable by the scroll process when the update process is changing regions (note exception below), unless the latency caused by disabling interrupts during index loading is acceptable to the system.

3. During a region change, the update process loads a memory location that tells the scroll process which region is now being updated.

4. The update process copies the correct index values for that region from a table maintained by the scroll process to the address processor chip registers. The index registers must be loaded in the order: pending, new, old. If the instruction(s) used to move each index value to the address processor chip is (are) interruptable, it is necessary to disable interrupts (at least for the priority of the scroll service interrupt) while each index value is copied.

5. If a new frame is started, the scroll process is awakened by the scroll service interrupt from the address processor chip. The scroll process must run to completion before returning to the update process. The scroll process updates the index table entries for the region being scrolled, so that new values are available to the update process.

6. The scroll process should load all six index registers in the address processor chip with the new values for the region being updated. If the update region is not the same as the scroll region, the registers need not be loaded. If the scroll process is certain that the index values in the address processor chip corresponded to the region being scrolled before the end of the frame that caused the scroll interrupt (perhaps there is only one region on the screen, or only the scroll region has been updated since the last scroll), then only the pending index registers need to be loaded.

7. Update processing may now continue.

3.3.5 Multiplane Support

The use of multiple video processor chips (data path chips) allows the simultaneous manipulation of data in many planes of memory. The video processor chips are controlled by, and exchange data on, the I/D bus. The registers for data transfers and the logic functions to be performed may be independently programmed for each video processor chip in the system.

3.3.5.1 Z-Axis Addressing -- Z-axis operations allow the exchange of data between the address processor and video processor chips on the I/D bus by using 16 bits of a data word to transfer 1 bit to/from each of 16 planes. This can be used for both the exchange of data between the MicroVAX CPU and the bitmap, and the loading of appropriate video processor chip data registers.

Each video processor chip can be programmed with a 6-bit plane address. The lower 4 bits specify the bit within an I/D bus data word to which a video processor chip will respond during a Z-axis operation; the upper 2 bits specify the Z-block (should be set to 00 for the VCB02). No two planes may have the same address. (This must be ensured by the MicroVAX CPU.)

During processor-to-bitmap (PTB) or bitmap-to-processor (BTP) data exchanges, data in a normal rectangle from the bitmap is transferred from/to the MicroVAX CPU memory by the DMA logic in the DMA gate array or the MicroVAX CPU itself. If Z-mode transfers are requested, each word transferred will be the color of one pixel. Since a VCB02 video subsystem can have a maximum of eight planes, the DMA gate array can be used to pack/unpack data as it is transferred to/from the MicroVAX CPU memory. Compressing the data reduces the memory required to store an image, and reduces the bus bandwidth required to transfer the data.

A Z-axis I/D bus cycle can be commanded directly, with an I/D command, to load either the source, foreground, background or scroll fill data registers in the video processor chips. The source register is one input to the logic units in the video processor chip. The foreground and background registers are selected on a bit-for-bit basis by the output of the logic unit. Loading them with a Z-axis command can program the video processor chips for the colors with which subsequent elements will be drawn. The fill registers in the video processor chips define the color that will be written into the new areas of scrolled regions.

3.3.6 Basic Address Calculation and Data Path Hardware

The address processor and video processor chips contain bitmap manipulation hardware to execute the commands provided by the MicroVAX CPU. All manipulations are based on rasterops, which can select pixels from an area of the screen (a source) and combine them with pixels selected from another area (a destination). The address processor chip handles all word and bit addressing, clipping, and control. The video processor chips provide bit alignment, data exchange, and logical combinations.

The three rasterop modes are: normal, linear pattern, and fill. The contents of the address processor and video processor chip rasterop parameter registers are not modified or destroyed by any operation of the system. Only those registers whose values are to be changed need to be loaded between operations. Of course, the contents of data registers may be changed by rasterops.

3.3.6.1 Address Processor Chip - Addressing -- Every rasterop has a destination and may have zero, one, or two sources. A zero source implies a constant source. The destination generator selects pixels to be modified and the source generators select pixels to be combined with the destination. No curve algorithms are implemented because all the useful ones require multiplications, and there is no "best" algorithm for all applications. Curves may be handled by passing a series of straight vectors to the address processor chip.

The destination is a parallelogram and may be of different size or orientation than the first source rectangle, with the data scaled or rotated to fit the destination. Rotation and scaling are used more for transforming picture elements, such as characters and patterns, than for transforming whole pictures. However, scaling can be used to "zoom" an image to an area of the screen by a scaled copy from another area, possibly off screen. The parallelogram (as opposed to simple rectangle) form of the destination is largely a fallout of its implementation as two independent vectors, but is useful for forming italic characters and performing operations on screens with nonsquare pixels. Rotation and scaling may not be suitable during complement mode drawing operations.

The sequence of source/destination operations is controlled by the address processor chip command register. Three bits select one of the eight possible combinations of sources and a destination. (The possible sources are: none, first, second, and both. The destination is either on or off. Note that not all combinations are useful, however.) The source and destination addresses are always computed, but these three bits determine whether they are used. The effect of the source data is determined by the programming of the video processor chips.

The origins for the first source and the destination areas may be offset by the addition of an X and a Y index. Index mode may be invoked for either the first source or the destination independently, but the X and Y index values are the same for both. This mode allows the command data to remain unmodified: when a region has been scrolled or is being scrolled (only the index values need to be updated); and/or the origin of a region may be (0,0) to the update process, regardless of its actual location in the bitmap.

All address computations in the address processor chip use 14-bit, 2's complement numbers for each X and Y. Beyond the 14-bit limit, calculations wrap from plus to minus in the normal fashion. The usable range of values for any of the DX or DY registers is +/- 12 bits (+/- 4095 pixels) because these values are multiplied by 2 in some internal calculations. However they are still specified as 14-bit, 2's complement numbers. Thirteen bits of the X address (including specification of the pixel within a word) and thirteen bits of the Y address are available at the output of the chip.

A 6-deep FIFO is provided to store the computed addresses before they are used in memory cycles. This allows better use of available memory cycles during scrolling. If only one destination or source is required, six operations can be stored (and two more can usually be computed while these are being used, so that at least eight operations can be performed during each scan of scrolling, if the screen format allows); and, if source/destination operations are required, three (or two for double source) operations can be stored. Figure 3-16 shows the address processor chip data path.

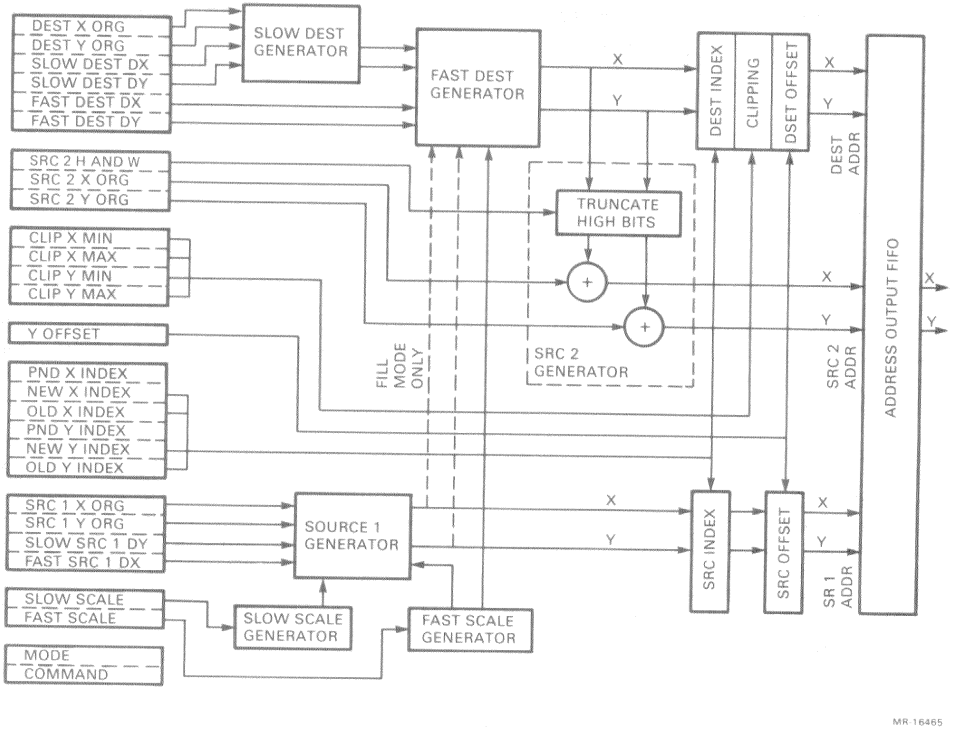


Figure 3-16 Address Processor Chip Data Path

Destination - Rotation

The destination is a parallelogram defined by two vectors and an origin. See Figure 3-17. Each vector is defined by two signed integers representing a horizontal delta (DX) and a vertical delta (DY), and both vectors start at the same origin. The area is scanned by addressing pixels along the path of one vector (the fast vector), starting from the destination origin (after indexing, if selected), until it is exhausted. Then, using the next pixel on the path of the other vector (the slow vector) as a new origin for another fast vector with the same DX and DY, fast vectors are scanned until all the pixels on the slow vector have been addressed. Thus, parallelograms of any size or rotation can be scanned. Parallelograms can be used to scan rectangles on screens having nonsquare pixels.

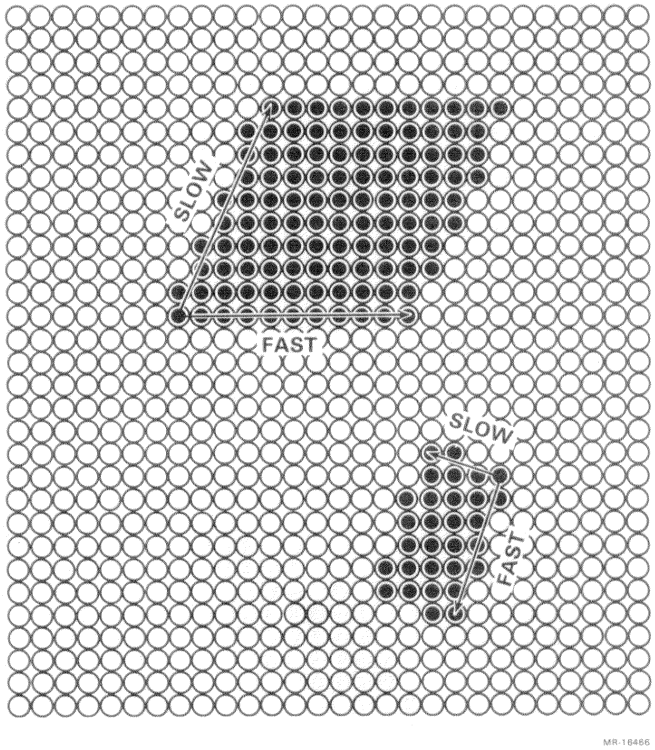


Figure 3-17 Example of Destination Rasters

The pixels on the path of the vectors are computed by Bresenham's Algorithm. Of the two components (DX and DY) of a fast or slow vector, one is generally longer than the other and is called the "major axis"; the shorter component is called the "minor axis". According to Bresenham's Algorithm, the number of pixels selected along the path of a vector is equal to the length of the major axis. The origin and all points up to, but not including, the last point on any vector (the point at "origin+delta") are selected. The last point is not included (the length of a vector would be one pixel too long) and an adjoining rasterop should not select the last point (its first point) a second time. Otherwise, a pixel would be restored to its original state during complement writing.

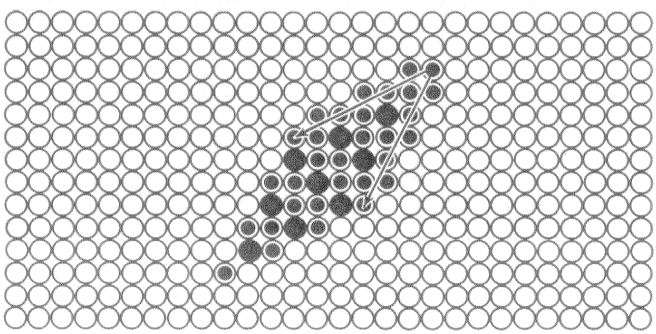
Because the destination can be indexed and is corrected for the effects of Y offset used in scrolling down, it can address any portion of the bitmap memory.

Holes and Duplications

Using this scanning technique for the destination, some fast and slow vector combinations will cause some pixels within the parallelogram to be addressed more than once. Some vector combinations will cause some pixels to be missed, and other combinations will cause both effects, or neither effect, within the parallelogram. The following combinations of fast and slow vectors will create duplications, as shown in Figure 3-18.

1. If both vectors point in one of the eight cardinal directions (0, 45, 90, etc.), no duplications will occur unless both vectors are in the same or opposite direction.

2. If one of the vectors points in one of the eight cardinal directions, duplications will occur only if the other vector is in one of the two adjacent octants, or one of the two octants adjacent to the opposite cardinal direction.



MR-16467

Figure 3-18 Example of Fast and Slow Vectors that Duplicate Pixels

3. If neither vector points in a cardinal direction, duplications will occur unless the vectors are in "perpendicular" octants, that is, octants separated by an intervening octant.

Note that no rectangular area of any orientation (on a screen with square pixels) will cause duplications. Duplicated pixels cause problems for a complement operation, but one would expect some overlapping of data when an image is compressed by any of the above conditions, or is scaled down. These complement problems can be minimized by doing transformations in off-screen memory without complement mode, and then copying the results to the visible screen without a transformation. The video processor chips provide two sets of control registers to support repeated 2-step operations.

Figure 3-19 shows combinations of fast and slow vectors that will create holes.

1. If either of the vectors is parallel to the X- or Y-axis, no holes can occur.

2. If neither of the vectors is parallel to the X- or Y-axis, holes will be generated only if one vector is in a quadrant adjacent to the quadrant containing the other vector.

Holes will be created in images, regardless of drawing mode, so these holes are filled with a modification to the scanning algorithm. Hole filling can be disabled for drawing single-width vectors.

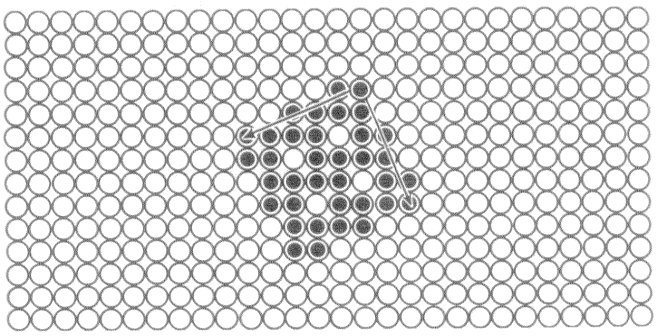


Figure 3-19 Example of Fast and Slow Vectors that Leave Holes

Bresenham Error Computation

Bresenham's Algorithm is implemented in the address processor chip by unconditionally incrementing the position register for the major axis, and incrementing the register for the minor axis according to an "error" computation.

1. At the start of a vector computation, the error register is initialized to minus the magnitude of the major axis. The X and Y position registers are set to the origin of the vector (the origin of the rasterop for the slow vector and the current slow vector position for the fast vector).

2. For each iteration of the algorithm, a point is plotted at the current position.

3. The position along the major axis is incremented.

4. Twice the magnitude of the minor axis is added to the error register.

5. If the error register becomes non-negative, the position along the minor axis is incremented, and twice the magnitude of the major axis is subtracted from the error register.

In addition, the initial value of the error register is offset by adding a programmable value to the built-in initialization described above.

For normal and linear pattern rasterops, the contents of the error 1 register are added to the destination slow vector error register, and the contents of the error 2 register are added to the destination fast vector error register. These error registers are normally set to zero. In fill mode, the error 1 register is used in the same way (controlling the A edge of the filled area), and the error 2 register is added to the source 1 error register (controlling the B edge). Control of the error initialization allows the center line of the drawn vector to be shifted by a half pixel to either side of its normal position, which is exactly through the centers of the pixels that are its end points. The slope of the line is unaffected by the error register initialization.

First Source - Scaling

The first source is an unrotated rectangle defined by a fast vector (a signed width; always parallel to the X-axis), a slow vector (a signed height; always parallel to the Y-axis), and an origin. The magnitude of the first source fast vector is the result of the length (major axis) of the destination fast vector multiplied by a fast scale factor. The sign of the source fast vector is determined by the sign of the source DX register, a 14-bit register with the sign in the MSB. (The remainder of the 2's complement source DX is not significant in normal mode; it is used for linear pattern and fill modes.) Similarly, the magnitude

of the source slow vector is the result of the length of the destination slow vector multiplied by the slow scale factor; the sign is determined by the sign of the source DY register.

A fast or slow scale factor is a number less than 1 with an indication of up or down scaling; it is stored in a 14-bit register. The MSB (bit 13) indicates up- or down-scaling; the magnitude is specified in the remaining 13 bits, with the binary point preceding bit 12. On each computation of a vector (fast or slow), the address processor chip increments either the source or destination vector, adds the scale factor plus 0.0000000000001B to an accumulator (initialized to zero) and, if bit 12 of the accumulator overflows, increments the other vector. If up-scaling is specified, the source is incremented only when the accumulator bit 12 overflows and the destination is incremented. The reverse is true if down-scaling is specified.

One pixel from the small vector may map to more than one pixel in the large vector. This results in nonuniform sampling of the small vector during the scaling process.

The scaling of fast and slow vectors is independent. Down-scaling causes pixels in the destination to be written a multiple of times. If complement writing is used, 2-step operations using off-screen memory may be used to obtain correct results, as suggested above for duplicate destination pixels.

To ensure expected operation of scaling for rational scale factors, it is necessary to round up any scale factor that has a remainder beyond the 13th bit from the binary point. This ensures the first pixel is not read one time too many. However, the address processor chip adds 0.0000000000001B to each scale factor. Therefore, a scale factor with a remainder should just be truncated, and any scale factor without a remainder (such as 1.00) should have 0.0000000000001B subtracted from it.

Due to the 13 bits of precision available, inaccuracies can exist in large scale factors. This problem becomes significant when the number of significant bits in a large scale factor is reduced to near or below the binary order of magnitude of the size of the source or destination in a scaled operation.

The first source rectangle is scanned by starting at the origin (after indexing, if selected) and selecting pixels along the fast vector (X direction) until the destination fast vector is exhausted and the fast scale accumulator allows incrementing to the next destination pixel. (The latter condition ensures correct down-scaling for the last destination pixel.) Then, if the slow scale accumulator allows, using the next pixel on the slow vector (Y direction) as a new origin for another fast vector of the same length, additional fast vectors are scanned until the destination slow vector and slow scale accumulator are exhausted. Because the first source can be indexed and is corrected for the effects of Y offset used in down-scrolling, it can address any portion of the bitmap memory.

The data read from the pixels addressed by the first source can be combined logically with data addressed by the second source, and by the destination to form new destination pixels. The first source is most commonly used to move objects, such as characters, in the bitmap or to move areas of the screen.

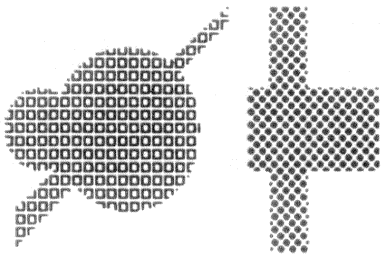
Fast Mode
When scaling or rotation of the fast vector is involved, computation and memory cycles occur one pixel at a time. However, if the fast vector of the destination is parallel to the X-axis, the fast scale factor is 1, and the source and destination fast vectors are in the same direction (have the same sign). (The last condition is not necessary if the first source is disabled.) The hardware will operate on words (16 bits) to increase the speed of writing. This "fast mode" is invoked automatically under the appropriate conditions.

Fast mode is disabled if linear pattern mode (fill-in Y-mode, or Z-mode), which is used for processor/bitmap transfers, is enabled.

Second Source - Tiles
The second source addresses are derived from the destination addresses by adding the destination address, before indexing, to an address (offset) stored in the source 2 origin registers. This allows a second image to be combined with the first source and destination (or the destination only). No scaling or rotation is provided between the second source and the destination, though, of course, the first source may still have these properties.

This use of the second source can be thought of as copying a source area, with clipping to the destination parallelogram. The second source is neither indexed nor adjusted for the effects of Y offset (used for down-scrolling), and is, therefore, only useful for objects and patterns stored in off-screen memory for the definition of off-screen memory). This limitation is not serious because most second source data will be from off-screen fonts.

A special form of the second source will read tile patterns if the source 2 height and width register is appropriately programmed. When tiling, the second source is used to texture other objects with a 2-dimensional rectangular pattern. An important requirement of such patterns is that they must mesh continuously when two independent patterned objects touch anywhere on the screen. The desired effect, shown in Figure 3-20, is as if copies of the pattern rectangles were used to tile the whole screen. When a patterned object is written, these tiles are exposed to view throughout the interior of the object.



MR-16469

Figure 3-20 Tiled Areas

The address processor chip achieves this effect by restricting the height and width of a pattern rectangle to integer powers of 2, adding only the low-order bits of the destination X and Y addresses (before indexing) to the X and Y components of the source 2 origin to create the source 2 X and Y addresses. When tiling is enabled, the effect of the source 2 origin is more like that of a pointer to the origin of the tile than that of an offset. If the destination is indexed, the phase of the tiles will drift across the bitmap with changes in index value. This is a desired effect because it allows patterned objects to mesh properly, even if they are written at different times to a region that is scrolling.

The height and width of a tile are independent of each other and range from 2^2 to 2^9. The tile width may never be set to less than the bus width (16 bits for the VCB02), but 4- and 8-bit-wide tiles are possible in 16-bit mode by repeating the tile elements within the tile cell. Tile origins must be on word boundaries in the memory.

As with the second source, no scaling or rotating of tile patterns is provided because the address processor chip is not capable of the calculations required to determine the starting phase of the resulting pattern. If scaling or non-power-of-two tile sizes are required, the pattern can be written more slowly by calculating the phase in the MicroVAX CPU, and using linear patterns to write one scan of the patterned area at a time.

3.3.6.2 **Video Processor Chip - Data Manipulation** -- The video processor chips perform the actual manipulation of the memory data accessed by the address processor chip. This consists of aligning the source data bits with the destination location, communicating this data to its own logic unit and/or to those of other video processor chips, and modifying the destination data with the logic unit. Figure 3-21 shows the rasterop data path for video processor chip functions.

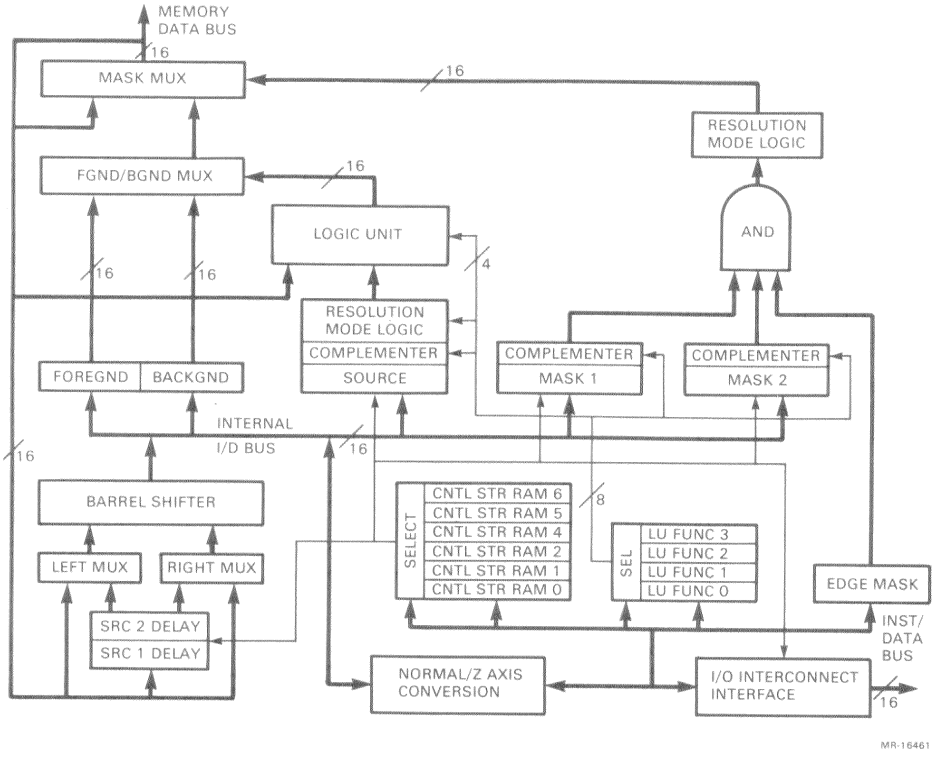


Figure 3-21 Video Processor Rasterop, Block Diagram

Barrel Shifter - Bit Alignment

During a bitmap memory read cycle, the barrel shifter accepts a 32-bit input and provides any contiguous 16-bit segment of this as an output. Both words of the input can be the current word being read, or one word can be the current word and the other a previous word held in one of two delay registers. Fast mode operations require the latter ability to avoid excess reads to the source. One delay register is always associated with the first source, the other always with the second source. The output of the barrel shifter can be directed to the source or mask registers on the video processor chips, and/or to the I/D bus for broadcast to the other video processor chips or the address processor chip. Only one video processor chip can drive the I/D interconnect during any source operation.

The address processor chip provides the shift constants. For the first source, the shift constant is the difference between the source and destination pixel addresses (low 4, 3, or 2 bits of the X addresses) after indexing. If selected for the second source, the shift constant is simply the difference between the low bits of the source 2 X origin and the X index, when selected for the destination.

Logic Unit and Source/Mask Registers

During a bitmap memory read-modify-write cycle, the logic unit combines the contents of the source register (loaded with a constant during a preceding source read cycle) with the current contents of the destination memory location. Any of the 16 possible bit-wise logical combinations may be requested. The output of the logic unit selects, on a pixel-for-pixel basis, a foreground or background color to form new destination data. The foreground and background registers are usually loaded explicitly by the MicroVAX CPU, a Z-axis load, or an ordinary load. They are also loaded implicitly during a Z-mode processor-to-bitmap transfer.

When using a full-resolution binary pattern (such as a text character or vector path) to write into a low-resolution plane, adjacent pattern bits must be spread so that all the bits of a low-resolution pixel are either on or off. This is accomplished by resolution mode logic between the source register and the logic unit. This logic ORs the adjacent 2 or 4 bits in 2- and 4-bit modes, respectively. If any of the bits in the group is a 1, both or all four bits of the group will be 1s at the input of the logic unit. A programmable complementer between the source register and the resolution mode logic spreads the 1s or the 0s in the source. If an operation requires moving data that has already been expanded to low-resolution colors, the resolution mode logic can be disabled.

Two mask registers are provided to control which bits in a destination word are to be modified. These registers are loaded in the way the source register is loaded. Loading the mask 1 register also loads the mask 2 register with the same data, so that only one mask is effective. Mask 2 may be loaded independently when two masks are required. Either a mask value or its complement may be used.

The address processor chip provides a special mask (called the edge mask), via the I/D bus during read-modify-write cycles, to define the bit or segment of a word involved in the current rasterop cycle. All these mask words are ANDed together, so that only bits selected by all three masks will be modified. Bits in the range of a rasterop but not addressed by the address processor chip cannot be modified. When no mask is being used, mask 1 should be loaded with all ones and both complementers should be turned off to prevent masking of destination bits. If only mask 1 is used, the complementers for both mask 1 and mask 2 should be set to the same state, or else all bits will be masked.

Control Store RAM and Function Registers

To provide flexibility in the sequencing and operations of rasterops, each video processor chip has two memories that define its functions. The control store RAM provides 6 locations, each of which define the data transfer functions to be performed on each memory cycle of a rasterop:

1. Where to send the read data from the barrel shifter (source, masks, I/D interconnect)

2. Where to send incoming I/D data (source, masks)

3. Whether to capture the read data in one of the delay registers that precedes the barrel shifter.

There are two banks of control store RAM, with three locations each. The first location of each bank controls data transfers during a source 1 read, and specifies the control of the delay register for the first source. The second location controls transfers during a source 2 read, and specifies the control of the other delay register for the second source. The third location controls the destination read-modify-write.

The two banks allow easy switching between two functions without the need of reprogramming the video processor chips (as when a function that must be performed with two rasterop steps is executed repeatedly). The address processor chip selects which bank to use (as specified by the MicroVAX CPU), and also defines the sequence of source and destination accesses. (The sequence is: source 1, source 2, and destination, if enabled.)

The control store RAM is required because the video processor chips often perform different operations at the same time. For example, during text operations, one video processor chip will be reading the font and transmitting it on the I/D bus, while the other chips will be receiving the font from the I/D interconnect.

The second memory bank of the control store RAM controls the logic unit, source resolution logic, and mask registers. This RAM has four locations of eight bits. Four bits select a logic function for the logic unit; two bits select normal or complement for each of the two masks; the last two bits select normal or complement for the source register, and enable the source resolution mode logic. This RAM is also addressed by the address processor chip during rasterops, as specified by the MicroVAX CPU. These registers permit convenient transitions between tasks requiring different logic selections, such as writing normal and reverse text. Four registers allow two orthogonal functions to be programmed.

3.3.6.3 **Processor-to-Bitmap and Bitmap-to-Processor Transfers --** Processor-to-bitmap and bitmap-to-processor transfers are similar to rasterops, except that either the source or destination is the MicroVAX CPU memory. Processor-to-bitmap transfers are called PTBs, bitmap-to-processor transfers are BTPs, and transfers of either type are called PBTs. All data for a PBT is transferred through the 6-deep I/D data FIFO. The two modes for PBTs are X-mode and Z-mode. The second source, scaling, and rotation are not available.

In an X-mode PBT, data from one plane is transferred to/from the MicroVAX CPU memory, with 16 adjacent bits in X occupying one word, followed by additional X words, and then by additional scans.

In a Z-mode PBT, all the bits from one pixel location are transferred to/from the MicroVAX CPU memory word, followed by additional pixel data along the fast vector, and then by additional scans. Since a VCB02 video subsystem can have a maximum of 8 planes, the DMA gate array can be used to pack/unpack data as it is transferred to/from the MicroVAX CPU memory. Compressing the data reduces the memory required to store an image, and reduces the bus bandwidth required to transfer the data.

A PBT transfers a continuous stream of data words to/from the MicroVAX CPU memory, either through the MicroVAX CPU itself or through the DMA logic in the DMA gate array. The data words are written/read by the hardware along the path of the destination/source fast vector, until exhausted, and then along additional fast vectors, starting at points along the path of the slow vector. Depending on the sophistication of the transferring device, these words can be all sequential in the MicroVAX CPU memory, or each fast vector's data can be indexed to different points in the MicroVAX CPU memory. The latter action could be used to access a rectangle of data within a larger rectangle already formatted in MicroVAX CPU memory. However, such indexing is an externally supplied function, not a part of the address processor chip.

In general, it is possible to execute PBTs while scrolling. X-mode transfers are restricted from use in a region that is horizontally scrolling because the number of words transferred depends on the alignment of the rectangle being transferred to the memory words, which changes during horizontal scrolling. Z-mode transfers have no restrictions during scrolling.

PBTs in a scrolling region (or in any on-screen location if some part of the screen is scrolling down) must operate continuously, adhering to a maximum time between words transferred. If a transfer becomes delayed, address calculations for scrolling become invalid, and data will transfer to/from the wrong screen location. There is no such problem for PBTs in off-screen memory.

It is recommended that the MicroVAX CPU compute exactly the number of words that will be transferred by the hardware with a PBT command. For a Z-mode transfer, this is simply the width multiplied by the height (in pixels) of the area being transferred. For an X-mode transfer, the width calculation depends on the width (in pixels), the bus width mode (16 bits for the VCB02), and the indexed X origin address. If exact calculations are not possible, too much data is accommodated by harmless additional reads or writes to the FIFO. However, if too little data is transferred, the address processor chip will not be satisfied, requiring a cancel command (or more data) to continue operation. The ability to cancel a BTP operation is useful when the MicroVAX CPU scans the bitmap data for a particular set of values, stopping when such is found. This is useful for a polygon flood operation.

3.3.6.4 **Performance** -- When no scrolling is in progress, destination-only and single-source rasterops are compute-bound in the address processor chip; therefore, they operate at approximately 500 K cycles/s (compute cycle rate). Rasterops with two sources are memory cycle-bound and operate at approximately 400 K cycles/s (depending on screen format). Cycles operate on either single pixels or words, depending on whether the conditions for fast mode exist. The address processor chip requires four compute cycles to initialize each rasterop.

If the whole screen is scrolling up or sideways, or any area is down-scrolling, all rasterops are memory cycle-bound. Destination-only rasterops operate at 300 K to 400 K cycles/s. Single-source rasterops operate at approximately 200 K cycles/s, 2-source rasterops operate at approximately 150 K cycles/s. Partially scrolling screens (except down-scrolling) have update speeds between the nonscrolling and full-scrolling speeds.

One additional performance limitation applies to updates to a region that is in motion (scrolling), or to any region when down-scrolling is in progress. Normally, when the screen refresh and scrolling process passes an area on the screen where updates are taking place (or source data is being read), the update process is temporarily halted while the refresh process passes to ensure that only old indexes are used below the refresh process, and only new indexes are used above.

This delay is necessary because the address processor cannot predict precisely when a computed address will be output to the memory. Normally, this causes an unnoticeable decrease in update speed. However, if the update process is proceeding down the screen faster than the refresh process (this can happen if fewer than eight memory cycles, depending on screen format, are required to write or read all the data at each Y address; e.g. vertical vectors), the update process will not be able to pass the refresh process. This limits progress to that of the refresh process.

To eliminate this effect, it is recommended that a tall, thin rasterop be written from bottom to top, rather than from top to bottom, in a region whose height is a significant portion of the screen height. This limitation does not apply when the new and old indexes are equal (or if they are disabled for both source and destination), and the Y offset is unchanged from that of the previous frame.

3.3.7 Application to Text

Normal rasterops are used to write characters. The address processor chip contains no additional hardware to assist in the display of text. This section describes common text procedures.

3.3.7.1 Font Storage and Access -- Fonts are stored in an undisplayed portion of the bitmap. Ordinarily, a font needs to be stored in only one plane. It is transmitted from that plane to itself and others when a character is written. This results in the writing of monochromatic characters (all of one foreground color and one background color) in the bitmap, as is normally desired. If multicolored or gray-scale (for anti-aliasing) characters are desired, the fonts can be stored across many planes.

Characters may be stored anywhere in the bitmap (though normally in off-screen memory). However, a character occupies a 2-dimensional space in the memory, described by height, width, and an X and Y origin. Thus, font storage in the memory must be allocated in a 2-dimensional fashion. Normally, the MicroVAX CPU keeps a table where it can look up the origins and widths of the characters in each font. While character origins may be on any pixel, faster operation results if characters are left-justified in bitmap memory word cells (16-bit words). In this way only left-shifting is required, resulting in fewer source reads.

3.3.7.2 **Normal Text** -- A display of unrotated, unscaled, fixed-pitch text might use the following sample settings in the address processor and video processor chips. Only those settings actually changing between characters need to be updated following initialization.

1. The update region is initialized with clipping limits, indexes, and resolution mode.

2. The video processor chips' control store RAMs are set for the proper data transfers during the rasterop. Normally, the settings would enable broadcast of the character data from the plane containing the font to all other planes.

3. The video processor chip logic functions, foreground, background, and (possibly) source and mask registers are set for some of the desired character attributes.

4. The address processor chip mode register is initialized as: rasterop = normal, hole fill = off, source index = off, destination index = on, and pen down.

5. The source 1 origin (X and Y) is set to the start of the first character to be transferred, and the DX and DY sign bits (the rest are not significant) are set for the direction of source scanning.

6. The destination origin (X and Y) is set to the desired screen location for the first character. The destination fast DX is set to the width and fast direction of the character cells, and the fast DY is set to 0. The destination slow DX is set to 0, and the slow DY is set to the height and slow direction of the character cells.

7. The fast and slow scale registers are set for scale factors of 1.

The following operations may now be performed to write each character in a string.

1. The MicroVAX CPU loads the command register in the address processor chip to start a source 1/destination rasterop, with the desired logic unit function register and CSR bank selected.

2. The address processor chip now copies the source to the destination using fast mode (because there is no rotation or scaling).

3. After waiting for the address processor chip to finish the initialization phase of the rasterop, yet while the character is still being written, the MicroVAX CPU reloads the source and destination origins and the command (if the same as the previous command) so that the next character starts immediately upon completion of the first.

4. This process is repeated until the text is exhausted or an attribute change is required.

The speed of text writing in this mode varies with the character size; but, for 9-wide X 15-high cells, about 20 K chars/s can be written, if fully supported by the MicroVAX CPU. The speed ranges from 15 K to 25 K chars/s for other useful character sizes. Because a full-page screen contains about 6000 characters, 300 ms are required to fill the screen with characters. Adding about 100 ms to clear the screen, but considering that an average text page really only contains about 3000 characters, about 250 ms will be required for a "new page" of average text. Obviously, if the MicroVAX CPU is not able to process characters as fast as the hardware can take them, the processor will determine the overall performance.

3.3.7.3 **Variable-Pitch Text** -- The MicroVAX CPU uses any selected technique to compute the character spacing. Obviously, fixed spacing is useful in many applications. Variable spacing is used to justify a line of text by spreading extra pixels throughout the line. One computation involves finding the "real" number (integer and fraction) of pixels for each character (or space). Then, each character is spaced by the integer number of pixels, plus one more pixel, if the fractions have accumulated to more than one. This algorithm also allows any non-integer, but otherwise fixed, pitch to be simulated. This is useful for compatibility when dumping the bitmap to a dot-matrix printer.

A variable-pitch font is displayed with a different spacing for each character (read by the MicroVAX CPU from its width table). This is accomplished by changing the destination origin and fast DX or the destination origin only. The first way allows the font to be stored compactly, with the characters only occupying their width in the memory. However, the speed is somewhat reduced because the MicroVAX CPU must wait for the previous rasterop to be completed before it reloads the DX register and starts a new rasterop. The second way allows the same speed of operation as for fixed-pitch text, but the characters probably need to be stored in cells the size of the largest character, so that portions of unwanted characters are not copied to the screen. Some loss of speed results from copying larger areas. If overstriking of variable-pitch characters is performed, care should be taken to center smaller characters on top of the largest one, and to space to the next character by the largest space.

If the pitch of any character is larger than the destination fast DX setting, care must be taken to clear the area between characters that is not covered by the rasterops. This can be done by clearing the area of the text line before writing the character string.

The performance of variable-pitch text is similar to that described for fixed-pitch text above.

3.3.7.4 **Rotated and Scaled Text** -- Display of rotated or scaled text is similar to that for normal text, except that the destination fast and slow vectors need not be parallel to the X and Y axes, respectively, and the scale factors need not be one. The execution of a rasterop defaults to one pixel at a time. For a rotated rectangular text cell, the fast DX and DY are chosen to be the closest integer pair to the desired angle and size. (The magnification of images along the 45 degree axes may need to be accounted for.) Figure 3-22 shows examples of scaled and rotated text.

To insure proper tracking of an arbitrary baseline, real destination origin updates could be made for variable-pitch text, but with integer and fractional parts for both X and Y that correspond to the correct baseline angle. Using real position updates allows accurate simulation of any orientation or scale of a fixed-pitch text string; the characters will be within one pixel of their intended position.

The performance of text scaling cannot be expressed in terms of characters/s because of the varying number of pixels/character. These characters can be written at 500 K pixels/s, taking 1.7 s to fill a 840 K pixel screen.

Figure 3-22 Text Scaling and Rotation Examples

3.3.7.5 **Character Attributes** -- To set the character attributes, appropriate address processor and video processor chip registers are loaded; all characters retain the current attributes until the settings are changed. However, only the registers for those attributes that actually change between strings need to be loaded. But, if a permanent display list is being maintained, it may be desirable to reset all the attributes periodically (maybe the beginning of each line). The following list describes the intended implementation of attributes.

1. Reverse -- Reverse is a specific case of a more general class of attributes that modifies the logic function being performed on the destination data. Two bits are provided in the address processor chip command word to address one of the four logic unit function/mask complement registers in the video processor chips. These function registers could be set to the current definition of reverse and some other function (such as switching from replace mode to overstrike mode).

2. Underscore -- An underscore can be drawn as a vector of the desired thickness, and linear pattern mode can be used if a dashed or other patterned underscore is desired. Another technique would be to overstrike each character with another character having an underscore.

3. Overstrike -- Overstrike may be accomplished by appropriate loading (or not loading) of the destination origin register and the use of OR writing modes. If characters from variable-pitch fonts are overstruck, care must be taken in positioning.

4. Invisible -- The pen up bit in the address processor chip mode register may be used to disable writing, or the MicroVAX CPU can simply omit any text strings that are invisible. If erasure of the invisible area is desired, this must be performed explicitly.

5. Intensity, Color and Blink -- These are Z-axis parameters implemented through the color map. To enable writing the appropriate color in the bitmap planes or subplanes, a Z-axis I/D bus command accomplishes a Z-axis load of the video processor chip foreground and/or background registers. This sets all the F/B bits in each plane or subplane to either 1s or 0s. Source patterns are then loaded (via the video processor CSRs) either into a video processor chip mask register to accomplish overstrike writing, or into the source register to accomplish replace mode writing. Otherwise, blink may be performed by periodic writing to the bitmap.

6. Size, Italics and Rotation -- Any size, italic slope, or rotation is available by setting the destination fast and slow vectors, fast/slow scale factors, and providing appropriate destination origin update, as discussed above.

7. Subscript and Superscript -- These character offsets are obtained by appropriate loading of the destination origin registers. In this way, subscripts and superscripts may be nested, and angled baselines may be accounted.

3.3.8 Application to Graphics and Additional Graphics Support

Most graphics functions use the features introduced above. Linear patterns and polygon fill add some additional features.

3.3.8.1 Points and Vectors -- Solid color points and vectors are plotted as specific parallelograms, without any source operations. The source or foreground/background registers in the video processor chips are loaded with constants according to the desired color (usually with a Z-axis I/D bus command). For a point, the location is loaded into the destination origin registers, the destination fast and slow vectors are loaded with a length of 1 (DX, DY or both equal to one), and the rasterop command is loaded. The point is drawn at the pixel addressed by the destination origin registers.

A single pixel may be read from the bitmap by using a Z-mode bitmap-to-processor transfer to address one pixel. This operation is permitted even while scrolling because the 6-deep data FIFO can always accommodate one to six words without being subjected to MicroVAX CPU delays. A PTBZ can be used to write one pixel, but not while scrolling because the processor might not supply the data quickly enough after the PTBZ command. To avoid this problem, the single pixel rasterop described above is recommended. Because of the computational overhead required to initialize a rasterop or PBT, it is best to read or write many pixels at a time, if possible.

Figure 3-23 shows examples of the various vector types described below.

For a single-width vector, hole fill is disabled, the start point is loaded into the destination origin registers, the slow vector is set to a length of 1, the fast vector is set to the DX and DY of the desired vector, and the rasterop command is loaded. The point initially addressed by the destination origin ([XO,YO]) is drawn, as are all the points up to, but not including, the point [XO+DX,YO+DY]. Loading the DX and DY into the fast vector, rather than the slow vector, is preferred so that horizontal vectors can take advantage of fast mode (words written rather than single pixels).

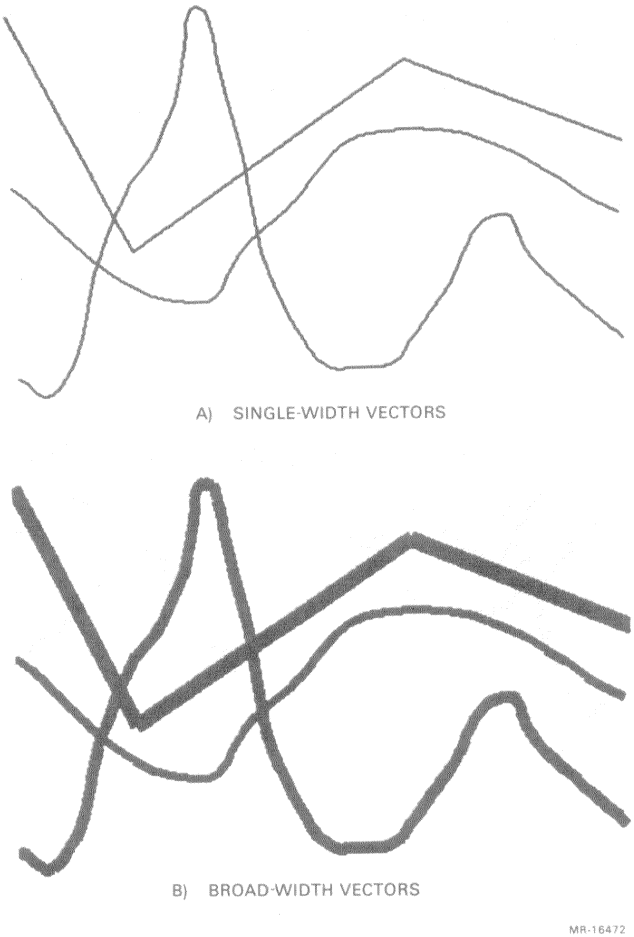


Figure 3-23 Vector Types

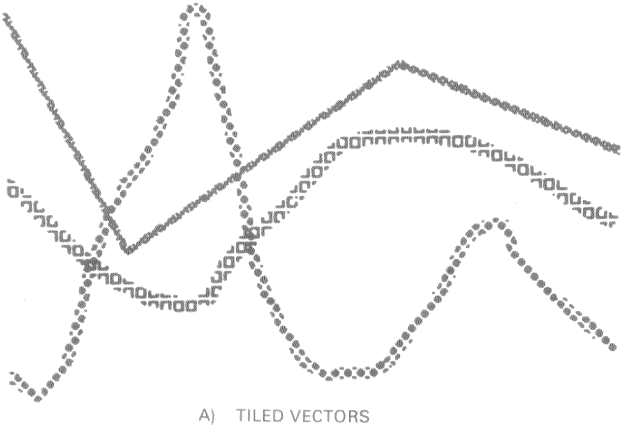
The set-up is the same for a broad vector except that hole fill must be turned on, and a slow vector with a length equal to the desired thickness and a direction perpendicular to the fast vector must be loaded. Take care at corners to prevent gaps between vectors (as seen in Figure 3-23). Gaps are filled with appropriate adjustments to the origin and length of the vectors, or a "vertex shape" may be copied to each vertex to cover the gaps. The area traced by a rectangular brush may be simulated by decomposing each vector segment into two parallelograms with one edge parallel to the side of the brush and the other edge parallel to the path.

Curves must be formed from a string of straight vectors or points.

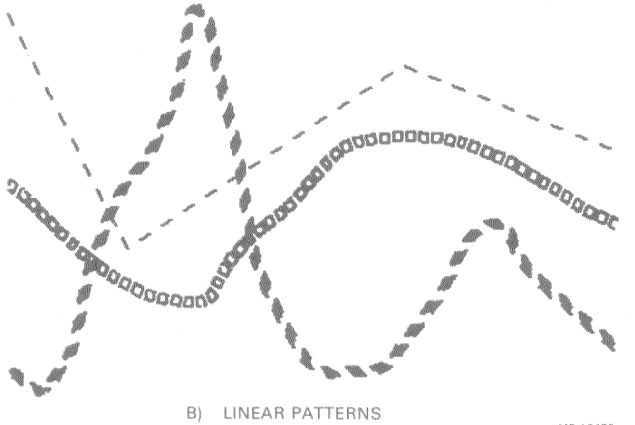
3.3.8.2 **Shading of Vectors - Linear and Tile Patterns** -- Vectors are shaded in one of two ways. Figure 3-24 shows examples of both.

1. Including a first source set for linear pattern mode provides linear patterns (like dashed lines, bullets, or stripes) that align themselves with the direction of a vector.

2. Including a second source set for tile mode provides tile patterns that are "uncovered" under the path of the vector.



A) TILED VECTORS



B) LINEAR PATTERNS

MR-16473

Figure 3-24 Vector Shading

Linear patterns are created by setting the address processor chip mode register to linear pattern mode. This uncouples the source and destination initialization from each other. The source DX (fast) and DY (slow) registers define the size and scanning direction for the pattern. (The magnitudes of the DX and DY registers are significant in this mode.) The size of the source pattern can be any integer in either the fast or slow directions. The path of a patterned vector is defined by the destination slow vector, and the width of the vector by the destination fast vector. Any fast or slow scale factor may be used in the normal manner. Linear patterns never operate in fast mode.

Linear patterns operate in the following manner.

1. As the destination scans its fast vector, the source provides pixels from its fast vector (as determined by the fast scaling).

2. If the source fast vector expires before the destination fast vector, the source reinitializes its fast vector at the origin (slow vector point) last used. Thus, the source scans its fast vector as many times as required to cover the length of the destination fast vector.

3. When the destination fast vector expires, the source and destination slow vectors are incremented (as determined by the slow scaling).

4. Both the source and destination fast vectors are initialized to the resulting new slow points. Fast scaling is also initialized. New fast vectors are scanned.

5. If the source slow vector expires before the destination slow vector, the source slow vector resets to the source 1 origin, and the whole source repeats again.

6. When the destination slow vector expires, a new destination vector can be initialized without disturbing the state of the source slow vector or the slow scale factor. However, if desired, loading a new source origin causes the source and slow scale to be initialized also.

The result is that a 2-dimensional pattern is scanned by the source transferred to the position, direction, and scale of the destination. If it is desired to scale the pattern so its size along a diagonal is the same as along an axis, the scale registers will need to be loaded along with each new destination fast and slow vector. Appropriate linear patterns for texturing vectors are normally coarse-grained to reduce the effect of sampling errors inherent in rotation and scaling.

Linear patterns require the fast vector to represent the discontinuous direction (narrow direction) of the destination, and the slow vector to represent the continuous direction (long direction). This protects the phase of the pattern from being disturbed when a new vector is commanded. The fast vector can be used to represent the long direction, but this requires that the phase of the linear pattern be reset between each vector command.

If it is desired to adjust the starting phase of the linear pattern (as is required if linear patterns are used to simulate filling an area with a tile whose width or height is not a power of 2), the source pattern can be written into the bitmap as a two-by-two set of four adjoining copies of the pattern. The phase can then be set by programming the source 1 origin to a point in the interior of one of the blocks (so that the source fast and slow vectors will overflow into the adjoining blocks). If the slow scale phase must also be controlled, the source and destination can be programmed to start the vector with pen up set in the mode register. Once the source origin and scale have reached the desired values, a new destination vector can begin with pen down.

To create tile patterned vectors, source 2 and the destination are enabled. The area under the destination are tiled as previously described. Appropriate tile patterns for texturing vectors are normally fine-grained and approximately isotropic (no stripes).

When patterned broad vectors are drawn, the same consideration must be given to gaps between vectors for solid vectors. (Notice that Figure 3-24 shows such gaps.) Also, the phase of linear patterns may need to be adjusted. No other considerations apply to tiled vectors.

3.3.8.3 **Fill Mode - Polygons** -- In general, a fill operation places a solid or textured object on the screen. The object's shape is described mathematically, as by a list of vertices. This is in contrast to a flood operation.

To help fill polygons, additional modes are available to appropriately configure the hardware. When polygons are being filled, the first source hardware is used internally to define one edge of a polygon. Therefore, polygons may be filled only with solid colors (destination only) or with the second source. By using the second source with fill mode, the interior of a polygon may be filled with a tile pattern; or, if tiling is disabled, a whole object from off screen can be moved, with "clipping" to the polygon outline (in addition to the normal rectangular region clipping).

The basic area fill is computed by defining two arbitrary vectors and writing the space between them by scanning fast, parallel to either the X- or Y-axis, and slow along the opposite axis. The direction of fast scanning (the fill axis) is selectable by mode, but the X-axis is preferred because fast mode (word access) is used. Each fast scan includes all points on both of the edge vectors.

Scanning is always from one vector (A) to the other vector (B), even if the vectors cross. (In such a case, the fast scanning direction reverses at the crossing and continues to fill the space between.) The last fast scan between the last point on any vector and a point on the other vector is not plotted (except under the conditions that cause doubling). This is similar to a normal rasterop, for which the first, but not the last, point is plotted.

A polygon to be filled entirely must be divided into two parts that can be directly filled by the VCB02 video subsystem hardware. Each part is intersected twice by every line that is parallel to the fill axis (excluding edge intersections, or edges parallel to the fill axis). If the border of a piece is traversed from the topmost (leftmost) vertex for an X (Y) fill axis, there is only one reversal of direction along the Y (X) axis. Figure 3-25 shows an example of a polygon that meets the preceding conditions for an X fill axis. To fill such a polygon:

1. The address processor chip is set to fill mode, with X fill, not baseline mode, source 2 parameters as required (if used), and video processor drawing modes as required.

2. One side of the polygon (from the uppermost point "top" to the lowermost point "bottom") is designated side A; and the other side is designated side B. It does not matter which side is designated A or B.

3. The vector strings that represent each side of the polygon are expressed as DX and DY pairs to the next point (as with ordinary vectors). (See the A0 in Figure 3-25.) After an initial origin load, all vectors are relative.

4. The vertices from both sides are sorted as a single group by increasing the Y coordinate. If a vertex on each side has the same Y value (always the case at the top and bottom), the A and B vertices are sorted as a pair. In the example: A0/B0, A1, B1, A2, A3/B2, A4, and A5/B3, A3, A4, and B2 all have the same Y coordinate.

5. The DX,DY pairs are sorted as single vectors or vector pairs according to the order determined by sorting the vertices. (Actually, the vertex information is used only to determine this vector order.) In the example: A0(DX:DY)/B0(DX:DY), A1(DX:DY),....., B2(DX:DY), and A4(DX:DY). (No vectors start from A5 or B3, but a final A(0:1)/B(0:1) vector pair may be added at the point A5/B3 to complete the last scan of the polygon.)

6. The destination origin registers are loaded with the start point (A0), and the source 1 X origin is loaded with the X component of the origin (B0). (Actually, B0 need not be the same point as A0, assuming they both have the same Y coordinate. The source 1 Y origin register is ignored for X fill. The roles of the two source 1 origin registers are reversed for Y fill.)

7. The first vector pair is loaded into the destination slow DX and DY registers (A vector) and the source DX and DY registers. (B vector, the magnitudes of the source DX, and DY are significant in fill mode.)

8. The address processor chip command register is then loaded with a rasterop command, including a source 2, if needed. Source 1 should never be enabled in fill mode.

9. The space between the two vectors is filled (by the hardware) parallel to the X-axis as the vectors are advanced synchronously in Y.

10. When the address processor chip signals that it needs more data (rasterop complete flag), the MicroVAX CPU loads the next vector or vector pair in the list into the appropriate vector registers (A to destination slow and B to source 1). The MicroVAX CPU then loads the address processor chip command register again, continuing with the previous step until all required vectors are plotted.

11. The process is terminated when the MicroVAX CPU loads new data into the destination origin registers (which causes the positions to be reinitialized), or switches the address processor chip out of fill mode.

Fill with Complement Mode

The procedure above plots all the points that would have been plotted by the edge vectors, plus all the points between. However, some of the points are plotted more than once, creating potential problems for complement mode operations (any operation that XORs or uses the complement of the destination). The MicroVAX CPU fixes this problem (for most of the important cases) by forcing that all pixels to be written an even number of times actually be written an odd number of times (ordinarily, writing doubled pixels a third time).

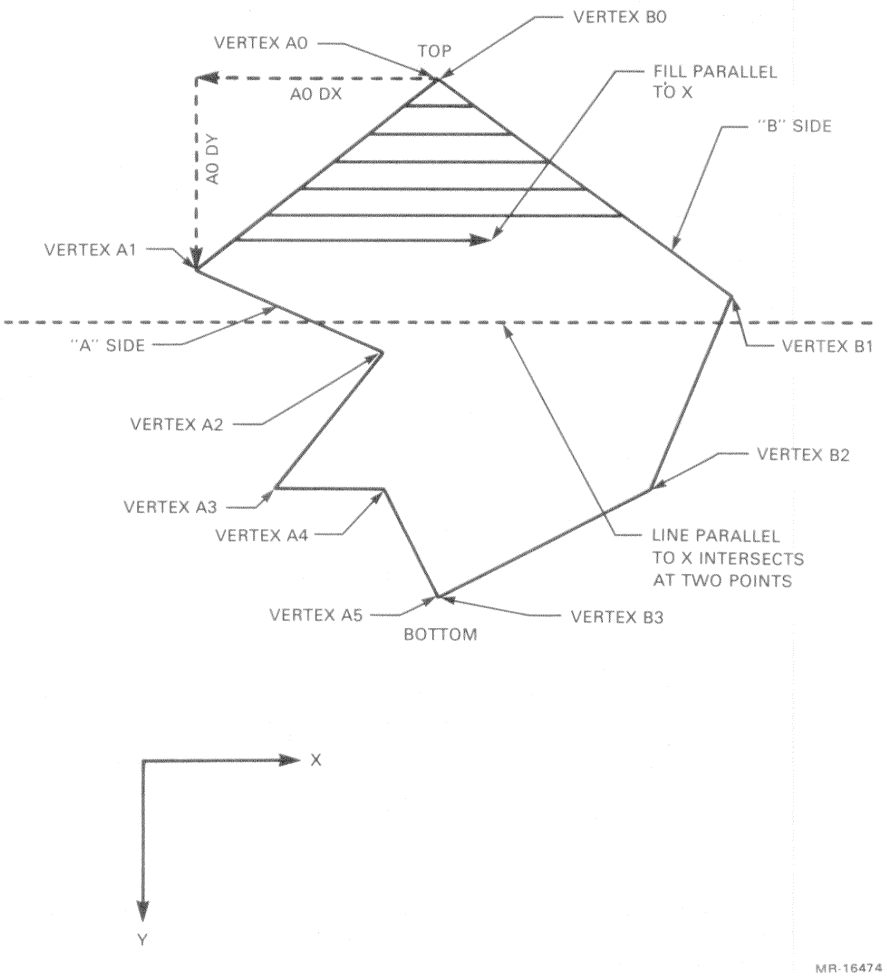


Figure 3-25 Example of a Polygon Fill

Doubling occurs each time an A or B vector is loaded, and when the previous vector or the vector on the other side (whether or not it has also expired):

1. Was (is) "inbound" (approaching the other vector)

2. Has its major axis parallel to the fill axis

3. Satisfies the following equation, which identifies the places along the minor axis of a vector where two or more points are addressed.

   ([(2dist + 1)maj - 1] modulo 2min) + 2min < 2maj

   where:

   dist = magnitude of the distance, perpendicular to the fill axis, from the start of a vector to the current drawing point (for X fill, the sum of the DYs from the start of the fill on the side that has expired, minus the sum of the DYs to the start of the vector being tested).

   maj = magnitude of the major axis (larger of DX or DY) for the vector being tested.

   min = magnitude of the minor axis (smaller of DX or DY) for the vector being tested.

According to the conditions above, any vector that expires has an inbound slope greater than or equal to 2:1 (major:minor), and a major axis parallel to the fill direction always causes doubling. Any other expiring vector never causes doubling. This can be by substituting major/2 for the minor axis in the equation above. For a vector that is not expiring, doubling will always occur for slopes greater than or equal to 2:1, but doubling will never occur for slopes less than or equal to 1:1. The equation must be evaluate for slopes between 2:1 and 1:1.

If doubling occurs, it can be corrected (tripled) by preceding the next vector with a unit length vector inbound and another unit length vector outbound. If both the A and B sides have expired, this fix is applied to both vectors.

This extra procedure for complement mode corrects all doubled pixels, except in two circumstances. If the two edges cross, some uncorrectable doubling may occur on one scan. (This operation is normally avoided, except in baseline mode.) If an inbound vector of stated slope is followed by an outbound vector that also meets the slope condition, the pixels common to the vectors but lie outboard of the vertex are doubled. If necessary, these conditions can be computed by the MicroVAX CPU and corrected later by tripling.

As an example of correction of doubling of pixels (see Figure 3-25), assume that vectors A1 (from A1 to A2) and A3 (from A3 to A4) are the only two inbound vectors with a slope flatter than 2:1. Assume also that B2 (from B2 to B3) is the only vector that is inbound with a slope between 2:1 and 1:1, but there are no expirations of the A side during its length. If the doubling is to be corrected, the following complete vector list would be used: A0(DX:DY)/B0(DX:DY), A1(DX:DY), B(-1:0), B(1:0), B1(DX:DY), A(1:0), A(-1:0), A2(DX:DY), A3(DX:DY)/B2(DX:DY), A(1:0), A(-1:0), A4(DX:DY), A(0:1)/B(0:1). The last vector pair is used to plot the last vertex of the polygon, if desired.

If two polygons are butted together (that is, one shares some edges with another), the common edges will be written twice. But because all the pixels on the edge vector have been written exactly twice, it is possible to ensure an odd number of writes to the edge pixels by writing only the edge vectors again (in the same direction as when filling), including the last point on the last vector.

Baseline Fill
A simplified fill mode is provided that fills from a single vector to either a horizontal (for Y fill) or vertical (for X fill) baseline. The position of the baseline is programmed by loading the source 1 Y origin (for Y fill) or the X origin (for X fill). In addition, the source 1 DY must be zero for Y fill, and the DX must be zero for X fill. In either case, the other delta must be nonzero. Only a single vector path is required, and the path is allowed to reverse the direction of its component that is parallel to the baseline (not allowed with 2-line mode). However, the space between two lines that converge on a point from the same quadrant is difficult to fill in this mode. (Some products have provided a fill to a point function for this case, but this creates more problems for complement mode.)

With baseline mode, the same problems and procedures apply when using complement operations, and if the vector path crosses the baseline and then reverses direction (parallel to the baseline, as when filling a circle). The baseline is written twice, as if two polygons had been butted together, and the baseline needs to be drawn again as a single vector to correct for the even number of writes.

3.3.8.4 Polygon Flood -- A flood operation colors or textures the interior of an area whose outline is described in the bitmap. This is in contrast to a fill operation. No explicit support for a polygon flood function is provided by the address processor chip. This function requires the path of writing in the bitmap to be conditional on the previous contents of the bitmap. In the hardware, the path of writing is controlled by the address processor chip, but only the video processor chips see the memory data.

Flood can be implemented by using the following commands.

1. The processor/bitmap transfer facility is used to read a scan of the bitmap in Z-mode or one plane of the bitmap in X-mode.

2. Each returned pixel value is checked for a match with a set of boundary colors, and a count is maintained of the pixels received.

3. When a match is found, the BTP transfer is terminated by loading a cancel command to the address processor chip.

4. A vector is commanded to write the desired pixels on the scan with a color or pattern in the normal way.

5. The flood algorithm then steps to a point on the next scan, or continues at a previous seed point, in the normal way.

The speed of this operation is determined by the BTP scanning time, plus the overhead time of the MicroVAX CPU (which could be significant if it takes more than 2 μs for each comparison, or if the flooded area is narrow). The drawing time is generally not significant as compared to scanning and decision-making. Using this technique, an empty full-page screen (1024 X 864) has been flooded in less than 2 seconds.

3.3.8.5 Objects -- Ordinary rasterops can be used to move objects or windows on the screen. In a multiplane environment, rectangles can be copied by moving the pixels in all planes, in parallel, to new [X,Y] locations in the same planes. This is referred to as a raster move because an entire rectangle is copied. The source 1 is indexed and offset, from either on or off screen. If a source 2 is used, it must be from off screen.

To move an object (that is, alter only the destination pixels) within the boundary of the object, the hardware must be able to distinguish between object and background. This can be done by defining the foreground of the object in one plane. The area of this plane within the rasterop rectangle is then transmitted to the other planes during the rasterop for use as a mask that prevents the alteration of background pixels. This operation is referred to as an object move. Alternatively, the second source can be used to copy an object from off screen, with "clipping" to the destination shape.

The paragraphs above described the movement of color objects (objects defined in more than one plane). It is also possible to move an object specified only by a foreground shape, and assign a foreground and background color to it at the time it is moved. This type of operation is described in the section on writing text, Paragraph 3.3.7.

3.4 ADDRESS PROCESSOR AND VIDEO PROCESSOR CHIP REGISTERS AND COMMANDS

This section defines the interface to the address processor and video processor chips in detail. There are basically two types of commands to the hardware: direct commands to the address processor chip, and I/D bus commands, which are passed through the address processor chip to the video processor chips and chip select registers.

3.4.1 Address Processor Chip Registers and Commands

All address processor chip commands are the results of address processor chip register loads by the MicroVAX CPU or the VCB02 video subsystem's DMA logic (in the DMA gate array). Before a description of the address processor chip commands and interface, the address processor chip registers are defined below.

3.4.1.1 Address Processor Chip Registers -- The address processor chip contains many loadable registers that hold parameters for rasterops, set system timing, and control the operation of the chip. Unless otherwise specified, the registers are write-only and contain 14 significant bits in 2's complement representation. (This includes the sign bit; two MSBs of a word are ignored, except for test purposes.) The loading of some registers causes the address processor chip to take specific additional actions. These actions are described with the corresponding registers.

Different types of coordinates are implied by the various position registers. "World coordinates" refer to the coordinate system of the image as viewed by the MicroVAX CPU before index values (if enabled) are added to translate the image to "device coordinates". Device coordinates describe a physical position on the screen: [0,0] is the upper left corner with X increasing to the right and Y increasing downward. "Memory coordinates" are obtained internally to account for down-scrolling by adding the Y offset to all addresses to the on-screen part of the memory (any memory location accessed by a major read cycle for screen refresh), delineated by the X limit and Y limit registers.

The DMA logic in the DMA gate array loads registers through the register address counter, and thereby, loads several sequential registers. The register addresses are assigned to allow all common combinations of registers to be adjacent. This allows loading to be accomplished with a minimum of address register loads in the DMA data list (DMA display list).

The register addresses (or the address of the first of a group of registers) are in hexadecimal and enclosed in brackets []. Where specified, bit assignments (or the LSB of a group of bits) are enclosed in angle brackets <>.

Interface Control Registers

[0] ADDRESS COUNTER (ADCT) -- This address controls a counter that provides indirect access to the address processor chip registers for use during DMA. If the MSB of the data word is a 0 when this register is written, the data is placed in the register pointed to by the contents of the counter, and then the counter is incremented. If the MSB of the data is 1 when this register is written, the low 6 bits of the data replace the original contents of the address counter. The only exception is that, if the address counter is pointing to either the I/D data register (IDD) or the I/D scroll data register (IDS), the MSB of the data is ignored by the address decoding, and all 16 bits are loaded into the appropriate I/D register. When a read is directed to the address counter, the register it points to is accessed and the counter is always incremented subsequently. (Note that only a few registers are actually readable.) The counter is cleared when the -INIT pin is asserted.

[1] REQUEST ENABLE (REQ)

[2] INTERRUPT ENABLE (INT) -- The unencoded bits of these two registers allow any selection of the corresponding 13 bits of the status register to be enabled for asserting the request or interrupt output pins. A 1 in either of these registers enables a status bit. If a bit in the status register is a 1 and that bit is enabled by one of the enable registers, the output pin corresponding to the enable register is asserted. Both enable registers are cleared by assertion of the -INIT pin on the address processor chip.

In the VCB02 video subsystem, the request register and pin control the data requests to the DMA gate array. In general, only one of these request conditions is set at a time because the DMA process waits for one specific event before it passes the next data word.

The interrupt pin and register are provided to control interrupt requests to the interrupt logic in the DMA gate array. During the interrupt service routine, the MicroVAX CPU needs to read the status register to determine the interrupting condition(s), if more than one interrupt is enabled.

[3] STATUS (STAT) -- The status register provides various indications of the internal progress of the address processor chip. The conditions that set and clear each bit are listed below. This register is read-only (except as noted below).

<0> Pause Complete -- Set when the screen refresh process reaches the Y address (in device coordinates) that the pause register was set to when the current frame began. Cleared by writing a word to the status register with a 0 at this bit position; unaffected by a 1 at this bit position or by values of other bits. This status bit will queue two pause events. Thus, if in one frame the pause register was set near the bottom of the screen and in the next frame the pause register was set near the top of the screen, both pause events will be separately detected, even if the second event occurs before the first event. However, this means that if the pause register is set within the active screen and the pause status bit has not been cleared within the previous two frames, two clears will be required to make this status bit stay low. The pause status bit can be prevented from being set by programming the pause register to a value greater than that of the end vertical period event in the YCT- registers. The -INIT pin clears any queued pause event, but one explicit clear by writing to the status register is still required to set the pause status bit low.

<1> Scroll Service (Frame Sync) -- Set at the start of a frame, when new scroll parameters may be loaded. Cleared by writing a word to the status register with a 0 at this bit position; unaffected by a 1 at this bit position or by values of other bits. This bit differs from the vertical blanking bit in that the latter is set at the beginning of the vertical blanking interval.

<2> Rasterop Initialization Complete -- Set when initialization of a rasterop, processor-to-bitmap transfer (PTB), or bitmap-to-processor (BTP) transfer is complete. This indicates it is safe to load the source 1 and destination origin registers, or to load a new (but not different) rasterop (but not PTP or BTP) command. Cleared when any rasterop, PTB, or BTP command is loaded. Set by a cancel command or the -INIT pin.

<3> Rasterop Complete -- Set when rasterop, PTB, or BTP address calculations are complete and no further command is pending. This indicates it is safe to load any rasterop parameters, such as: origins, DX/DY pairs, source 2 parameters, scale factors, and the mode register, or to load a new (but not different) rasterop (but not PTB, or BTP) command. Cleared when any rasterop, PTB, or BTP command is loaded. Set by a cancel command or the -INIT pin.

<4> Address Output Complete -- Set when all addresses calculated by all pending rasterops, PTBs, or BTPs have been used. This indicates it is safe to load any update parameters, including: clipping boundaries, indexes, I/D data, or different commands. In addition, during BTP (bitmap-to-processor) commands, this status bit does not go high until the IDD FIFO is also empty, indicating that all data transferred by the BTP has been picked up by the MicroVAX CPU or DMA controller (DMA gate array). Cleared when any rasterop, PTB or BTP command is loaded. Set by a cancel command or the -INIT pin.

<5> I/D Data Receive Ready -- Set if the I/D data FIFO has a data word available for reading. Cleared if the FIFO becomes empty. When a rasterop, PTP, BTP, or cancel command is loaded to the command register or bus, initialization occurs, the FIFO is initialized, and this bit is cleared. Note that this bit is set any time data is in the FIFO (such as during PTB, BTP, or I/D command sequences), and should be masked when not used.

<6> I/D Data Transmit Ready -- This bit has four interpretations, depending on the commands being executed by the address processor chip:

a. When a rasterop, PTB, or BTP command is loaded to the command register, the FIFO is initialized and this bit is set. During a rasterop or PTB command, this bit is set if the I/D data FIFO (6 words long) can accept an additional data word. The bit is cleared if the FIFO becomes full. Though this bit responds during rasterop commands, I/D data is not used, and any data loaded remains in the FIFO during the command.

b. During an I/D command (after an I/D command has been loaded and until some other command is loaded), this bit is set if a new data word may be loaded to the I/D data register. Cleared when an I/D command is loaded to the command register. This bit should also be checked before loading another command after an I/D command. However, if at least three major cycles (one major cycle plus a sync cycle, about 3 μs) pass before the new command is loaded, this status bit need not be checked.

c. When a cancel command is loaded, this bit is momentarily set low. When execution of the cancel command is complete, this bit is set high to indicate that the IDD FIFO is clear and a new command may be loaded. This bit should be checked to indicate completion of a cancel command. This is not required if at least four major cycles are guaranteed to pass before the next command is loaded.

d. If the I/D data register is loaded when a rasterop, PTB, or BTP is not in progress (see IDD register), or the -INIT pin is asserted, the FIFO is initialized and this bit is set.

<7> I/D Scroll Data Ready -- Set if a new data word may be loaded to the I/D scroll data register. Cleared when an I/D command is loaded to the I/D scroll command register.

<8> A portion of at least one rasterop clipped at the top boundary.

<9> A portion of at least one rasterop clipped at the bottom boundary.

<A> A portion of at least one rasterop clipped at the left boundary.

<B> A portion of at least one rasterop clipped at the right boundary.

<C> A portion of at least one rasterop was successfully drawn (did not clip at any boundary).

The preceding five clipping bits accumulate clipping activity by having clipping results for each destination write cycle ORed with their previous contents. These bits are asserted high. The last bit may be used to aid in selecting an algorithm by indicating that a rasterop was at least partly inside the clipping rectangle. Clipping information for a rasterop is complete after the address complete status bit is asserted. Reading the clipping status bits during a rasterop can yield undefined results.

These five bits are selectively cleared by a write to the status register with a 0 in each corresponding clipping bit position to be cleared, and a 1 in positions to be unaltered. Zeros or ones in other bit positions have no effect on these status bits. Clearing of the clipping status bits during a rasterop gives unpredictable results. These bits should be cleared only after the address complete status bit is asserted.

<D> Vertical Blanking -- Set at the start of the vertical blanking interval. Cleared by writing a word to the status register with a 0 at this bit position; unaffected by a 1 at this bit position, or by values of other bits. This bit differs from the scroll service bit in that the latter is set near the end of the vertical blanking interval.

[4] RESERVED -- Test function.

[5] SPARE

[6] RESERVED -- This register address is used only for a test function because it precedes the IDD register. (If this address were accessed through the address counter, it would force a load to the IDD.)

[7] I/D DATA (IDD) -- When a rasterop, PTB, BTP, or cancel command is loaded into the command register or the -INIT pin is asserted, the 6-deep FIFO at this register address is cleared. During PTP or BTP commands, the FIFO is either read or written (depending on the direction of the transfer command) to transfer data between the processor and the bitmap. The I/D interconnect data FIFO is 16 bits wide. When screen data is passed through the I/D data FIFO (in X-mode), the LSB corresponds to the leftmost pixel in the word.

Whenever a load to the I/D data register occurs and the address output complete status bit is set (a PTB, BTP, or rasterop command is not in progress), the FIFO is cleared as the data word is loaded; only the loaded word remains in the FIFO. (Only the last word loaded to this register is available for transmission on a subsequent I/D command.) Reading this register under the same conditions does not clear the FIFO. Note that the address output complete status bit continues to indicate that a BTP operation is not complete until all data has been removed from the FIFO.

[8] COMMAND (CMD) -- This address accesses the same command register described for the mode register. The duplicate address is provided to ease access through the address counter because the command register is frequently written after writes to the I/D data or mode registers.

[9] MODE (MDE) -- Sets various rasterop execution modes as described by the following bits.

<0> 00 = Normal Mode -- The first source is coupled to the destination so that the source size is determined from the scaled destination size.
01 = Reserved.

<1> 10 = Linear Pattern Mode -- The first source is uncoupled from the destination so that the source pattern size is determined by the source DX and DY registers.

11 = Fill Mode -- The destination slow generator computes the A edge vector, the first source generator computes the B edge vector, and the space between is filled.

The next two bits are ignored, except when fill mode is selected in the preceding two bits.

<2> 0 = The fill area is scanned parallel to the X-axis (X fill).
1 = The fill area is scanned parallel to the Y-axis (Y fill).

<3> 0 = Normal. Two edge fill.
1 = Baseline fill. The B edge generator is locked to form a vertical or horizontal baseline for fill (depending on the scan direction selected with the preceding bit).

<4> 0 = Hole fill disabled. This is the normal setting for single-pixel-wide destinations.
1 = Hole fill enabled. This is the normal setting for all other destinations.

<5> 0 = First source indexing disabled.
1 = First source indexing enabled.

<6> 0 = Destination indexing disabled.
1 = Destination indexing enabled.

<7> 0 = Pen up, writing disabled.
1 = Pen down, writing enabled.

[A] COMMAND (CMD) -- Address processor chip command register, used for all commands by the update process. Some combinations of bits result in undefined operations. When performing a series of operations (for example, text or vector strings), an identical command may be reloaded after the rasterop initialization complete status bit is set: if only the source 1 and destination origins have been changed, or after the rasterop complete bit is set; or if other rasterop parameters have been changed. To load a new command, the address output complete status bit must be set.

<0> 8 Bits. I/D Command.

<8> 2 bits. Function select.
00 = Cancel all active and pending commands (except ICS).
01 = I/D command.
10 = Rasterop command.
11 = Processor-to-bitmap or bitmap-to-processor transfer command.

<A> Destination Enable.

<B> First Source Enable.

<C> Second Source Enable.

<D> Test Function (normally 0).

<E> Test Function (normally 0).

<F> Reserved (normally 0).

Scroll Registers

During active scrolling, scroll registers must be loaded by the scroll process before the start of vertical blanking. The address processor chip performs various functions during vertical blanking, depending on the type of scroll activity pending for the next frame.

[B] RESERVED -- This register address is used only for a test function because it precedes the IDS register. (If this address were accessed through the address counter, it would force a load to the IDS.)

[C] I/D SCROLL DATA (IDS) -- This register contains data to be transmitted to the I/D interconnect during scroll process I/D commands. The last word loaded to this register is transmitted on the next I/D scroll command. This register is 16 bits wide.

[D] I/D SCROLL COMMAND (ICS) -- I/D command register for the scroll process. Commands directed through this register are transmitted on the I/D interconnect without interference to any update activity that may be in progress (assuming that only scroll-related registers in the video processor chips are addressed).

[E] SCROLL X MIN (PXMN) -- Left boundary of the scroll region; the left-most pixel in the scroll region.

[F] SCROLL X MAX (PXMX) -- Right boundary of the scroll region; the first pixel outside the right side of the scroll region. The two LSBs of the X boundaries are ignored. The X boundaries may be specified only to a multiple of four pixels.

[10] SCROLL Y MIN (PYMN) -- Top boundary of the scroll region; the uppermost pixel in the scroll region.

[11] SCROLL Y MAX (PYMX) -- Bottom boundary of the scroll region; the first pixel outside the bottom of the scroll region.

Scroll boundaries are in device coordinates so that they are unaffected by the index values (that is, they stand still on the screen). These registers are double-buffered so that the values loaded become active at the start of the following frame.

[12] PAUSE (PSE) -- The Y device coordinate that specifies the scan that, when displayed, causes the pause status bit to be set or a second pause event to be queued. (See the description of the pause complete status bit above.) This register is double-buffered so that comparison with the new value loaded begins at the start of the following frame and continues throughout that frame. Only the low 11 bits of this register are significant.

[13] Y OFFSET (PYOF) -- Offset added to device coordinates to get memory coordinates. Decremented by the MicroVAX CPU during down-scrolling; ranges between 0 and the height of the displayed portion of the bitmap memory (the same value minus 1; see below). The offset value loaded to this register should be the one in effect when the following frame is complete, and stored in the Y limit register, due to the multiple buffering of this register.

[14] Y SCROLL CONSTANT (PYSC) -- Specifies the vertical distance to be scrolled in one frame time. Loading of this register causes the region selected by the scroll boundaries to be scrolled during the following frame (by the vertical distance loaded to this register, or the horizontal distance loaded to the scroll constant registers in the video processor chips). If no value is loaded during a frame, no scrolling occurs in the next frame.

<0> 12 Bits. The vertical magnitude (unsigned distance) of the scroll. If a nonzero X scroll constant is specified in the video processor chips, the Y scroll constant must be zero.

<C> 0 = Up, left or right scrolling.
1 = Down-scrolling. If down scrolling is specified, the scroll direction bits in the video processor chips should also be set to down and left.

<D> 0 = Normal scrolling. The scroll direction bits in the video processor chips should be set according to the desired scroll direction.
1 = Erase mode. The entire scroll region is cleared to the fill color regardless of the settings of the scroll constants. Up-scrolling must be specified in both the address processor and video processor chips.

Update Control Registers

[15] PENDING X INDEX (PXI)

[16] PENDING Y INDEX (PYI)

[17] NEW X INDEX (NXI)

[18] NEW Y INDEX (NYI)

[19] OLD X INDEX (OXI)

[1A] OLD Y INDEX (OYI)

Index values are added to the rasterop addresses to adjust them for scrolling and the location of regions on the display. The pending values are those automatically loaded into the new registers at the start of the next frame. The contents of these registers are not destroyed by this load; so, if no scrolling takes place in the next frame, these registers need not be reloaded. The new values are the indexes that apply to data that has already been moved during the current frame. The contents of these registers are loaded into the old registers at the start of the next frame, before the pending values are loaded into the new registers. The old values are the indexes that apply to data that has not yet moved. The index values apply to the region that is being updated because they only affect the update addresses.

These registers are normally loaded by the scroll process, if the scrolling region is also the current update region, and are also loaded by the update process whenever the update region changes. All these registers may be loaded directly, as is required when changing from one update region to another. They must always be loaded in the order: pending, new, old.

[1B] CLIP X MIN (CXMN) -- Left clipping boundary; the left-most pixel in the update region.

[1C] CLIP X MAX (CXMX) -- Right clipping boundary; the first pixel outside the right side of the update region. The two LSBs of the X boundaries are ignored. The X boundaries may be specified only to a multiple of four pixels.

[1D] CLIP Y MIN (CYMN) -- Top clipping boundary; the uppermost pixel in the update region.

[1E] CLIP Y MAX (CYMX) -- Bottom clipping boundary; the first pixel outside the bottom of the update region.

Clipping boundaries are in device coordinates so that they are unaffected by the index values. Because clipping is in device coordinates, the clipping boundaries stand still on the screen during scrolling. Thus, they are not usable as a drawing function, but merely to contain the image within the borders of the scroll window. They affect whether or not destination data is written and the setting of the clipping status bits. All rasterop calculations are still performed.

[1F] SPARE

Rasterop Control Registers

[20] FAST SOURCE 1 DX (FSDX) -- Fast extent for the first source. This can only be in the + or - X direction; there is no Y component.

[21] SLOW SOURCE 1 DY (SSDY) -- Slow extent for the first source. This can only be in the + or - Y direction; there is no X component.

Only the sign bit (MSB, bit 13) of the source 1 delta registers is significant in normal mode. (Even the signs are ignored if the source 1 is not enabled.) The whole value is significant in linear pattern and fill modes. In fill mode, only values in the range 2_^12-1 to -2_^12 are allowed in the source 1 delta registers because these values are multiplied by 2 in some calculations internal to the address processor chip.

[22] SOURCE 1 X ORIGIN (SXO) -- X coordinate of the first source origin.

[23] SOURCE 1 Y ORIGIN (SYO) -- Y coordinate of the first source origin.

The first source origin components can be in either world or device coordinates, depending on the setting of index mode for the first source. The loading of either of these registers in linear pattern mode resets the source pattern scan and the slow scale calculations.

[24] DESTINATION X ORIGIN (DXO) -- X coordinate of the destination origin.

[25] DESTINATION Y ORIGIN (DYO) -- Y coordinate of the destination origin.

The destination origin components can be in either world or device coordinates, depending on the setting of index mode for the destination. Loading of either of these registers during fill mode causes a new fill position to be initialized. Otherwise, all vector extents are relative to the end of the previous vector.

Neither the first source nor destination origin registers are used during any mode after the rasterop initialization is complete. These registers may be loaded with new values after the rasterop initialization complete status bit is set.

[26] FAST DESTINATION DX (FDX) -- X component of the destination fast vector.

[27] FAST DESTINATION DY (FDY) -- Y component of the destination fast vector.

[28] SLOW DESTINATION DX (SDX) -- X component of the destination slow vector.

[29] SLOW DESTINATION DY (SDY) -- Y component of the destination slow vector.

Only values in the range 2_^12-1 to -2_^12 are allowed in the destination delta registers because these values are multiplied by 2 in some calculations internal to the address processor chip.

[2A] FAST SCALE (FSC) -- Scale factor relating the fast vectors of the first source and destination in normal and linear pattern modes.

[2B] SLOW SCALE (SSC) -- Scale factor relating the slow vectors of the first source and destination in normal and linear pattern modes.

For each of the scale factors, the MSB (bit 13) indicates up-scaling (if a 0) or down-scaling (if a 1). The magnitude is specified in the remaining 13 bits, with the binary point preceding bit 12. The address processor chip adds 0.0000000000001B to each scale factor. The address processor chip ignores the scale factors unless both the source 1 and destination are enabled.

[2C] SOURCE 2 X ORIGIN (S2XO) -- X coordinate of the second source origin.

[2D] SOURCE 2 Y ORIGIN (S2YO) -- Y coordinate of the second source origin.

The source 2 origins are added to the unindexed destination addresses (or low bits of them, if tiling is enabled). The second source origin components are always memory coordinates. The source 2 cannot be indexed and it is not adjusted for the effects of Y offset; therefore, it must be off screen. If tiling is enabled, the source 2 origin must be on a word boundary in the memory.

[2E] SOURCE 2 HEIGHT AND WIDTH (S2HW) -- The contents of this register determine the size of the source 2 tile.

<0> 3 Bits. Tile width (W) from 0 to 7; the width of the tile is 2_^(W+2) (from 4 to 512).

<3> Reserved (normally 0).

<4> 3 Bits. Tile height (H) from 0 to 7; the height of the tile is 2_^(H+2) (from 4 to 512).

<7> 0 = Tile height and width enabled; high bits of destination component truncated before adding to source 2 origin.
1 = Tile height and width disabled; all destination address bits added to source 2 origin.

[2F] ERROR 1 (ERR1) -- Error adjustment added to the error register during rasterop initialization for the slow destination (A side in fill mode).

[30] ERROR 2 (ERR2) -- Error adjustment added to the error register during rasterop initialization for the fast destination in normal and linear pattern mode, and for the source 1 (B side in fill mode).

Screen Format Control Registers

[31] Y SCAN COUNT 0 (YCT0)

[32] Y SCAN COUNT 1 (YCT1)

[33] Y SCAN COUNT 2 (YCT2)

[34] Y SCAN COUNT 3 (YCT3)

These four Y parameter registers program all vertical timing for the VCB02 video subsystem. Each register determines the time of one vertical event, such as the assertion of vertical blanking. The event is specified by the two MSBs (bits 12 and 13). The low 11 bits determine the time of the event (scans for vertical synchronization events, and scans minus 1 for other events), starting from vertical unblanking. The events are stored in the registers in order of increasing time (low 11 bits), starting in YCT0. The highest time value is programmed for the vertical period event. If the horizontal period is set for an odd number of major cycles (N odd, where "N" is defined for the horizontal period event, below), there must be an even number of scans in a frame. This is so there will still be an even number of major cycles in a frame. (The vertical period event must be set to an odd number.) Vertical blanking is deasserted (set low) when the vertical period is reached. The bits of the Y scan count registers are assigned as follows.

<0> 11 Bits. Time of event from deassertion of vertical blanking (scans for vertical synchronization events, and scans minus 1 for other events). The end vertical period event is set to the number of scans in the frame, minus 1.

<B> Reserved (normally 0).

<C> 2 Bits. Event.

00 = End vertical period. Set vertical blanking low in following scan.
01 = Set vertical blanking high.
10 = Set vertical sync low.
11 = Set vertical sync high.

The VCB02 video subsystem provides a composite synchronization signal. The vertical synchronization signal chooses between two events in the X scan counter that provide the variable-width horizontal synchronization pulse needed in a composite synchronization system. When using composite synchronization, the vertical synchronization interval must be delineated by the high state of the vertical synchronization signal, allowing the longer horizontal synchronization time to be determined by the "100" or "101" event in the X scan counter (see below).

A sample assignment of events for a composite sync application: YCT0 sets vertical blank high, YCT1 sets vertical sync high, YCT2 sets vertical synchronization low, and YCT3 sets vertical blanking low and restarts the vertical counter.

A system synchronization request is generated by the address processor chip in the scan prior to the deassertion of vertical blanking (scan prior to the first displayed scan). The exact timing of the synchronization request within that scan is determined by the X scan count registers described below.

[35] X SCAN CONFIGURATION (XCON)

<0> 6 Bits. Number of major read cycles used for each scan. This is normally set to the smallest integer greater than or equal to the number of pixels to be displayed on each scan divided by 128. In the VCB02 video subsystem, this must be set to 8 (1024/128 = 8). Memory refresh is accomplished approximately every 0.5 ms, plus any gap occurring during vertical retrace. See the discussion of the XCT- and XL registers for required relationships to this register.

<6> 2 Bits. Bus width mode must be programmed here and in the video processor chips before any bitmap memory access is attempted. Once bus width is initialized, most operations are independent of the bus width. Exceptions are X-mode processor/bitmap transfers, source 2 tile origins, and video processor scroll boundaries. The assignment of X and Y address bits to the address processor chip output pins depends on bus width and the memory configuration bit below. The VCB02 is a 16-bit bus-width system and requires both these bits set.

00 = 4-bit bus width.
01 = 8-bit bus width.
10 = Undefined.
11 = 16-bit bus width.

<8> Memory configuration controls the number of row addresses refreshed on each scan. In "greater than or equal to 1024" mode, 8 are refreshed. In "greater than or equal to 512" mode, 4 row addresses are refreshed. The number of pixels read in each scan is the number of refresh reads programmed in XCON * 16 * 8. The VCB02 video subsystem requires this bit to be reset, which selects 1024 mode (XCON * 16 * 8 = 1024).

0 = $1024 \leq$ Number of pixels read from memory/scan < 2048
1 = $512 \leq$ Number of pixels read from memory/scan < 1024

[36] X LIMIT (XL) -- Width of that portion of the memory read by the screen refresh process (could be more than displayed). This register tells the address calculation hardware what extent of the memory, in X, the Y offset (used for down-scrolling) should be applied to. The number loaded to this register must be the number of read cycles set in the X scan configuration register, times 128. (For the VCB02: 8 X 128 = 1024).

[37] Y LIMIT (YL) -- Height of the portion of memory read by the screen refresh process (864 for the VCB02), plus the number of extra scans required when down-scrolling (between 16 and 32 is suggested). This register tells the address calculation hardware what extent of the memory, in Y, the Y offset (used for down-scrolling) should be applied to.

The Y offset is applied to all of the rectangular area in the memory defined by the corners: [0,0] and [XL-1,YL-1], inclusive. The memory in the rectangle from address [0, height of displayed screen (scans)] to (but not including) the address [X limit, Y limit] is not usable for any purpose. The memory between the right edge of the screen and X limit (if any) can be used, but only for Y addresses from 0 to the height of the displayed screen. The addresses referred to in this paragraph are in device coordinates.

[38] X SCAN COUNT 0 (XCT0)

[39] X SCAN COUNT 1 (XCT1)

[3A] X SCAN COUNT 2 (XCT2)

[3B] X SCAN COUNT 3 (XCT3)

[3C] X SCAN COUNT 4 (XCT4)

[3D] X SCAN COUNT 5 (XCT5)

[3E] X SCAN COUNT 6 (XCT6)

The X scan count registers are similar in function to the Y scan count registers. These seven X parameter registers program most horizontal timing for the VCB02 video subsystem. Each register determines the time of one horizontal event, such as the assertion of horizontal blanking. The event is specified by 3 high-order bits. The low 11 bits determine the time of the event (in 16ths of a major cycle, nominally 60 ns; but some events are not allowed to be programmed to this precision). This starts from a reference time preceding the beginning of the memory cycle that reads the first data to be refreshed on a scan (or the corresponding time during vertical retrace). The events are stored in the registers in order of increasing time, starting in XCT0. The highest time value is programmed as the horizontal period event. Bit <E> must be programmed in all seven registers, even if the "end horizontal period" event is in an earlier register. The bits of the X scan count registers are assigned as follows.

<0> 11 Bits. Time of event from a reference time 3 1/4 major cycles before the start of the first memory cycle on a scan, in 16ths of a major cycle. The start of a memory cycle is defined as the rising edge of PHI 2 during the RAS precharge that begins the cycle. No 2 events may occur in the same 1/4 of a major cycle; that is, no 2 events may have times that are equal in all but the low 2 bits.

<B> 3 Bits. Event.
000 = Set horizontal blank low.
001 = Set horizontal blank high.
010 = Set horizontal sync low except during vertical sync event.
011 = Set horizontal sync high except during vertical sync event.
100 = Set horizontal sync low.
101 = Set horizontal sync high.
110 = End horizontal period.
111 = Set sync request event.

<E> A 1 must be programmed for this bit in the XCT- register following the register that contains the "sync request" event. All other registers must contain a 0 in this bit.

<F> Test function in XCT6 (normally 0).

The period must be set to N X 16-4. The integer multiple N must be equal to twice the number of read cycles in XCON plus at least 2 (an integer number of major cycles). A larger N lengthens the horizontal retrace interval by units of 1 major cycle. N must be at least 10.

Horizontal blanking is normally set low shortly after the start of the first memory cycle of the scan to allow for the pipeline delay of data in the address processor and video processor chips and in the video output circuitry. The address processor chip produces the 3 1/4 major cycle delay mentioned above, and the video processor chip produces an additional 10 alpha clock delay (5/8 major cycle). Horizontal blanking is set high after the desired number of pixels are displayed on a scan.

The polarity of horizontal sync is programmable. Horizontal synchronization starts when set high by the 101 event (or low by the 100 event). Horizontal synchronization ends when set low by the 010 event (or high by the 011); except when vertical synchronization from the Y scan counter is high, when horizontal synchronization is set low by the 100 event (or high by the 101 event). The 100 or 101 events always change horizontal synchronization; one of these events should be set to occur after the 010 or 011 event when composite sync is generated.

System synchronization is generated by logic external to the address processor chip one major cycle after the synchronization request event occurs. Synchronization request always occurs during the scan preceding the end of vertical blanking. This event must be programmed in the range M X 32 + 4 to M X 32 + 16, inclusive. The integer multiple M must be in the range 0 to (N/2)-2, inclusive (integer truncation is implied), where N is defined for the period event above. The sync request must also be set for a time during the scan when no other X scan counter events occur during the following sync interval (two major cycles). Event times affected are in the range M X 32 + 21 to (M + 1) X 32 + 23, inclusive. The two low bits of the synchronization request event time are not significant (only 240 ns intervals allowed).

A sample assignment of events for a composite synchronization application is: XCT0 sets horizontal blanking low, XCT1 sets synchronization request, XCT2 sets horizontal synchronization low for vertical interval (100 event), XCT3 sets horizontal blanking high, XCT4 sets horizontal synchronization high, XCT5 sets horizontal synchronization low except during vertical interval (010 event), and XCT6 ends the horizontal period.

[3F] SYNC PHASE (SYNP) -- 11 Bits. The value loaded in the horizontal synchronization counter at each system synchronization time (once per frame). This must be set for (M + 1) X 32, where M is defined for the synchronization request event. (This assumes the required synchronization interval of two major cycles.) The low 5 bits of this register are ignored (not programmable).

3.4.1.2 **Address Processor Chip Commands** -- All commands to the hardware are passed to the two command registers in the address processor chip (except scrolling, which is initiated by loading the scroll constant register). The command register (CMD) is used for all screen update functions, and the I/D scroll command register (ICS) is used only for I/D bus register loads by the scroll process. The command register has 13 significant bits (the 3 MSBs are reserved for test functions), as shown in Figure 3-26.

The low byte contains the I/D bus (interconnect) portion of any command. It consists of an opcode nibble and a nibble that usually specifies a register address. The next two bits select one of four functions - I/D bus, processor/bitmap transfer (PTB or BTP), rasterop, or cancel - and the next three bits enable a combination of the destination and two sources. The ICS register is identical, except that it uses only the low byte to specify the I/D operation, and an I/D command is assumed.

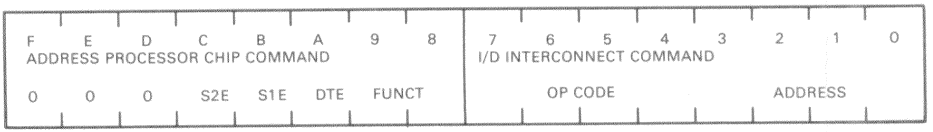


Figure 3-26 Command Register Format

Not all combinations of the command bits are meaningful; only the specific commands described in this section are allowed. Illegal commands yield undefined results; they are not NOPs.

All commands (except cancel) involve the video processor chips. Only those video processor chips selected by the chip select register respond to commands on their I/D bus (interconnect) inputs and/or allow writes to their associated memory planes. Participation in scrolling is determined by the setting of the scroll enable bit in each video processor chip. When multiple video processor chips are involved in an operation (such as a color rasterop or a register load common to many chips), all the corresponding chip selects are set. If only one video processor chip is to be active (as for a unique register load), only the one chip select is enabled. A separate chip select register is used for loading video processor chips when using the scroll I/D command register.

The chip select registers have an unencoded enable bit for each video processor chip in the system. The chip select registers are connected to the I/D bus (interconnect) and are loaded with an external I/D bus (interconnect) load.

If a rasterop command has been loaded and the initialization phase is complete (indicated by the initialization complete status bit), another identical command may be loaded, effectively requesting a repeat of the previous command. The command register is not actually double-buffered (only the reloading is remembered), so the previous command will not complete properly if a different command is loaded before the previous command finishes. (The address output must be complete before a new command is loaded.) The ability to queue a repeated command is used to reduce register loading overhead between rasterops of a series, such as when displaying a text string. PTB or BTP commands cannot be queued in this way because loading a new PTB or BTP command clears the IDD FIFO.

Register Loading

The loading of address processor chip registers does not require the execution of any command. The registers are loaded either directly, by use of the address pins, or indirectly, through the address counter. The interrupt and request pins, in conjunction with the status register, are used to determine the safe time to load a register.

All video processor chip registers and chip select registers are loaded using the address processor chip I/D command. Though any I/D interconnect instruction can be transmitted with this command, only these register loads are permitted. No reading or active cycles that are controlled by the address processor chip as part of rasterops are allowed. Figure 3-27 shows the I/D bus video processor register load command format.

The "address" in Figure 3-27 specifies the desired video processor chip register. Only those video processor chips whose addressed registers are to be changed should be chip selected. Figure 3-28 shows the I/D bus Z-axis video processor register load command format.

The "reg" in Figure 3-28 specifies the register to be loaded:

00 = Source register
01 = Foreground register
10 = Scroll fill register
11 = Background register.

"Z block" should be set to zero for the VCB02 video subsystem. Z block can specify one of the four possible groups of planes or subplanes whose values are specified by the 16-bit I/D bus data. All video processor chips pertinent to the Z-axis load should be chip selected. Figure 3-29 shows the I/D bus external register load command format (used for loading the chip select registers).

This instruction is ignored by all video processor chips, regardless of the setting of the chip select registers. Any data except 00H may be encoded in the seven low bits of the command for the external devices to decode. However, the address processor chip will always source data during the I/D interconnect data transfer cycles. The I/D bus (interconnect) instruction 00H is reserved for the address processor chip to transmit during idle I/D bus cycles.

The instructions above can be used through either the update (IDD and CMD) or scroll (IDS and ICS) ports. The corresponding data register must be loaded before one of the command registers is loaded. When the I/D data register is loaded (except during PTB or BTP commands), this data word can be transmitted on a following I/D command, regardless of the previous state of the IDD FIFO. Of course, the action of the IDS register is the same because it is only a single register. After the command is loaded to the appropriate register, the corresponding status flag should be detected (by interrupt, request, or poll) before a new data word or another command is loaded.

Figure 3-27 I/D Bus Video Processor Register Load Command Format

Figure 3-28 I/D Bus Z-Axis Video Processor Register Load Command Format

Figure 3-29 I/D Bus External Register Load Command Format

In order to avoid interference with update operations, only the scroll process should load the following registers in the address processor chip.

1. I/D SCROLL DATA (IDS)
2. I/D SCROLL COMMAND (ICS)
3. SCROLL X MIN (PXMN)
4. SCROLL X MAX (PXMX)
5. SCROLL Y MIN (PYMN)
6. SCROLL Y MAX (PYMX)
7. Y OFFSET (PYOF)
8. Y SCROLL CONSTANT (PYSC)
9. PAUSE (PSE), if used for scroll synchronization

The following address processor chip registers may be loaded by the scroll process with the necessary synchronization between the scroll and update processes.

1. PENDING X INDEX (PXI)
2. PENDING Y INDEX (PYI)
3. NEW X INDEX (NXI)
4. NEW Y INDEX (NYI)
5. OLD X INDEX (OXI)
6. OLD Y INDEX (OYI)

The scroll process should load only the following registers in the video processor chip.

1. SCROLL CONSTANT
2. FILL
3. LEFT SCROLL BOUNDARY
4. RIGHT SCROLL BOUNDARY

Rasterop Command

Prior to issuing a command to start a rasterop, all pertinent registers should have been loaded, including the mode register. During a rasterop, the appropriate status bits should be detected before any registers are changed. All functions are invoked by the command shown in Figure 3-30.

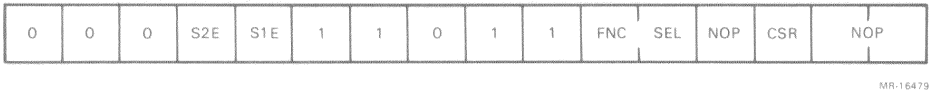


Figure 3-30 Rasterop Command Format

In Figure 3-30, the first two bits enable the first or second source, or both. (Source 1 should never be enabled in fill mode.) The next bit is always set high to enable the destination. (If it were disabled, no action would be taken by the hardware, except the setting of the status bits.) The I/D byte specifies the active cycle for the rasterop. FNC SEL is a 2-bit code that selects one of the four logic unit function registers in each video processor chip. "CSR" specifies the bank of CSRs to be referenced during the source and destination cycles.

The address processor chip fills in the last two CSR address bits according to the cycle in progress. The video processor chips participating in the rasterop are selected with the chip select register. There is no equivalent to a Z-axis form of a rasterop command.

Three status bits are used to monitor the progress of a rasterop command. Initialization complete indicates that the source 1 and destination origins may be loaded and the previous rasterop command may be requeued (by loading the CMD register again). This is useful to reduce wasted time between text characters. Rasterop complete indicates that rasterop parameters may be loaded. Address output complete indicates that any registers may be loaded. Normally, only one of these status bits is enabled at a time.

Processor-to-Bitmap or Bitmap-to-Processor Transfers

Processor-to-bitmap transfers (PTBs) or bitmap-to-processor transfers (BTPs) are similar to rasterops, except that either the source or the destination is the (MicroVAX CPU) memory. These transfers use the same rasterop status bits for control of the process, with the addition of two I/D data buffer bits.

All data for a PTB or BTP is transferred through the I/D data FIFO (IDD). This 6-deep FIFO is cleared when any PTB or BTP (or rasterop, or cancel) command is written to the command register. Data may then be written to the IDD address during a processor-to-bitmap (PTB) transfer, if the transmit status bit is set. Data may be read from this address during a bitmap-to-processor (BTP) transfer, if the receive status bit is set. A PTB or BTP command is complete when the address output complete status bit is set. All data will have been transmitted/read from the FIFO and any register can be loaded.

Scaling cannot be applied to a PTB or BTP because the address processor chip ignores the scale registers unless both the source 1 and destination are enabled. A PTB or BTP never enables both.

When PTB or BTP are used during scrolling, data must be transferred continuously to prevent computed addresses in the address processor chip from becoming invalid. The MicroVAX CPU or DMA controller (in the DMA gate array) must transfer two words every horizontal scan time.

X-Mode PTB and BTP

In an X-mode PTB, data is transferred from the processor (MicroVAX CPU) memory to 1 plane with 16 bits adjacent in X occupying one word, followed by additional X words, and then by additional scans. In an X-mode BTP, data from one plane is transferred to the processor memory with 16 bits adjacent in X occupying one word, followed by additional X words, and then by additional scans. The LSB of a word is the leftmost pixel in the displayed word; the MSB is the rightmost pixel. The area of the bitmap that is involved in a transfer of data to or from the processor memory is determined by setting the rasterop destination and source to span an unrotated rectangle. Figure 3-31 shows the processor-to-bitmap X-mode command format.

Figure 3-31 Processor-to-Bitmap X-Mode Command Format

In Figure 3-31, the FNC SEL and CSR bits have the same meaning as for rasterops, except that any CSR location can be used to control the register to which the processor data will be loaded. The chip select register controls the plane(s) to which the processor data is (are) being transmitted. The destination rasterop registers are programmed for the origin and size of the rectangle to be transferred. The fast DY and slow DX must both be zero.

The destination may start on any pixel, but no shifting of the processor data will occur. If the destination starts in the middle of a bitmap memory word, the bits in the first processor word on each fast vector that will be outside the destination rectangle are masked. Similar masking occurs on any trailing bits in the last word on each fast vector. Because there is no shifting available, PTB commands cannot be directed to any region that is scrolling horizontally. This can be avoided by transferring the data to an off-screen area, and then using a normal rasterop to transfer to a scrolling region. If the desired alignment of destination data is known at the time data is read from the bitmap with a BTP command, the correct shifting can be applied at that time. Figure 3-32 shows the bitmap-to-processor X-mode command format.

In Figure 3-32, the CSR bit has the same meaning as for rasterops, except that any source CSR location can be used to assign the video processor chip to the I/D interconnect. (Otherwise, if more than one video processor chip are enabled by the CSR, the transmitting chip may be selected by the chip select register.) The use of one of the source CSRs is required to obtain proper function of a delay register. The destination rasterop registers are programmed for the size of the rectangle to be transferred. The fast DY and slow DX must both be zero. The source registers are programmed for the origin of the data. The source DX and DY sign bits must be the same as those for the destination.

Shifting of the data is determined by the low four (or three, or two) bits of the destination origin. The first bit of the source is shifted to the bit addressed by the destination origin (just as in any rasterop). Like the masked bits on each fast vector of the PTB command, the corresponding bits (those not in the destination) are undefined in the BTP data words. BTP data may be read from a vertically scrolling region because normal indexing and offsetting apply to this source.

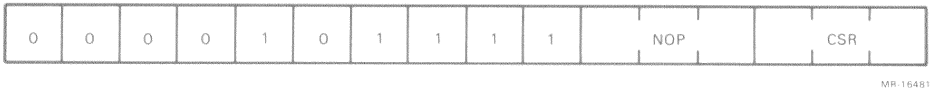


Figure 3-32 Bitmap-to-Processor X-Mode Command Format

Z-Mode PTB and BTP

In a Z-mode PTB or BTP, all the bits from one pixel location (one bit from each plane) are transferred from or to a processor memory word, followed by additional pixel data along the fast vector, and then by additional scans. The bitmap area is again determined by an unrotated rectangle addressed by the rasterop destination and source. Because the Z transfer occurs one pixel at a time, there are no restrictions on shifting or use during scrolling, other than the requirement of continuous data transfer. Figure 3-33 shows the processor-to-bitmap Z-mode command format.

In Figure 3-33, F/B is set to 0 to select the foreground register, and to 1 to select the background register. The source register cannot be used for PTBZ operations. The use of logic unit function register 10 is assumed in the PTBZ command, and is normally set to force 1 to select the foreground register, or 0 to select the background register. The planes being transmitted to are controlled by the plane address, the "Z block" address, and/or the chip select register. The destination rasterop registers are programmed for the origin and size of the rectangle to be transferred. The fast DY and slow DX should both be zero. The destination may start on any pixel. Figure 3-34 shows the bitmap-to-processor Z-mode command format.

The plane address, the "Z block" address, and/or the chip select register are used to select the video processor chips that form the Z word. The destination rasterop registers are programmed for the size of the rectangle to be transferred. The fast DY and slow DX should both be zero. The source registers are programmed for the origin of the data. The sign bits of the source DX and DY determine the direction of scanning.

Figure 3-33 Processor-to-Bitmap Z-Mode Command Format

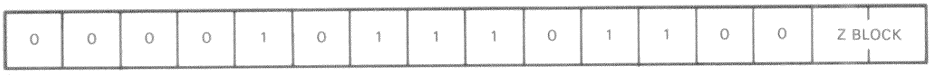


Figure 3-34 Bitmap-to-Processor Z-Mode Command Format

Cancel Command
The cancel command stops all operations, regardless of the command in progress, except for commands entered through the scroll I/D port. The rasterop status bits are set to their "completed" states, and the address processor output FIFO is cleared. The IDD FIFO is also cleared, and the ID status bits set accordingly. The pause, scroll service, I/D scroll data, clipping, and vertical blanking status bits are not affected. Figure 3-35 shows the cancel command format.

Figure 3-35 Cancel Command Format

During execution of the cancel command, the I/D data transmit ready status bit is held low. When this bit goes high, the cancel command is complete and any new data or command may be loaded. This bit should be tested before the next command is loaded, unless the software guarantees four major cycles of delay.

3.4.2 Video Processor Chip (I/D Bus) Commands

The instruction/data interconnect is used to control all the video processor chips and chip select registers. The I/D bus (interconnect) also provides for the exchange of data between video processor chips or between the video processor chips and the address processor chip. The chip select registers for the video processor chips control the video processor chips' participation in I/D bus transfers and rasterops. Two chip select registers are used in the VCB02 video subsystem: one for the scroll process to access the video processor chips, the other for the update process.

3.4.2.1 Video Processor Chip Registers -- The basic width of a video processor chip register is 16 bits. All the video processor chip registers are writable - none are readable. The register addresses (or the address of the first of a group of registers) are in hexadecimal and enclosed in brackets []. Where specified, bit assignments (or the LSB of a group of bits) are enclosed in angle brackets <>.

[0] RESOLUTION MODE -- 2 bits. Controls the actions of the mask bits (the combination of mask 1, mask 2, and the edge mask), the source register, the rasterop barrel shift constant, and Z-axis commands to make a video processor chip perform as 1, 2 or 4 planes. The VCB02 color maps cannot take advantage of resolution mode, so this feature is not supported. For normal VCB02 operation, 1-plane resolution must be selected.

<0> 00 = 1 Plane. Each mask/source bit controls its respective mask mux/logic unit bit independently of the others. All barrel shifter constant bits are significant. The video processor chip acts as a single plane for Z-axis operations, receiving or transmitting the one bit corresponding to its plane address. The plane address may be programmed to any value.

01 = 2 Planes. Even and odd mask/source bits are ORed (after the complementers controlled by the LUF registers), and the result is used to control both the corresponding bits of the mask mux/logic unit. The LSB of the rasterop barrel shift constant is truncated so that data in this plane moves only by a multiple of two bits. The video processor chip acts as two planes for Z-axis operations, receiving or transmitting the two bits corresponding to its plane address and the next most significant bit. The plane address must be programmed to an even value.

10 = Undefined results.

11 = 4 Planes. The four mask/source bits from each nibble are ORed (after the complementers controlled by the LUF registers), and the result is used to control all four corresponding bits of the mask mux/logic unit. The two LSBs of the rasterop barrel shift constant are truncated so that data in this plane moves only by a multiple of four bits. The video processor chip acts as four planes for Z-axis operations, receiving or transmitting the four bits corresponding to its plane address and the next three most significant bits. The plane address must be programmed to a multiple of four.

NOTE

The rasterop barrel shift constant must be truncated in low-resolution video processor chips to keep the bits in their proper subplanes. The address processor chip cannot supply each video processor chip with a different shift constant. (Each video processor chip may be programmed for a different resolution.) On the other hand, this effect cannot be provided for scrolling. The error caused by truncation must not be accumulated and the scrolling region may not be the region for which the resolution mode has been set. Horizontal scrolling should be only in increments of the lowest-resolution plane involved.

[1] BUS WIDTH -- 2 Bits (starting with <2>). Allows either 4, 8, or 16 bits of memory to be used, depending on the number of pixels required by the display. The VCB02 video subsystem is a 16-bit system and requires the bus width to be set to 16-bit mode. All video processor chips and the address processor must be set to the same bus width. The video bus speed for the 16-bit bus width is 4 bits every 60 ns.

<2> 00 = 4-bit bus width.
01 = 8-bit bus width.
10 = Undefined results.
11 = 16-bit bus width.

[2] SCROLL CONSTANT -- This register is double-buffered; data loaded becomes active at the beginning of the following frame.

<0> 4 Bits. X scroll constant. For left scroll, this nibble is the magnitude of the scroll, 0 to 15 pixels per frame time. For right scroll, this nibble is the magnitude of the scroll minus 1. The values 0 to 15 result in scrolls of 1 to 16 pixels per frame time, respectively. If the scroll disable bit is set in this register, or if a nonzero Y scroll constant is programmed in the address processor chip, this nibble must be 0. If the region being scrolled contains some video processor chips set for low-resolution, the low-order one or two bits of this nibble should be 0 for left scrolls, or all 1s for right scrolls.

<4> Horizontal scroll direction.

0 = Left. Also use 0 for up, down, and disabled scrolling. If a nonzero Y scroll constant is programmed in the address processor chip, this bit must be 0.
1 = Right.

<5> 0 = Disables scrolling (horizontal or vertical) of all data accessed by this video processor chip (disables writing for memory plane). Also, the other bits in this register must be set to down, left, and zero X scroll constant.
1 = Enable scrolling.

<6> Vertical scroll direction. This bit accounts for some asymmetries between up- and down-scrolling.

0 = Down scroll. Also used with disable scroll.
1 = Up, left, or right scroll.

[3] PLANE ADDRESS -- 6 Bits. Plane address for Z-axis operations. No two video processor chips should be given the same (or overlapping) plane address(es). If the fill register is loaded in Z-mode by the scroll process, the plane addresses must be established permanently. However, if the scroll process loads the fill registers individually, different update regions could have different plane address arrangements.

<0> 4 Bits. Bit address within Z-axis block (0 to 15). Addresses the bit (or low-order bit in a low-resolution video processor chip) on which the video processor chip will exchange data on the I/D bus during Z-mode transfers. This must be a multiple of two or four in a low-resolution video processor chip.

<4> 2 Bits. Z-axis block address (0 to 3). These are essentially the high-order bits of a 6-bit plane address (intended for use in systems with more than 16 planes), but separate blocks must be addresses with separate Z commands. For the VCB02, these bits should be 00.

[4] LOGIC UNIT FUNCTION -- 4 registers of 8 bits. Any one of the four registers can be selected for use during a rasterop or PTB command. During Z-mode processor-to-bitmap commands, function register 10 is used for modification of the bitmap.

<0> 4 Bits. Logic unit function. The video processor chip logic unit combines the word read from the destination (D) during a destination cycle (read-modify-write cycle) with the contents of the source register (S) (normally loaded during a previous source memory cycle). This occurs after the latter has passed through a complementer and resolution logic. The logic unit then uses the result (F) to select the foreground or background color.

```
0000 = F = ZEROs
0001 = F = NOT(D  OR      S)
0010 = F = NOT(D) AND     S
0011 = F = NOT(D)
0100 = F =     D  AND NOT(S)
0101 = F =            NOT(S)
0110 = F =     D  XOR     S
0111 = F = NOT(D  AND     S)
1000 = F =     D  AND     S
1001 = F = NOT(D  XOR     S)
1010 = F =                S
1011 = F = NOT(D) OR      S
1100 = F =     D
1101 = F =     D  OR  NOT(S)
1110 = F =     D  OR      S
1111 = F = ONEs
```

<4> 0 = Use mask 1.
1 = Use the complement of mask 1.

<5> 0 = Use mask 2.
1 = Use the complement of mask 2.

<6> 0 = Use the complement of source.
1 = Use source.

<7> 0 = Enable resolution mode logic for the source register.
1 = Disable resolution mode logic for the source register (pass bits through to the logic unit without combining adjacent bits, regardless of resolution mode register setting).

The following four registers have 16 bits. The LSB is the leftmost pixel in a word, the MSB is the rightmost pixel. All four registers can be loaded by I/D bus register load commands. The mask and source registers can also be loaded by data transfers during rasterop cycles. The source and fill registers can be loaded with constant data (solid color) by a Z-axis register load.

[8] MASK 1 -- Programmable mask register 1. Used to control which bits are modified by a read-modify-write cycle. Loading mask 1 loads mask 2 with the same data, so that only one mask is effective.

[9] MASK 2 -- Programmable mask register 2. Used to control which bits are modified by a read-modify-write cycle.

The outputs of the two mask registers are followed by independent complementers (controlled by the logic unit function registers). The outputs of the complementers are ANDed with each other and with the edge mask (which defines those bits involved in a rasterop). The only bits modified are those selected by all three masks. (A 1 from the AND enables modification of a bit.) Resolution mode logic follows the AND function. If only mask 1 is used, both complementers must be enabled or disabled together to prevent the masking of all bits.

[A] SOURCE -- Source word (normally loaded by a source memory cycle) for the logic unit to combine with the destination to select the foreground or background colors for each pixel. The source data is processed by a complementer and resolution mode logic before entering the logic unit. The Z-axis address for this register is 00.

[B] FILL -- Data to be inserted into memory words to fill the blank space created by scrolling. Normally, a solid color fill is desired, and is provided with a Z-axis load, if loaded directly. This register is double-buffered and data loaded becomes active on the following frame. The Z-axis address for this register is 10.

Because the scroll region boundaries can be set within a word, the video processor chips must be notified which bits of the leftmost word and the rightmost word are actually in the scroll region. The video processor chip can be programmed to place the region boundaries on any bit position (for possible future extension), but the address processor chip cannot specify the boundary closer than a multiple of four. Therefore, only those boundaries that select groups of four bits should be selected. These registers are double-buffered and data loaded becomes active on the following frame.

[C] LEFT SCROLL BOUNDARY -- 16 Bits. All bits are clear, corresponding to the pixels that are to be scrolled in the word that contains the left edge of the region. (All other bits are set.) Normally, this requires clearing all bits from the pixel on the edge through the MSB of the word; but if both edges are contained in one word, only bits from the left edge through the right edge are cleared.

[D] RIGHT SCROLL BOUNDARY -- 16 Bits. All bits are clear from the LSB (left edge) through the bit corresponding to the rightmost pixel that is to be scrolled in the word that contains the right edge of the region. (All other bits are set.) All bits are set unless the right boundary between words (LSB not scrolled) or both edges are contained in the same word.

[E] BACKGROUND COLOR -- Selected by a 0 at the output of the logic unit. The Z-axis address for this register is 11.

[F] FOREGROUND COLOR -- Selected by a 1 at the output of the logic unit. The Z-axis address for this register is 01.

The foreground and background registers have 16 bits. The LSB is the leftmost pixel in a word, the MSB is the rightmost pixel. Both registers can be loaded by I/D bus register load commands and by a Z-axis register load. The outputs of the logic unit select these registers on a bit-by-bit basis as the inputs to one side of the mask mux. (The mask mux then selects between this new data and the old destination data.) Either of these two registers is used to pass data during Z-mode PTB operations.

CONTROL STORE RAM -- 6 registers. The video processor chip control store RAM controls the transfer of data within the video processor chip, and to/from other I/D bus devices, during rasterops. Addressing of its contents during rasterops is controlled by the address processor chip and occurs once for each update memory cycle.

[10] CSR 0 -- Controls the first source read if bank 1 is selected and source 1 is enabled in the address processor chip command register.

[11] CSR 1 -- Controls the second source read if bank 1 is selected and source 2 is enabled in the address processor chip command register.

[12] CSR 2 -- Controls the destination read-modify-write if bank 1 is selected and the destination is enabled in the address processor chip command register.

[13] Reserved.

[14] CSR 4 -- Controls the first source read if bank 2 is selected and source 1 is enabled in the address processor chip command register.

[15] CSR 5 -- Controls the second source read if bank 2 is selected and source 2 is enabled in the address processor chip command register.

[16] CSR 6 -- Controls the destination read-modify-write if bank 2 is selected and the destination is enabled in the address processor chip command register.

[17] Reserved.

NOTE

These assignments of CSR functions are defined by the address processor chip. The video processor chip assumes no assignment of CSRs, except that CSR 0 and 4 are linked to the first delay register, and CSR 1 and 5 are linked to the second delay register.

Each of the CSR words has the following bit assignments.

<0> 2 Bits. External load. Selects the register into which incoming data from the I/D bus (if any) will be loaded following a memory read.

00 = None
01 = Source
10 = Mask 1 and mask 2
11 = Mask 2

<2> 2 Bits. Internal load. Selects the register into which incoming data from the local memory plane (from the barrel shifter) will be loaded following a memory read.

00 = None
01 = Source
10 = Mask 1 and mask 2
11 = Mask 2

<4> I/D bus (interconnect) output control. Only one plane should be enabled to output its shifted data on the I/D bus, for all other planes, during any one memory read cycle (that is, for any one CSR address).

0 = Disable output.
1 = Enable output.

<5> Barrel shifter delay control. When executing a fast mode rasterop, one output word is usually formed from a part of each of two source words. To avoid excessive reads to the source raster, the previous word can be held in a delay register so its remainder can be used with the next word. This bit controls one delay register when the active CSR is 0 or 4 (for source 1), and controls the other delay register when the active CSR is 1 or 5 (for source 2).

0 = Do not load delay register.
1 = Do load delay register.

NOTE

With two separate delay registers selected by the CSR address, the delay enable bit is no longer necessary. It should be programmed to a 1.

There may be no reason to ever program either destination CSR to other than 000000. These two registers may be unnecessary.

3.4.2.2 **Instruction/Data Bus Instructions** -- I/D bus instructions are specified by a 1-byte code. Rasterop, PTB, and BTP commands are further specified by a subinstruction byte. Most I/D interconnect instructions are also accompanied by a 16-bit data transfer.

I/D Instructions Issued by the MicroVAX CPU

The addressed video processor chip register of 16-bit data transmitted on the I/D interconnect is contained in the word loaded to the IDD (or IDS) register of the address processor chip (previous to loading an I/D bus command). This instruction is always initiated by the MicroVAX CPU. The address processor chip never transmits the instruction without an explicit command to do so, as shown in Figure 3-36.

Figure 3-37 shows the instruction format that loads all bits of the addressed register (in the video processor) with the contents of 1 of the 16 I/D bus data bits. The low four bits of the plane address programmed into each video processor chip selects the I/D data bit used during the load. Only those video processor chips in the addressed Z-block that are chip selected are updated. The Z-block should be 0 for the VCB02. If a video processor chip is set for a low-resolution mode, a pair of bits or a nibble is duplicated into each of the 8 pairs or 4 nibbles of the selected register. Z-axis register addresses are defined as follows.

00 = Source
01 = Foreground
10 = Fill
11 = Background

The VCB02 video subsystem uses instructions designed to control external I/D devices, which load the update and scroll chip select registers. The format of the external register load instructions is shown in Figure 3-38. I/D instructions are distinguished by bit 7 = 0 in the instruction byte. Instruction 00H is reserved for NOP.

Address Processor Chip I/D Instructions

The only direct I/D bus (interconnect) instructions a programmer will use are the video processor chip and chip select register loads. The remaining I/D interconnect instructions are used by the address processor chip to control rasterops.

The instruction shown in Figure 3-39 is transmitted by the address processor chip when the I/D bus is idling.

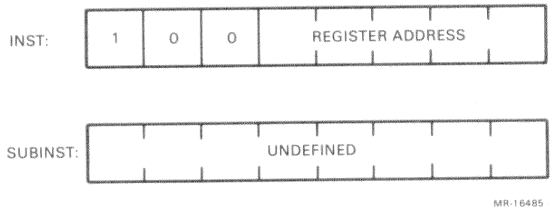


Figure 3-36 Video Processor Chip Register Load Instruction Format

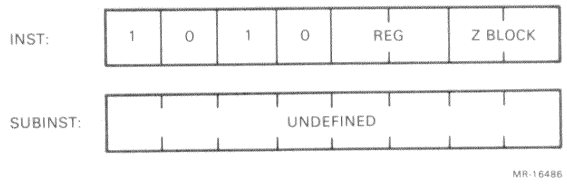


Figure 3-37 Video Processor Chip Z-Axis Register Load Instruction Format

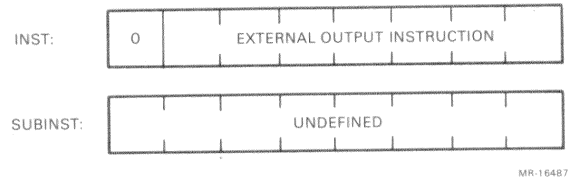


Figure 3-38 External Register Load Instruction Format

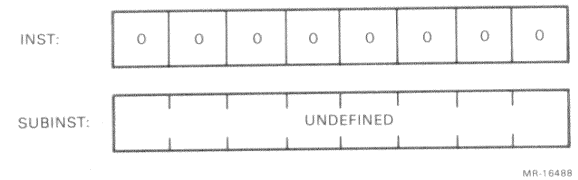


Figure 3-39 No Operation Instruction Format

The instruction shown in Figure 3-40 is essentially identical to the video processor chip Z-axis register load, except that the instruction is generated with an accompanying read-modify-write cycle (as part of a processor-to-bitmap transfer). Bit F/B is set to 0 to select the foreground register, and to 1 to select the background register. The source register cannot be used for PTBZ operations because the propagation delays in the video processor chip are too long for the path through the source register. The use of logic unit function register 10 is assumed in the PTBZ command. An edge mask is provided to select the one bit in the memory being written. However, the Z data being written by the current memory cycle is that which was transmitted during the previous I/D bus cycle (not the concurrent cycle). These are really ordinary Z-axis register loads interleaved with destination-only rasterop cycles. The right and left mask specifiers are equal for this instruction and are the address of the selected bit.

The address processor chip uses the instruction shown in Figure 3-41 during a bitmap-to-processor transfer, along with an accompanying memory read cycle. The bit address specifies the bit in the memory word for which the color is being read. The video processor chip generates a barrel shift constant from the bit address and the low four bits of the plane address. Each of the video processor chips addressed by the Z-block transmits its bit (or bits, if a low-resolution chip) on the appropriate I/D bus pin.

The instruction shown in Figure 3-42 is generated by the address processor chip with an accompanying memory read cycle to control the handling of the read data. The CSR address selects the controlling CSR location, which is either 0, 1, 4 or 5. The subinstruction controls the use of the barrel shifter to align the source data with the destination location. The barrel shifter has a 32-bit input and selects a 16-bit segment as an output. If the shift constant is 0, the leftmost 16 bits are selected. If the shift constant is 15, the rightmost bit of the left word and the left 15 bits of the right word are selected. The inputs to the left or right word of the barrel shifter can come either from the data read on the current cycle, or from the data previously stored in the delay register that corresponds to the addressed CSR. If the LD bit = 0, the current word is used for the left input. If the LD bit = 1, the delay register is used for the left input. The right input is selected similarly.

The instruction shown in Figure 3-43 is generated by the address processor chip with an accompanying read-modify-write cycle to control the logic unit register and the edge masks. The CSR address selects the controlling CSR location and is either 2 or 6. One of the four logic unit function and mask control registers is selected. The subinstruction sends the edge mask to select the bits to be written into the word. The left mask is the bit address of the leftmost pixel to be modified. The right mask is the bit address of the rightmost pixel to be modified. Therefore, if one pixel is to be modified, both mask values are the same. The LSB, or leftmost pixel, is 0. The MSB, or rightmost pixel, is 15.

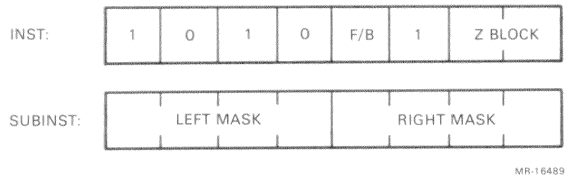


Figure 3-40 Z-Axis Write (Two Interleaved Instructions) Format

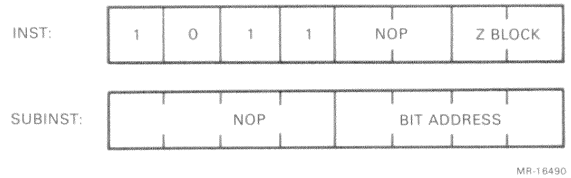


Figure 3-41 Z-Axis Read Instruction Format

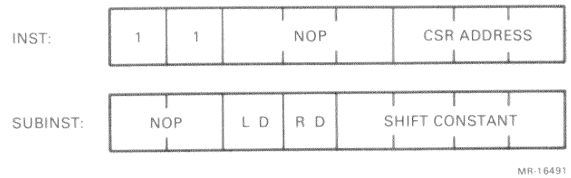


Figure 3-42 Active Cycle for Memory Read Instruction Format

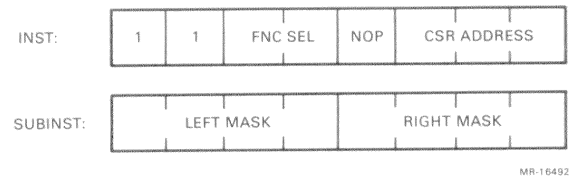


Figure 3-43 Active Cycle for Read-Modify-Write Instruction Format

3.4.3 Physical Configuration

This section describes system-wide interconnection, timing, and functional issues specific to the VCB02 video subsystem.

3.4.3.1 Address Processor Chip Pins -- The address processor chip is contained in an 84-pin, leaded, square, surface-mount package. Pin numbers are indicated in brackets []. Groups of pins are listed in order, starting with the high-order pin.

Processor Interface

- DAT<15:0> (Input/Output) [57, 58, 59, 60, 63, 64, 65, 66, 69, 70, 71, 72, 73, 74, 75, 76]

  DAT<15:00> are connected to PB<15:00> on the DMA gate array and are used for the read/write data transferred from/to any register in the address processor and video processors. DAT<15:00> are tri-stated to receive register data unless RD is high, and -AS and -DS are low.

- AD<5:0> (Input) [77, 78, 79, 80, 81, 82]

  AD<5:0> are connected to the AA<6:1> pins of the DMA gate array and select an address processor chip register. AD<5:0> are latched on the falling edge of -AS.

- -AS (Input) [84]

  The falling edge of -AS latches the address on AD<5:0> and provides the address processor chip select function. In the VCB02 video subsystem, -AS and -DS are both connected to the ADDAS L pin on the DMA gate array. ADDAS L is asserted by the DMA gate array only for address processor chip bus cycles. When data is transferred to the address processor's ADCT register, -AS increments the register pointer.

- -DS (Input) [83]

  The falling edge of -DS latches processor data. During a read, the low level enables DAT<15:0>, and the rising edge disables DAT<15:0> output buffers. -DS and -AS are both connected to the ADDAS L pin on the DMA gate array. ADDAS L is asserted by the DMA gate array only for address processor chip bus cycles.

- RD (Input) [1]

  RD is connected to the DMA gate array pin ADDWR L. When RD is asserted (high) a read cycle is selected. When RD is deasserted (low) a write cycle is selected. RD is latched by -DS.

- -REQ (Output) [11]

  -REQ is connected to the DMA gate array pin ADDRQ L. -REQ is asserted by the address processor chip when a bit in the request enable register matches the corresponding bit in the status register. ADDRQ L synchronizes the transfer of data by the DMA logic (in the DMA gate array) with the execution of instructions by the address processor.

- -INT (Output) [10]

  -INT is connected to the IRQ1 L pin on the DMA gate array. -INT is asserted by the address processor chip when a bit in the interrupt enable register matches the corresponding bit in the status register. IRQ1 L is used by the DMA gate array to request interrupt service from the MicroVAX CPU for the address processor.

- -INIT (Input) [2]

  -INIT is connected to the QBUS INIT signal. -INIT resets the address processor chip command processor, halts commands, and clears the address counter, IDD, and address output FIFOs. It also disables the I/D bus drivers on the address processor chip until the first occurrence of SYNC following deassertion of -INIT.

Memory Address and Video Processor Chip Interface

- MEMAD<10:0> (Output) [52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 40]

  Bitmap memory address bits; multiplexed row and column addresses for the bitmap. Addresses are clocked out by ADDCLK: one row and eight column addresses for a major cycle, or one row and one column address for a minor cycle.

- ADDCLK (Input) [25]

  Bitmap memory address clock. Nine cycles per major memory cycle; two cycles per minor memory cycle.

- 128/-16 (Output) [37]

  Bitmap memory cycle select. High selects a major cycle for a 128-bit read or write (in 16-bit mode). Low selects a pair of minor cycles for a 16-bit read or write (also 16-bit mode).

- R/-W (Output) [36]

  Bitmap memory read/-write. Selects read or write for major or minor cycles. If write is indicated for a minor cycle, external timing implements the required RMW sequence.

- -WE<3:0> (Output) [35, 34, 33, 32]

  Bitmap memory write enables; four write enables to disable writes to nibbles in memory words that are outside the scroll or clipping regions. -WE<0> controls the low-order nibble (leftmost on the screen). A set of values is provided for each column address output by the address processor chip. Low enables writing.

- FORCE (Output) [38]

  High indicates major cycles during a down-scrolling frame so that writing can be forced in all planes not participating in down-scrolling.

- SCROLL (Output) [39]

  Scroll enable. High indicates all bitmap bus words that are contained in a scrolling region, except for the rightmost word in the region. A state output is provided for each column address output by the address processor chip during a screen refresh major cycle.

- ADS (Input) [28]

  Address disable. When high, allows the address processor chip to complete its current memory cycles and then suspend all update (minor) memory cycle activity. ADS is always low in a VCB02 video subsystem so that the address processor never suspends memory cycles.

I/D Bus (Interconnect)

- ID<7:0> (Input/Output) [22, 21, 20, 19, 18, 16, 15, 14]

  I/D bus bit. The 8-bit bus always transmits during the two instruction cycles. It is tri-stated during the two data cycles unless a PTB or I/D command is being executed.

- IDCTL (Output) [12]

  Chip select control. IDCTL is used to select the outputs of the scroll or update chip select register for connection to the video processor CS input. Scroll chip select registers are selected when IDCTL is high, and update registers are selected when IDCTL is low. IDCTL is high during execution of commands in the ICS register.

Monitor Timing

- BLANK (Output) [7]

  Video composite blanking signal. High indicates blanking.

- CMPSYN (Output) [6]

  Video composite/horizontal synchronization. Composite synchronization; programmable polarity.

- VRTSYN (Output) [5]

  Video vertical synchronization. Not used in the VCB02 video subsystem; programmable polarity.

Clocks

- PHI1, PHI2 (Input) [53, 55]

  Phase 1, 2 clock inputs. Non-overlapping clocks; period is 1/4 major cycle.

- PHI3, PHI4 (Input) [26, 27]

  Phase 3, 4 clock inputs. Non-overlapping clocks; period is 1/8 major cycle.

- -SYN/REQ (Output) [8]

  System synchronization request. Low indicates that a system synchronization should be initiated by the external timing generator (in the following major cycle; reset to high state by the assertion of SYNC).

- SYNC (Input) [9]

  System synchronization. Issued by the external timing generator during the first part of a synchronization interval to reset the phase of address processor chip timing logic. Also, a number of double-buffered registers are updated. Note that this is not a chip reset pin. All clock signals are held frozen during the entire synchronization interval (two major cycles).

Power

- VDDL1, VDDL2, VDDL3, VDDI, VDDM, VDDC, VDDD [3, 23, 62, 17, 41, 54, 67]

  Power. +5 V power supply for logic, I/D bus, memory address bus, clocks, and data bus.

- GNDL1, GNDL2, GNDL3, GNDI, GNDM, GNDC, GNDD [4, 24, 61, 13, 42, 56, 68]

  Power return. Ground for logic, I/D bus, memory address bus, clocks, and data bus.

- VBB [31]

  VBB bypass.

- Spare [29, 30]

3.4.3.2 **Video Processor Chip Pins** -- The video processor chip is contained in a 68-pin, leaded, square, surface-mount package. Pin numbers are indicated in brackets []. Groups of pins are listed in order, starting with the high-order pin.

Bitmap Memory

- DIO<15:0> (Input/Output) [19, 18, 17, 16, 15, 14, 13, 12, 11, 8, 7, 6, 5, 4, 3, 2]

  Bitmap data input/output bus. The DIO<15:0> output drivers are tri-stated when R/-W is high.

- SCROLL (Input) [10]

  Scroll enable. High indicates all DIO<15:0> bus words that are contained in a scrolling region, except for the rightmost word in the region.

- PE (Output) [25]

  Plane enable. When high, this tri-state signal allows the bitmap memory to be written during a rasterop or scroll write cycle. This output driver is tri-stated when R/-W is high.

- 128/-16 (Input) [65]

  Major or minor cycle. When high, this signal activates internal logic that is associated with major cycles (128-bit process). When low, this signal activates internal logic that is associated with minor cycles (16-bit process).

- R/-W (Input) [64]

  Bitmap memory read/bitmap memory write. When high, all DIO<15:0> and PE output drivers are tri-stated and the video processor chip may receive bitmap memory data. When low, all DIO<15:0> and PE output drivers are enabled.

  When 128/-16 is low, the falling edge of R/-W is used to latch into the bitmap memory data on the DIO<15:0> lines.

- LTCLK (Input) [66]

  The latch clock is generated by the timing logic on the base module. When R/-W and 128/-16 are high, the falling edge of LTCLK is used to latch into the bitmap memory data on the DIO<15:0> lines.

  When R/-W is low and 128/-16 is high, the rising edge of LTCLK causes valid scrolled data to be shifted out on the DIO<15:0> lines. There are eight cycles of LTCLK per major cycle.

I/D Bus (Interconnect)

- ID<7:0> (Input/Output) [31, 32, 33, 35, 36, 38, 39, 46]

  Instruction/data input/output interconnect. Normally tri-stated to receive instructions or data, the ID<7:0> output drivers can be enabled during execution of Z-axis or active I/D instructions.

- CS (Input) [40]

  Chip select. While receiving an instruction, if ID<7> and CS are high, the arriving instruction is latched and executed. Otherwise, the arriving instruction is ignored.

Video Output

- VID<3:0> (Outputs) [56, 57, 58, 59]

  Video output. Screen refresh data is shifted out on these output lines on the rising edge of ALPHA.

Clocks

- ALPHA (Input) [55]

  Data on the VID<3:0> lines is shifted out on the rising edge of ALPHA; period is 1/16 of a major cycle.

- PHI1, PHI2 (Inputs) [50, 49]

  Non-overlapping clocks that determine overall timing and control of the video processor chip; period is 1/4 of a major cycle.

- SYNC (Input) [63]

  System synchronization. When asserted, synchronizes the phase of internal video processor chip states to the remainder of the VCB02 video subsystem. Also, a number of double-buffered registers are updated. Note that this is not a chip reset pin.

Power

- VDD 1, VDD 2 [53, 54]

  +5 V power supply.

- VSS <8:0> [Drivers: 1, 68, 67, 27, 28, 44, 45; Logic: 29, 30]

  Power return. Driver ground; logic ground.

- VBB, CAVITY [61, 60]

  Substrate bias. VBB must be connected to CAVITY.

High-Speed Timing

Clock Generation
The address processor and video processor chips require the following clocks.

- PHI1, PHI2 -- Basic system clocks for both the address processor and video processor chips; determine I/D bus timing.

- PHI3, PHI4 -- Serial arithmetic clocks for the address processor chip.

- ALPHA -- Video output clock for the video processor chip.

- ADDCLK -- Address output clock for the address processor chip.

- LTCLK -- Data clock for 128-bit cycles (major cycles) for the video processor chip.

Clock timing is in increments of 30 ns (nominal); however, the actual period for PHI 3 and 4 is 115.6 ns, rather than the nominal 120 ns. All other clocks scale accordingly.

The skew between any two of the above clocks does not exceed +/- 5 ns. This is achieved by deskewing all these clocks through latches clocked at 30 ns.

ADDCLK has nine cycles per major cycle or two per minor cycle. LTCLK has eight cycles for every major cycle or pair of minor cycles. Though LTCLK does not clock data at the pins during minor cycles, it is still used internally.

System Synchronization
All system timers are synchronized to each other once every frame (vertical scan of the screen). Synchronization occurs at a time when no screen refresh or other user-perceivable activity is in progress.

Synchronization is accomplished by freezing all clocks and memory timing signals for two major cycles (one compute cycle in the address processor chip). This is called the synchronization interval. During the first part of the synchronization interval (the first major cycle) the SYNC pins on the address processor and all video processor chips are asserted. The clocks must be stopped at the start of any major cycle (rising edge of every other PHI2) that is permitted by the programming of the X SCAN COUNT registers. During the synchronization interval, the clocks must remain in the following states.

- PHI1 - Low
- PHI2 - High
- PHI3 - High
- PHI4 - Low
- ALPHA - High
- ADDCLK - High
- LTCLK - High
- Other memory timing signals as required

After the synchronization interval, all clocks must proceed exactly as if the two major cycles of the synchronization interval had never occurred. Basically, the only clocks that do not freeze are the ones that are needed to time the synchronization interval.

The synchronization interval is initiated when the address processor chip issues the SYN/REQ signal. The exact timing of this signal is programmed in the X scan count registers. The clock generating logic responds to SYN/REQ by starting a synchronization interval at the next major cycle boundary.

Processor bus activity can proceed as normal during the synchronization interval.

I/D Bus (Interconnect)
The instruction/data interconnect is tied to the address processor and video processor chips and to the chip select registers.

The I/D bus transmits two bytes of a command from the address processor chip, and transfers two bytes of data from the address processor or video processor chips every minor cycle.

Transmissions are synchronized to memory cycles, as required. A byte is transmitted during each of the four PHI1 and PHI2 pulses in a minor cycle. Each of the two PHI1 and two PHI2 pulses are labeled A and B. The rising edge of PHI2B starts a minor cycle, and the sequence is PHI2B, PHI1B, PHI2A, PHI1A. The command/data sequence is as follows.

1. PHI2A - Instruction 1
2. PHI1A - Subinstruction 1
3. PHI2B - Low byte data 0
4. PHI1B - High byte data 0

5. PHI2A - Instruction 2
6. PHI1A - Subinstruction 2
7. PHI2B - Low byte data 1
8. PHI1B - High byte data 1

9. PHI2A - Instruction 3
10. PHI1A - Subinstruction 3
11. PHI2B - Low byte data 2
12. PHI1B - High byte data 2

13. Etc.

Notice that the data for instruction 1 follows instruction 2. This provides for certain pipelining requirements of source rasterop memory cycles.

External I/D Bus (Interconnect) Devices
In addition to the address processor and video processor chips, the chip select registers are attached to the I/D bus. These devices are write-only because the address processor chip always transmits data during an I/D interconnect command. External I/D commands are distinguished from video processor chip commands by a zero in bit 7 of the instruction byte. The subinstruction is undefined during external I/D instructions, and so, is not of use to an external device. The external register numbers for the chip select registers are

- Update Chip Select = 60L
- Scroll Chip Select = 40L.

The logic that controls the loading of the chip select registers decodes the instruction byte that is present only during PHI2A cycles. The logic synchronizes to the PHI1/PHI2 A/B cycles by using SYNC, which always occurs during a PHI2B cycle.

Chip Selects
Each video processor chip has a chip select pin that controls whether or not it responds to I/D bus instructions, and through the PE pin and write enable logic, whether it allows writing in its associated memory during rasterops. An unencoded pin from a chip select register is provided for every video processor chip in a VCB02 video subsystem. This allows any combination of video processor chips to be enabled. Chip selects are sensed by video processor chips only during the instruction cycle (PHI2A).

Because the scroll process needs to access the video processor chips without disturbing the update process, it requires a separate set of chip selects. The scroll chip selects are provided by a register connected to the I/D bus.

Memory Address Bus

The bitmap address bus is common to all bitmap planes. Two groups of pins on the address processor chip control the address bus to the bitmap memory:

1. The MEMAD, -WE, and SCROLL pins are clocked by the ADDCLK pin. The MEMAD pins provide one row address and one column address for each 16-bit (minor) cycle, and one row and eight column addresses for each 128-bit (major) cycle. A valid set of write enables and a scroll are provided for each column address. The -WE pins are meaningful only during write cycles. The SCROLL pin is meaningful only for 128-bit read cycles. ADDCLK has a high interval that includes the first 30 ns of the PHI2B that starts a major cycle (or pair of minor cycles). The rising edge of the pulse that includes this interval brings the first column address (or only column address for a minor cycle) from the address processor chip. The previous pulse fetches the row address, and the following pulses (for a major cycle) fetch additional column addresses. There are nine cycles of ADDCLK in a major cycle and two cycles of ADDCLK in a minor cycle. ADDCLK is the same for reads as for writes.

   The series of column addresses in a major cycle form a sequential up-count. The MEMADs are latched externally to the address processor chip (on the 4-plane module) to deskew the propagation delay of the address processor chip, in order to obtain tight memory timing. This latch is also clocked by ADDCLK, so that the last column address of the previous cycle is going to the memory while the row address of the next cycle is coming from the address processor chip. Similarly, the row address is applied to the memory while the first column address is fetched from the chip.

   The scroll pin indicates memory words that are contained in the scroll region. It comes out of the address processor chip with the address of the associated word and is latched along with the MEMADs. However, it is accepted by the video processor chip as if it were data coming from the memory; therefore, it is delayed further so that it matches the data timing.

   NOTE

   The scroll pin from the address processor chip does not connect directly to the scroll pin of the video processor chip.

2. The 128/-16, R/-W, and FORCE pins are provided by the address processor chip on the PHI2A that precedes the PHI2B that starts each major cycle (or pair of minor cycles). These pins warn the clock generator of the type of cycle that should be generated next. A 16-bit write cycle means that a 16-bit read-modify-write cycle (containing both read and write portions) is next.

NOTE

The 128/-16, R/-W pins from the address processor chip do not connect directly to the 128/-16, R/-W pins of the video processor chip.

The force pin indicates major cycles in a down-scrolling frame.

Memory Address Bit Assignments

The sequence of address outputs at the MEMAD<10:0> pins of the address processor chip is a function of the bus width and memory configuration bits in the X scan configuration register of the address processor chip. The X and Y address bits are multiplexed out at row and column times for the memory in such a way that memory refresh is guaranteed by keeping the fastest moving X and Y addresses in the low eight row addresses. Table 3-1 lists the address bits, numbered according to their significance in address calculations. The lowest four X address bits (not shown in Table 3-1) access pixels within a memory word in the 16-bit bus width mode used by the VCB02 video subsystem. The VCB02 does not use address bit 10.

These addresses are mapped with an external multiplexer to generate the row and column addresses for the memories as shown in Table 3-2. Note that 11 bits of Y are used for a total of 2048 scan lines of memory. The low 4 bits of X (X0 through X3) are not used since data is accessed 16 bits at a time. X4 is not used in the address bits supplied to the memories, but is used to multiplex between two banks of memory. Five bits of X are supplied to the memories as an address. This results in 1024 pixels on a scan line (16 X 2 X 32 = 1024). The VCB02 memory is organized as 1024 (horizontal) X 2048 (vertical) pixels.

Memory Refresh

Memory refresh is accomplished by the screen refresh process. For each horizontal scan, 8 unique row addresses are generated. To generate a complete sequence of 256 row addresses, 32 scans must occur. The VCB02 displays 864 scans, which is a multiple of 32, so there are no incomplete sequences of 256 row addresses at the bottom of the screen display. In the VCB02, a complete 256-address sequence is finished in less than 0.6 ms. This is far less than the 4 ms required by the dynamic memory. However, no refresh is accomplished during vertical retrace. Therefore, the worst-case time to complete a refresh (1.263 μs) is the refresh period (593 μs) plus the vertical retrace time (36 scan lines = 667 μs).

Table 3-1 Multiplexed MEMAD<10:0> Bits

MEMAD Bit	10	9	8	7	6	5	4	3	2	1	0
Row Time	X12	Y11	X10	X9	Y4	Y3	Y2	Y1	Y0	X8	X7
Column Time	Y12	Y10	X11	Y5	Y9	Y8	Y7	Y6	X6	X5	X4

Note: Greater than 1024 mode; 16-bit bus width.

Table 3-2 Memory Address Bits

Address Bit	7	6	5	4	3	2	1	0
Row Address	X9	Y4	Y3	Y2	Y1	Y0	X8	X7
Column Address	Y10	Y5	Y9	Y8	Y7	Y6	X6	X5

Memory Configuration

The VCB02 video subsystem was designed to work with 64 K X 4 K dynamic RAMs (with page mode access). The data rate during major cycles requires approximately 90 ns cycles. Two banks of RAM are multiplexed to achieve this data rate.

Since the VCB02 has two screens of memory per plane, 64 K X 4 RAMs are arranged in even and odd banks (of 4 chips each), with successive major cycle words accessing alternate banks (selected by address bit X 4).

Memory Data Bus

The memory data lines for each plane are connected to the video processor chip data lines through a transceiver with integral transparent latch. The latch is held open to pass data directly to or from the memory during minor (16-bit) cycles, with the direction controlled by R/-W. The latch is used during major (128-bit) cycles to extend the hold time of memory data. The latch is clocked by LTCLK.

The type of memory cycle in the video processor chip is controlled by the 128/-16 and R/-W pins. These pins are changed at the time of the PHI2B clock that starts each major or minor cycle. If the minor cycle is a read-modify-write (signaled from the address processor chip as a 16-bit write), the R/-W pin initially indicates a read, and, at the appropriate time in the cycle, changes to a write to latch data and reverse the bus.

NOTE

The 128/-16, R/-W pins from the address processor chip do not connect directly to the 128/-16, R/-W pins of the video processor chip.

The memory data bus is controlled differently for major and minor cycles.

1. Major cycles -- LTCLK controls the clocking of data. During 128-bit reads, data is latched in the video processor chip on the falling edge of LTCLK. During 128-bit writes, data propagates from the video processor chip on the rising edge of LTCLK. The timing of LTCLK is different for 128-bit reads and writes. During 16-bit cycles, LTCLK continues to run, using the 128-bit read cycle timing.

   The scroll signal is applied to the video processor along with each corresponding memory word during 128-bit reads.

2. Minor cycles -- During a 16-bit read, data is latched in the video processor chip on the falling edge of the PHI2A in the middle of the cycle. During a read-modify-write, data is latched by R/-W falling or PHI2A falling, whichever is earlier. The falling edge of R/-W causes the video processor chip to reverse its direction and enable write data.

Write Enable Circuits

The clipping and scrolling logic use write enables of the bitmap memories to selectively write memory words and parts of words. In a VCB02 video subsystem, the write enables are arranged in groups of four adjacent bits, separately for each plane. This allows clipping/scrolling to boundaries at each multiple of four bits horizontally, and also allows each plane to be independently enabled for rasterops or scrolling. These groups of write enable pins are controlled by a logical combination of signals from four sources.

1. The four -WE pins from the address processor chip provide the write enables for each nibble of each 16-bit word associated with each column address. The -WE signals are latched externally on ADDCLK, similar to the MEMAD signals. The write enables are further delayed to match required memory timing. These signals provide the 4-bit resolution in the clipping and scrolling boundaries, and are used by the write enable logic for each plane.

2. The PE pin from each video processor chip is used for the write enable logic of only the associated plane to enable or disable rasterops or scrolling for that plane. For rasterops, the PE pin responds to the state of chip select present during each I/D instruction of the rasterop. It follows the associated bit of the update (not scroll) chip select register. For scrolling, the PE pin is controlled by the scroll enable bit in the video processor chip scroll constant register.

3. The force pin from the address processor chip is delayed until the rising edge of PHI2B starts a major (or minor) cycle, and is then combined in the write enable logic of all planes. This pin is asserted during major cycles if down-scrolling is active anywhere in the frame. This signal is used to force writing in planes where scrolling is disabled (because all data, rather than no data, must actually move in these planes during down-scrolling). Thus, writing is disabled in any plane where PE is not asserted, unless both FORCE is asserted and PE is not asserted.

4. The write enable logic also uses high-speed clocks to generate the high-resolution edges in the final write enable waveforms.

Video Bus and Video Output Circuits
The video bus from the video processor chip is clocked by ALPHA and provides data on every clock. The data is deskewed by a short set-up time shift register on the next 60 ns edge. Ultimately, the 4-bit video data is shifted in a serial stream through a color map and digital-to-analog converter, and sent to the video monitor.

The BLANK and CMPSYN signals from the address processor chip are precision outputs that provide the address processor composite video blanking and composite synchronization of the monitor. These signals are generated with 60 ns resolution and are normally deskewed with the video data. Because the blanking output is only programmable in 60 ns increments, external pipelining is used to ensure the blanking signal matches the higher-resolution video signal, so that the correct pixels are blanked and unblanked.

3.4.3.3 **Initialization** -- The INIT pin resets the operational aspects of the chip: stops rasterops, clears FIFOs, resets status bits, and clears the interrupt enable registers. This is different from the system SYNC pin, which resets timers and counters.

3.5 DMA GATE ARRAY

The DMA gate array provides a complete DMA Q22-Bus interface, address processor display list processing (using the template RAM), hardware cursor support, and I/O decode (including interrupt handling).

The DMA gate array (DC7035) is a semi-custom-integrated circuit constructed using double-level metal HCMOS gate array technology, and packaged in a ceramic 120 PGA (pin grid array). The DMA gate array was designed to provide a specialized DMA engine that interfaces the Q22-Bus to the address processor chip, and assists in the execution of video display lists and the transfer of bitmap data.

A display list is merely a sequence of commands and data that effect the desired video operation when executed by the address processor chip. By using the DMA engine provided by the DMA gate array, the MicroVAX CPU need only create the display list and initiate the DMA process. This frees the MicroVAX CPU from the burden of transferring the data to the address processor chip and monitoring the execution of the display list. Display lists transferred and executed by the DMA gate array are referred to as "DMA display lists."

The gate array provides dedicated address and control signals for the address processor chip. The data is transferred from/to the Q22-Bus through the gate array to/from a second 16-bit data bus called the "private bus." The private bus is shared by the DMA gate array, the address processor chip, and an external RAM called the "template RAM."

The template RAM is partitioned into three segments. The first 64 words are reserved as a FIFO for the DMA process. The last 32 words are used to store cursor bitmap data. The remainder of the template RAM is used to store display list routines, particularly those routines common to many applications.

Display lists stored in the template RAM are referred to as "template display lists." Special commands have been designed to allow template display lists to be called from DMA display lists. Commands have also been provided that allow template display lists to fetch commands and data from the DMA display list data stream (from the FIFO). As a result, routines stored in the template RAM can serve as command "templates" that fetch variable data from the FIFO. Using template display list routines minimizes the number of commands that must be transferred in a DMA display list. Fewer bus cycles are required to transfer the reduced DMA display list; therefore, overall system performance is improved. The DMA gate array provides dedicated address and control signals for the template RAM.

In addition to the DMA engine, the DMA gate array includes an interrupt controller, address decoding logic, and video cursor logic. The interrupt controller provides masks, generates a variable interrupt vector (the vector base is programmable), and resolves interrupt priorities for three interrupt sources. One interrupt comes from the internal DMA logic. The interrupt outputs of the address processor chip and the communications controller chip on the VCB02 base module are connected to two interrupt input pins provided by the DMA gate array.

The address decoding logic provides various chip enables for a 56-Kbyte memory address block. Enable signals are provided for external ROM, the template RAM, the address processor chip, and an external address decoder. The base of the 56-Kbyte address block is programmable to any 64-Kbyte boundary in the Q22-Bus memory address space.

The cursor logic generates a 2-plane, 16 X 16 pixel, bitmap cursor. Cursor data is read from the template RAM during the horizontal sync interval. The start address of the cursor is programmable to any pixel location and may be positioned off the active screen refresh area. The gate array outputs the cursor data at the appropriate time, every fourth VCB02 video pixel clock cycle. The data is output as two 4-bit words.

The following sections explain the operation of the DMA gate array.

3.5.1 Address Decoding

A power reset disables the address decoder in the DMA gate array. To enable the address decoder, a value must be written to the I/O page control and status register (CSR) and decoded external to the gate array by logic on the base module. When this address is detected, the external decoder asserts the CONSEL L signal pin on the gate array. If the cycle is a write cycle, the DMA gate array writes the bus data into its memory base register, asserts the signal pin RPLYO H, and enables its address decoding logic. If the cycle is a read cycle, the gate array asserts RPLYO H, and the contents of the external I/O page CSR are output. (The gate array does not drive the data lines.)

The value written into the memory base register selects the starting address, or base address, of a 56-Kbyte block of addresses to be decoded by the DMA gate array. The base address is always on a 64-Kbyte boundary in the Q22-Bus memory address space. All locations in this 56-Kbyte block are word-addressable only.

The DMA gate array segments the 56-Kbyte block of addresses and generates enable signals, as shown in Figure 3-44. The gate array asserts RPLYO H during any cycle that accesses any location in the 56-Kbyte address block. All addresses can be expressed as the sum of the Q22-Bus memory base, VCB02 memory base, and offset (the offset within the selected 56-Kbyte block). The Q22-Bus memory base is the system address of the first location in the 4-Mbyte Q22-Bus memory address space. The VCB02 memory base is the offset of the first location of the VCB02 memory address space (within the Q22-Bus memory address space). The VCB02 memory base is selected by the value written to the memory base register.

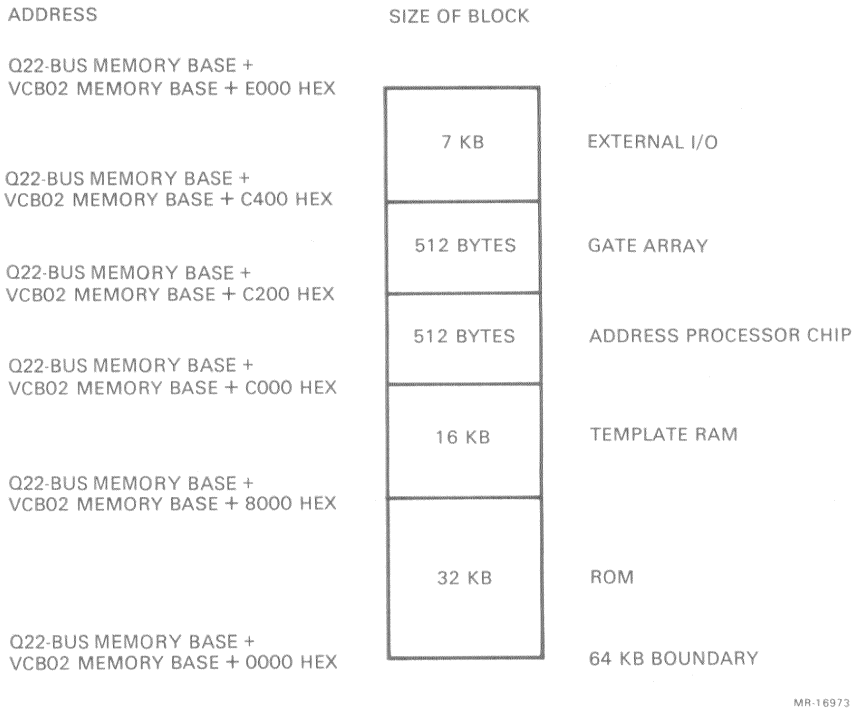


Figure 3-44 DMA Address Map

3.5.1.1 **ROMENB Decode** -- When any location in the first 32 Kbytes (offset 0000 hex to 7FFF hex) of the 56-Kbyte address block is accessed, the DMA gate array asserts the signal pin ROMENB L. ROMENB L is used as an enable for the external console emulation/diagnostic ROM on the base module. The DMA gate array does not drive the bus data lines when ROMENB L is decoded.

3.5.1.2 **RAMOE Decode** -- The DMA gate array asserts the signal pin RAMOE L when a Q22-Bus cycle accesses any location in the next 16 Kbytes (offset 8000 hex to BFFF hex) of the 56-Kbyte address block. RAMOE L enables the template RAM for access. The gate array completes the Q22-Bus cycle by performing the desired operation on the template RAM. Q22-Bus slave accesses of the template RAM are allowed at any time, and temporarily halt any display list execution that may be in progress. Execution of the display list resumes when the slave cycle is completed.

3.5.1.3 **Address Processor Chip Addresses** -- The next 256 word locations (512 bytes, offset C000 hex to C1FF hex) are reserved for address processor chip registers. If one of the address processor chip locations is being accessed, the gate array completes the Q22-Bus cycle by performing the desired operation on the address processor chip. The DMA gate array temporarily halts the execution of display lists to allow Q22-Bus slave accesses at any time. Execution of the display list resumes when the slave cycle is completed.

3.5.1.4 **Gate Array Addresses** -- The next 256 word locations (512 bytes, offset C200 hex to C3FF hex) are reserved for registers internal to the DMA gate array. Q22-Bus address bits AD09 H through AD01 H are decoded to select registers, as shown in Table 3-3. Currently, only nine register addresses are utilized.

Table 3-3 Gate Array Address Decoding

Q22-Bus Address Bit									
09	08	07	06	05	04	03	02	01	Register Selected
X	X	X	X	X	0	0	0	0	0: Control and status register
X	X	X	X	X	0	0	0	1	1: DMA address counter (15:00)
X	X	X	X	X	0	0	1	0	2: DMA address counter (21:16)
X	X	X	X	X	0	0	1	1	3: DMA byte counter (15:00)
X	X	X	X	X	0	1	0	0	4: DMA byte counter (21:16)
X	X	X	X	X	0	1	0	1	5: FIFO register
x	x	x	x	x	0	1	1	0	6: Cursor X position register
X	X	X	X	X	0	1	1	1	7: Cursor Y position register
X	X	X	X	X	1	0	0	0	8: Interrupt register

X = Don't care

3.5.1.5 **IOENB Decode** -- If a location in the next 7 Kbytes (offset C400 hex to DFFF hex) of the 56-Kbyte address block is accessed, the DMA gate array asserts the signal pin IOENB L. IOENB L is used as an enable for an external decoder on the base module that segments the 7-Kbyte address block to provide register space for the communications controller, the memory page control and status register, and the color maps (red, green and blue). The DMA gate array does not drive the data bus lines when IOENB L is decoded.

3.5.2 **DMA Engine**

The DMA logic provided by the DMA gate array can perform three basic types of DMA transfers:

1. Processor-to-bitmap transfers (PTB)
2. Bitmap-to-processor transfers (BTP)
3. Display list transfers.

The byte enable bit (08), DMA mode 0 bit (09), and DMA mode 1 bit (10) in the control and status register (CSR) select the type of DMA transfer to be performed. These bits should not be modified unless the DMA byte counter is zero and the FIFO is empty.

For each DMA process, the MicroVAX CPU must load the Q22-Bus start address of the data and the number of bytes to be transferred into the 22-bit DMA address counter and the 22-bit DMA byte counter, respectively. Two registers are provided for each counter.

To load the DMA address counter, the low word (16 bits) must first be written to register 1 (DMA address counter <15:00>), then, the remaining 6 bits must be written to register 2 (DMA address counter <21:16>). Similarly, to load the DMA byte counter, the low word (16 bits) must first be written to register 3 (DMA address counter <15:00>), then, the remaining 6 bits must be written to register 4 (DMA byte counter <21:16>). Loading the DMA byte counter starts the DMA process; it should, therefore, be loaded last. Once the DMA process is initialized, the DMA gate array completes the selected DMA operation without any assistance from the MicroVAX CPU.

The control and status register DMA done bit (15), DMA error bit (07), slave parity error bit (06), and bus timeout bit (05) provide status information on the DMA process. (Refer to Paragraph 3.5.6.1 for more details.) If CSR bit 01 (DMA interrupt enable) is set, the DMA gate array interrupts the MicroVAX CPU if any of the DMA status bits are set. (One of the bits must change from a 0 to a 1 to generate the interrupt.)

Whenever the DMA gate array is the bus master and the slave device is a non-block-mode device, the gate array transfers a minimum of 4 words (unless the DMA byte counter is less than 4). During the fourth word transfer the gate array checks the Q22-Bus DMA request signal (signal pin DMRI H). If no other device is requesting bus mastership (DMRI H is not asserted), the gate array continues transferring data up to maximum of 8 words. When the DMA gate array is the bus master and the slave device supports block-mode transfers, the gate array does a minimum of 8 word transfers (unless the DMA transfer counter register is less than 8). If no other device is requesting bus mastership during the eighth transfer, a maximum of 16 word transfers are performed.

Each byte or word transferred via DMA by the DMA gate array passes through a 64-word FIFO. The FIFO registers are actually the first 64 word locations of the template RAM. The gate array maintains two pointers into the RAM that control the loading and unloading process. The MicroVAX CPU can access the FIFO by addressing register 5 (FIFO register). When the MicroVAX CPU accesses the FIFO register, the appropriate pointer (read or write) is incremented and used as the address for the template RAM cycle. FIFO data passes through the DMA gate array to the Q22-Bus. The MicroVAX CPU can determine how many words are in the FIFO by reading register 2 (DMA address counter <21:16>). Bits 08 through

13 of register 2 indicate the number of words in the FIFO (0 = empty, 63 = full). The MicroVAX CPU can access the FIFO locations directly by addressing the first 64 words in the template RAM. Accessing the FIFO by addressing the template RAM does not affect the FIFO pointers. The MicroVAX CPU should not access the FIFO register or modify any of the FIFO locations by addressing the template RAM unless there is no active DMA process.

The MicroVAX CPU can be used to load/unload the FIFO instead of the DMA engine provided by the DMA gate array. Once the DMA mode is selected, the DMA gate array transfers the data between the address processor chip and the FIFO identically, independent of the loading or unloading method (MicroVAX CPU or DMA).

If the address processor chip is the DMA destination, the DMA gate array transfers a word out of the FIFO whenever one or more locations are filled, provided the address processor chip's DMA request signal (signal pin ADDRQ L) is asserted. If the address processor chip is the DMA source and has data ready (signal pin ADDRQ L is asserted), the gate array transfers the data to the FIFO whenever at least two locations are free.

If bitmap-to-processor byte mode is selected, DMA data can be "byte packed" as it passes through the DMA gate array. Byte packing takes the low bytes of two consecutive words in the FIFO and "packs" them into one word for transfer to the Q22-Bus destination. The low byte of the first word read from the FIFO becomes the low byte of the packed word. The low byte of the second word read from the FIFO becomes the high byte of the packed word. The packed word is then transferred to the Q22-Bus destination. If the MicroVAX CPU is used to unload the FIFO instead of the DMA engine, it must perform the byte packing task. Paragraph 3.5.2.2 describes the intended use of byte packing.

If processor-to-bitmap byte mode is selected, DMA data can be byte unpacked as it passes through the DMA gate array. Byte unpacking takes the low byte of a word read from the Q22-Bus and "unpacks" it into two words for transfer to the FIFO. The low byte of the Q22-Bus word is written to the low byte of the first FIFO location. The high byte of the Q22-Bus word is written to the low byte of the next FIFO location. If the MicroVAX CPU is used to load the FIFO instead of the DMA engine, it must perform the byte unpacking task. Paragraph 3.5.2.1 describes the intended use of byte unpacking.

When the Q22-Bus is the DMA source and byte unpacking is disabled, the DMA gate array initiates a DMA transfer only when there are at least 17 free locations in the FIFO. When the Q22-Bus is the DMA destination and byte packing is disabled, the DMA gate array initiates a DMA transfer whenever there are at least 16 FIFO locations containing data (except at the end of the DMA process, when the DMA byte counter is less than 32). Twice as many FIFO locations are used for each Q22-Bus word transferred when byte unpacking is enabled.

When the Q22-Bus is the DMA source and byte unpacking is enabled, the DMA gate array initiates a DMA transfer only when there are at least 33 free locations in the FIFO. When the Q22-Bus is the DMA destination and byte packing is enabled, the DMA gate array initiates a DMA transfer whenever there are at least 32 FIFO locations containing data (except at the end of the DMA process, when the DMA byte counter is less than 64). Twice as many FIFO locations are used for each Q22-Bus word transferred when byte packing is enabled.

3.5.2.1 **Processor-to-Bitmap Transfers** -- Processor-to-bitmap DMA mode is selected by setting the DMA mode 1 bit (10) and resetting the DMA mode 0 bit (09) of the control and status register (CSR). The Q22-Bus is the DMA source and the address processor chip is the DMA destination during processor-to-bitmap transfers. All words transferred are data words written to the address processor chip IDD register (address processor chip register 7). The two types of processor-to-bitmap transfers are single-plane and Z-axis transfers.

A single-plane processor-to-bitmap transfer moves data from Q22-Bus memory to one bitmap plane (the video memory associated with one video processor chip). Each word contains 16 bits that are adjacent horizontally. (The end of a horizontal line is adjacent to the start of the next horizontal line.) The LSB (bit 0) of the word is the leftmost pixel in the display, the MSB (bit 15) is the rightmost pixel.

A Z-axis processor-to-bitmap transfer moves data from the Q22-Bus memory to a single pixel location. (Each plane receives one bit of the data.) Since the VCB02 supports a maximum of eight planes, two words can be byte packed into one word to reduce the amount of Q22-Bus memory required to store a copy of a multiplane bitmap image. The gate array can unpack the bytes as the data is read from the Q22-Bus. Each byte of the word is stored in separate word locations in the FIFO. The low byte is written to the FIFO first. The high byte is written to the low byte of the next word location in the FIFO. Subsequently, the DMA gate array transfers the unpacked data from the FIFO to the address processor chip. By using byte unpacking, the number of Q22-Bus transfers can be reduced, resulting in higher system performance. Byte unpacking is enabled by setting the byte enable bit (08) in the control and status register.

3.5.2.2 **Bitmap-to-Processor Transfers** -- Bitmap-to-processor DMA mode is selected by resetting the DMA mode 1 bit (10) and setting the DMA mode 0 bit (09) of the control and status register (CSR). The Q22-Bus is the DMA destination and the address processor chip is the DMA source during bitmap-to-processor transfers. All words transferred are data words read from the address processor chip IDD register (address processor chip register 7). The two types of bitmap-to-processor transfers are single-plane and Z-axis transfers. All data read from the address processor chip is loaded into the FIFO directly. The DMA gate array transfers the data from the FIFO to the selected Q22-Bus location.

A single-plane bitmap-to-processor transfer moves data from one bitmap plane (the video memory associated with one video processor chip) to Q22-Bus memory. Each word contains 16 bits that are adjacent horizontally. (The end of a horizontal line is adjacent to the start of the next horizontal line.) The LSB (bit 0) of the word is the leftmost pixel in the display, the MSB (bit 15) is the rightmost pixel.

A Z-axis bitmap-to-processor transfer moves data from a single pixel location. (Each plane contributes one bit of the data.) Since the VCB02 supports a maximum of eight planes, two words can be byte packed into one word to reduce the amount of Q22-Bus memory required to store a copy of a multiplane bitmap image. The byte packing is performed as the data is read from the FIFO for transfer to the Q22-Bus. Byte packing takes the low byte of the first word read from the FIFO and saves it in the low byte of a holding register. The low byte of the second word read from the FIFO is written to the high byte of the holding register. The byte packed data in the holding register is then transferred to the Q22-Bus. By using byte packing, the number of transfers on the Q22-Bus can be reduced, resulting in higher system performance. Byte unpacking is enabled by setting the byte enable bit (08) in the control and status register.

3.5.2.3 **Display List Transfers** -- Display list DMA mode is selected by resetting the DMA mode 1 bit (10) and setting the DMA mode 0 bit (09) in the control and status register (CSR). The Q22-Bus is the DMA source and the address processor chip is the DMA destination during display list transfers. Both commands and data may be contained in the words read from the Q22-Bus. The Q22-Bus DMA logic in the DMA gate array transfers words from the Q22-Bus to the FIFO (the first 64 words of the template RAM).

The transfer of data and commands from the FIFO to the address processor chip is controlled by the display list processor logic in the gate array. As the words are read from the FIFO, the display list processor conditionally decodes each word. If the word is decoded as a display list assist command, the display list processor executes the command as specified in Paragraph 3.5.5. Some display list assist commands are transferred to the address processor chip, others are not. All words read from the FIFO that are not recognized as display list assist commands are transferred to the address processor chip. Command and data encoding are detailed in Paragraph 3.5.5.

3.5.3 Interrupt Controller

The DMA gate array provides a 3-bit interrupt controller. One interrupt is connected internally to the DMA logic, the other two are connected to the input pins IRQ1 L and IRQ2 L. On the VCB02 base module, IRQ1 L is connected to the address processor chip's interrupt output, and IRQ2 L is connected to the communications controller interrupt output. When set, the external interrupt enable bit (02) of the control and status register (CSR) masks the external interrupts. When set, the DMA interrupt enable bit (01) in the CSR masks the internal DMA interrupt. When the mask is set the interrupt(s) is(are) ignored. If any of the interrupts are asserted but not being masked, the gate array asserts IRQ L (interrupt output). An external interrupt is asserted by generating a high-to-low transition on the input pin and maintaining the low level until the interrupt is acknowledged. The DMA interrupt is asserted if the slave device reports a memory parity error (CSR bit 06), or a bus timeout condition occurs (CSR bit 05). The DMA interrupt is also asserted when the DMA done bit (CSR bit 15) is set.

During a Q22-Bus interrupt acknowledge cycle, the gate array latches the three interrupts when DIN H is asserted. If none of the interrupts is asserted, the gate array asserts IAKO H when IAKI H is asserted. If a nonmasked interrupt is asserted when IAKI H is received, the gate array outputs a priority encoded interrupt vector, asserts RPLYO H, and clears the highest priority interrupt. Bits 00, 01, and 09 through 15 of the interrupt vector are always 0 during an interrupt acknowledge cycle. Bits 04 through 08 of the vector are programmable by writing to the interrupt register. Bits 02 and 03 of the vector are encoded according to the highest priority interrupt asserted, and unmasked as shown in Table 3-4.

The interrupt register is readable. While the vector register is being read, the interrupts are latched and are read on bits 12 through 15 of the vector register. (Refer to Paragraph 3.5.6.9.)

Table 3-4 Interrupt Register Vector Bits 02 and 03

Priority	Interrupt	Vector Bit 03	Vector Bit 02
Highest	DMA Interrupt	0	0
Second	IRQ1 L	0	1
Lowest	IRQ2 L	1	0
	NONE	1	1

3.5.4 Cursor Logic

The cursor logic generates a 2-plane (A and B), 16 X 16 pixel, bitmap cursor for use by external video output logic. The cursor data can be enabled for output at any pixel location, and may be positioned on- or off-screen (that is, on = active video, off = video blanking). When set, the cursor enable bit (00) in the control and status register (CSR) enables the output of cursor data. The position is selected by the values written to the cursor X position register and cursor Y position register. The X position register and Y position register should only be modified during the vertical sync period, or when the cursor enable bit (00) in the CSR is reset (cursor is disabled).

The Y position register is loaded into an up counter (requires 2's complement data) on the second horizontal sync pulse during the vertical retrace period. (BLANK H remains asserted for at least two horizontal sync periods.) The Y counter is incremented by the trailing edge of BLANK H (the start of each active video line). The contents of the Y counter are checked on the leading edge of CSYNC H (during the horizontal retrace period preceding the next scan line). If the cursor enable bit (00) in the CSR is set, and all except the four least significant bits of the Y counter are set, the cursor is enabled for output on the new display line, and two cursor data words (16 bits) are read from the template RAM.

The last 32 locations of the template RAM are reserved for cursor bitmap data. The first 16 of these words are cursor "A" data, the last 16 words are cursor "B" data. The least significant 4 bits of the Y register select which cursor "A" and cursor "B" word to read. As the Y counter is incremented by succeeding scan lines, the address of the cursor A and cursor B data words increments. (As the monitor is scanned from top to bottom, the cursor data is output from lowest address to highest address.) The reading of cursor data has the highest cycle priority. (Display list execution, Q22-Bus DMA, Q22-Bus slave cycles, and display list execution are all lower in priority.)

When the cursor data is enabled for output on a display line, the X counter determines the position on the line that the data is output. Bits 02 through 10 of the X position register are loaded into an up counter (requires 2's complement data) by horizontal sync pulses (present on CSYNC L). The X counter is incremented (effectively by 4 since the X counter begins with bit 02) by CURCLK H (cursor clock). CURCLK H is one-fourth the VCB02 video pixel clock frequency.

The cursor data is output when all bits of the X counter are set. (Bits 02 through 10 are all 1s.) The least significant two bits of the cursor X position register shift the cursor data so that it is pixel-aligned. (Refer to Paragraph 3.5.6.7 for details). The cursor data must be shifted up to speed by shift registers external to the gate array. The shift registers should be loaded synchronously on the leading edge of CURCLK H.

3.5.5 Display List Data and Commands

The commands described in the following section provide flexibility in the use of the template RAM as a local store RAM for routines that can be called from DMA display lists (display lists transferred from the Q22-Bus by the DMA gate array). In addition to commands that start and stop execution of template display lists (routines stored in the template RAM), commands are included that allow data and commands stored in the FIFO to be used by template display lists. One of the commands allows processor-to-bitmap transfers to be imbedded in a DMA display list. Each command is described in detail in the following sections.

3.5.5.1 Display List Data -- All words in a display list that have bit 14 and 15 reset, as shown in Figure 3-45, are considered to be display list data by the gate array and are transferred to the address processor chip's ADCT register (register 0). Bits 0 through 13 are data bits and may have any value. The address processor chip writes the word to the register being selected by its ADCT register. Display list data can be used in template display lists and DMA display lists.

3.5.5.2 JMPT @ ADDRESS Command -- The JMPT @ ADDRESS command has bits 15 and 13 reset, and bit 14 set, as shown in Figure 3-46. When executed by the display list processor, the JMPT @ ADDRESS command loads the word address (A01 through 13) contained in bits 0 through 12 into an address counter in the gate array called the template address counter. If the address bits A07 through A13 are reset, the display list processor resumes execution of the display list contained in the FIFO. If any of the address bits A07 through A13 are set, the display list processor executes the template display list at the specified address. This command provides the ability to "call" a template display list from a DMA Display list, to "jump" within template display lists, and to end the execution of the template display list and return to execution of the DMA display list in the FIFO. The JMPT @ ADDRESS command is not transferred to the address processor chip.

3.5.5.3 PTB NWORDS Command -- The PTB NWORDS command has bit 15 reset, and bits 14 and 13 set, as shown in Figure 3-47. Bits 00 through 12 (N00 through N12) are a 2's complement count value. When executed by the display list processor, the PTB NWORDS command loads the count into a counter in the gate array. The command is not transferred to the address processor chip. The gate array transfers the next N words (N is the count value) from the FIFO directly to the address processor chip's IDD register (register 7). Display list execution resumes from the FIFO (even if the PTB NWORDS command was executed from a template display list). The PTB NWORDS command allows processor-to-bitmap transfers of up to 8 K words to be imbedded in a DMA display list. However, the data must not be byte packed.

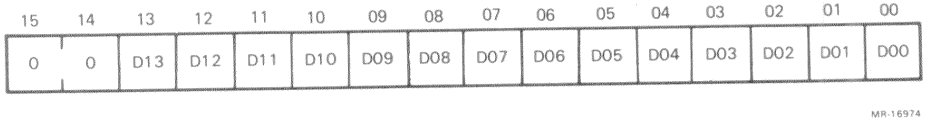


Figure 3-45 Display List Data

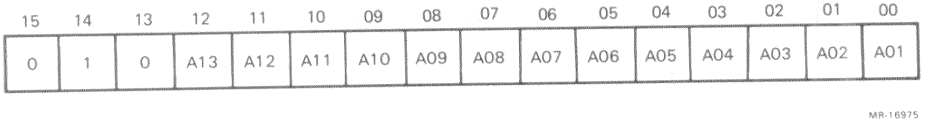


Figure 3-46 JMPT @ ADDRESS (Jump to Template @ Address)

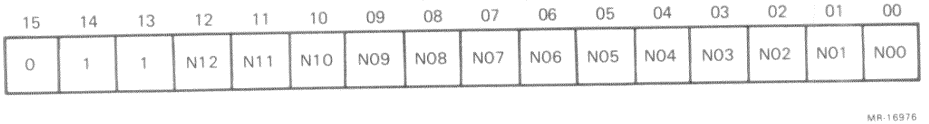


Figure 3-47 PTB NWORDS (Processor-to-Bitmap Transfer Number of Words)

3.5.5.4 Other Display List Commands -- All other display list commands recognized by the DMA gate array are encoded in bits 12 through 15 of the word, as shown in Figure 3-48. Bits 00 through 11 of the word may be used as data for loading the address processor chip's ADCT register. (The address processor chip only uses bits 00 through 05. Bits 06 through 11 are reserved.) All commands must have bit 15 set. All commands that have bit 14 cleared are transferred to the address processor chip, causing the ADCT register to be loaded (bit 15 set). When bit 13 of the command word is set, the display list processor logic fetches the next word from the FIFO. (The word may be data or a command.) Bit 12 disables the command decoding of the next word fetched by the display list processor logic (allows 16-bit data to use bits 11 through 15). Though many combinations of bits 12 through 15 are anticipated to be useful in the execution of display lists, some combinations of these bits may not be.

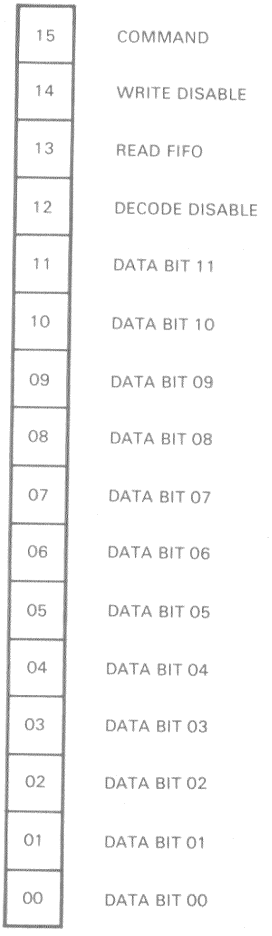


Figure 3-48 Other Display List Commands

3.5.6 Register Descriptions

The following paragraphs describe the usage of each bit of all registers in the DMA gate array accessible from the Q22-Bus. All registers are word-accessible only.

3.5.6.1 Control and Status Register (Register 0) -- The control and status register (CSR) provides various control and status bits for the various functions the DMA gate array provides. Each bit is described in Figure 3-49.

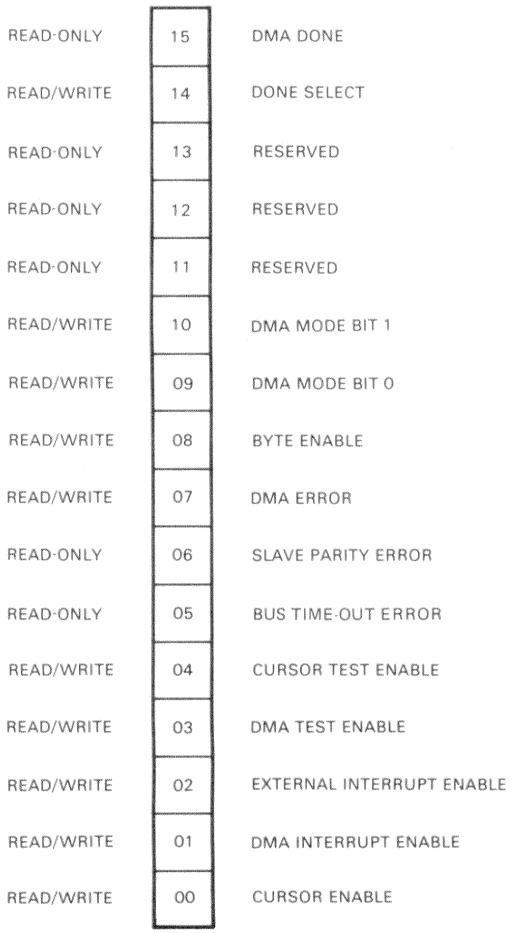


Figure 3-49 CSR Register Bitmap

Bit 15: DMA Done (Read-Only)

The DMA done bit (15) is reset whenever the DMA byte counter is not 0; it is also reset whenever the done select bit (14) is set and the FIFO is not empty.

The DMA done bit (15) is set whenever the DMA byte counter is 0 and the FIFO is empty; it is also set whenever the done select bit (14) is reset and the DMA byte counter is 0.

Asserting the signal pin QINIT H sets the DMA done bit (15). Writing the DMA done bit (15) has no effect.

Bit 14: Done Select (Read/Write)

The done select bit (14) selects the conditions that enable the DMA done bit (15) to be set. Writing a 0 to the done select bit (14) enables the DMA done bit (15) to be set whenever the DMA byte counter is 0. The DMA byte counter must be 0 and the FIFO must be empty to set the DMA done bit (15) if a 1 is written to the done select bit (14). See the description for CSR bit 15 above. Asserting the signal pin QINIT H resets the DMA select bit (14).

Bit 13: Reserved (Read-Only)

CSR bit 13 is reserved for future use. Writing bit 13 has no effect. When the CSR is read, bit 13 is always 0.

Bit 12: Reserved (Read-Only)

CSR bit 12 is reserved for future use. Writing bit 12 has no effect. When the CSR is read, bit 12 is always 0.

Bit 11: Reserved (Read-Only)

CSR bit 11 is reserved for future use. Writing bit 11 has no effect. When the CSR is read, bit 11 is always 0.

BIT 10: DMA Mode Bit 1 (Read/Write)

CSR bits 8, 9, and 10 control the operation mode of the DMA engine. Table 3-5 shows the bit settings for the various DMA functions. Asserting the signal pin QINIT H resets bit 10. The DMA mode 1 bit (10) should not be modified unless the DMA byte counter is 0 and the FIFO is empty.

Bit 09: DMA Mode Bit 0 (Read/Write)

CSR bits 8, 9, and 10 control the operation mode of the DMA engine. Table 3-5 shows the bit settings for the various DMA functions. Asserting the signal pin QINIT H resets bit 9. The DMA mode 0 bit (09) should not be modified unless the DMA byte counter is 0 and the FIFO is empty.

Table 3-5 CSR Bits 8, 9, and 10

CSR 10	CSR 09	CSR 08	DMA Function
0	0	X	Halt
0	1	X	Display List
1	0	0	Bitmap-to-Processor (word)
1	0	1	Bitmap-to-Processor (byte packed)
1	1	0	Processor-to-Bitmap (word)
1	1	1	Processor-to-Bitmap (byte unpacked)

X = Don't care.

Bit 08: Byte Enable (Read/Write)

CSR bits 8, 9, and 10 control the operation of the DMA engine. (Refer to Table 3-5.) The byte transfer enable bit (08) is effective during processor-to-bitmap (PTB) and bitmap-to-processor (BTP) transfers only. Setting the byte enable bit (08) enables byte packing/unpacking during PTB/BTP transfers. (Refer to Paragraphs 3.5.2.1 and 3.5.2.2.) The byte enable bit (08) should not be modified unless the DMA byte counter is 0 and the FIFO is empty.

CSR bit 10: DMA Mode 1

CSR bit 09: DMA Mode 0

CSR bit 08: Byte Enable

Bit 07: DMA Error (Read/Write)

Writing a 0 to CSR bit 07 has no effect. Writing a 1 to CSR bit 07 resets the DMA error bit (07), the slave parity error bit (06), and the bus timeout error bit (05).

The DMA error bit (07) is reset when the slave parity error bit (06) and the bus timeout error bit (05) are both reset. Asserting the signal pin QINIT H resets the DMA error bit (07).

Bit 06: Slave Parity Error (Read-Only)

If a slave parity error occurs while the DMA gate array is the bus master, the slave parity error bit (06) and the DMA error bit (07) are set. Writing a 1 to CSR bit 07 or asserting the signal pin QINIT H clears the slave parity error bit (06). Writing CSR bit 06 has no effect.

Bit 05: Bus Timeout Error (Read-Only)

If a bus timeout error occurs while the DMA gate array is the bus master, the bus timeout error bit (05) and the DMA error bit (07) are set. Writing a 1 to the CSR bit 07 or asserting the signal pin QINIT H clears the bus timeout error bit (05).

Bit 04: Cursor Test Enable (Read/Write)

Writing a 1 to the cursor test enable bit (04) enables a special test mode for the cursor logic in the DMA gate array. This test mode is intended for use in device testing only. For normal operation, the cursor test enable bit (04) must be reset.

Bit 03: DMA Test Enable (Read/Write)

Writing a 1 to the DMA test enable bit (03) enables a special test mode for the DMA logic in the DMA gate array. This test mode is intended for use in device testing only. For normal operation, the DMA test enable bit (03) must be reset.

Bit 02: External Interrupt Enable (Read/Write)

Writing a 1 to the external interrupt enable bit (02) enables the signal pins IRQ1 L and IRQ2 L to generate an interrupt. Writing a 0 to the external interrupt enable bit (02) disables all external interrupts. Asserting the signal pin QINIT H resets this bit.

Bit 01: DMA Interrupt Enable (Read/Write)

Writing a 1 to the DMA interrupt enable bit (01) enables the DMA done bit (15), or the slave parity error bit (06), or the bus timeout error bit (05), to generate an interrupt. Writing a 0 to the DMA interrupt enable bit (01) disables all DMA interrupts. Asserting the signal pin QINIT H resets this bit.

Bit 00: Cursor Enable (Read/Write)

Writing a 1 to the cursor enable bit (00) enables the cursor logic to output cursor data. Writing a 0 to the cursor enable bit disables the output of cursor data. This bit should be modified only during the vertical retrace period. Asserting the signal pin QINIT H resets this bit.

3.5.6.2 DMA Address Counter (15:00) (Register 1) -- The DMA address counter (15:00) is the low word of the 22-bit DMA address counter provided by the DMA gate array. The low 16 bits of the Q22-Bus DMA source/destination address are loaded into this register. If an odd value is written to the DMA address counter, the first DMA transfer is a byte transfer and the counter is incremented by 1. If an even value is written to the DMA address counter, the first transfer is a word and the counter is incremented by 2. With the exception of the last transfer, all subsequent DMA transfers are word transfers and the counter is incremented by 2.

If the DMA byte counter is even for the last transfer (byte count = 2), the transfer is a word transfer and the DMA address counter is incremented by 2. If the DMA byte counter is odd for the last transfer (byte count = 1), the transfer is a byte and the DMA address counter is incremented by 1. The DMA address counter (15:00) overflows into the DMA address counter (21:16) when it is at maximum count and is incremented. Whenever the DMA address counter (15:00) is written, the DMA address counter (21:16) must subsequently be written to maintain valid data. The counter should not be written unless the DMA byte counter is 0 (the Q22-Bus DMA process is complete). The DMA address counter (15:00) is readable. Figure 3-50 shows the DMA address counter register 1 bitmap.

3.5.6.3 DMA Address Counter (21:16) (Register 2) -- The DMA address counter (21:16) is the high 6 bits of the 22-bit DMA address counter provided by the DMA gate array. The high 6 bits of the Q22-Bus DMA source/destination address are loaded into this register. The counter is incremented when the DMA address counter (15:00) is at maximum count and is incremented. Whenever the DMA address counter (15:00) is written, the DMA address counter (21:16) must subsequently be written to maintain valid data. The counter should not be written unless the DMA byte counter is 0 (the Q22-Bus DMA process is complete). The DMA address counter (21:16) is not readable. Bits 07 through 15 of this register have no effect when written. Figure 3-51 shows the DMA address counter register 2 bitmap.

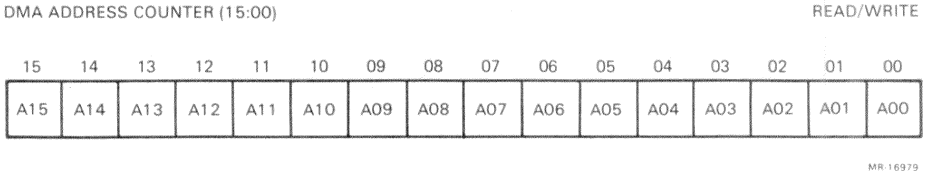


Figure 3-50 DMA Address Counter Register 1 Bitmap

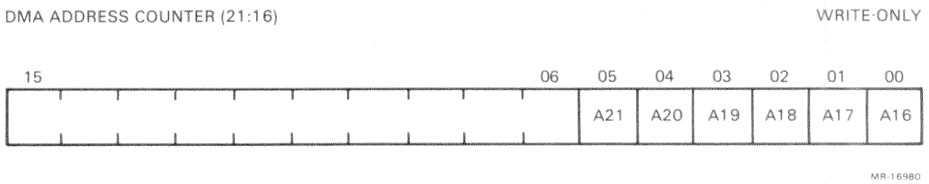


Figure 3-51 DMA Address Counter Register 2 Bitmap

3.5.6.4 DMA Byte Counter (15:00) (Register 3) -- The DMA byte counter (15:00) is the low word of the 22-bit DMA byte counter provided by the DMA gate array. The low 16 bits of the Q22-Bus DMA byte count are loaded into this register. If an odd value is written to the DMA address counter, the first DMA transfer is a byte transfer and the byte counter is incremented by 1. If an even value is written to the DMA address counter, the first transfer is a word and the byte counter is incremented by 2. With the exception of the last transfer, all subsequent DMA transfers are word transfers and the counter is incremented by 2.

If the DMA byte counter is even for the last transfer (byte count = 2), the transfer is a word transfer and the counter is incremented by 2. If the DMA byte counter is odd for the last transfer (byte count = 1), the transfer is a byte transfer and the counter is incremented by 1. The DMA byte counter (15:00) 3-111 overflows into the DMA byte counter (21:16) when it is at maximum count and is incremented. Whenever the DMA byte counter (15:00) is written, the DMA byte counter (21:16) must subsequently be written to maintain valid data. The counter should not be written unless the DMA byte counter is 0 (the Q22-Bus DMA process is complete). The DMA byte counter (15:00) is readable. Figure 3-52 shows the DMA byte counter register 3 bitmap.

3.5.6.5 DMA Byte Counter (21:16) (Register 4) -- The DMA byte counter (21:16) is the high 6 bits of the 22-bit DMA byte counter provided by the DMA gate array. The high 6 bits of the Q22-Bus DMA byte count are loaded into this register. The counter is incremented when the DMA byte counter (15:00) is at maximum count and is incremented. Whenever the DMA byte counter (15:00) is written, the DMA byte counter (21:16) must subsequently be written to maintain valid data. The counter should not be written unless the DMA byte counter is 0 (the Q22-Bus DMA process is complete). The DMA byte counter (15:00) is readable. Figure 3-53 shows the DMA byte counter register 4 bitmap.

When read, bits 08 through 13 contain the FIFO count (number of words in the FIFO). If the FIFO count is 0, the FIFO is empty. If the FIFO count is 63, the FIFO is full. Reading the DMA byte counter (21:16) temporarily halts the execution of display lists. Display list execution resumes when the Q22-Bus read cycle is completed. Bits 07 through 15 of this register have no effect when written.

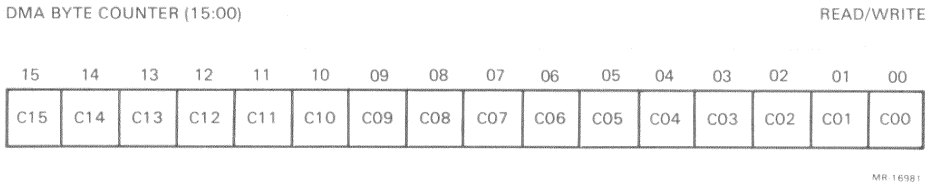


Figure 3-52 DMA Byte Counter Register 3 Bitmap

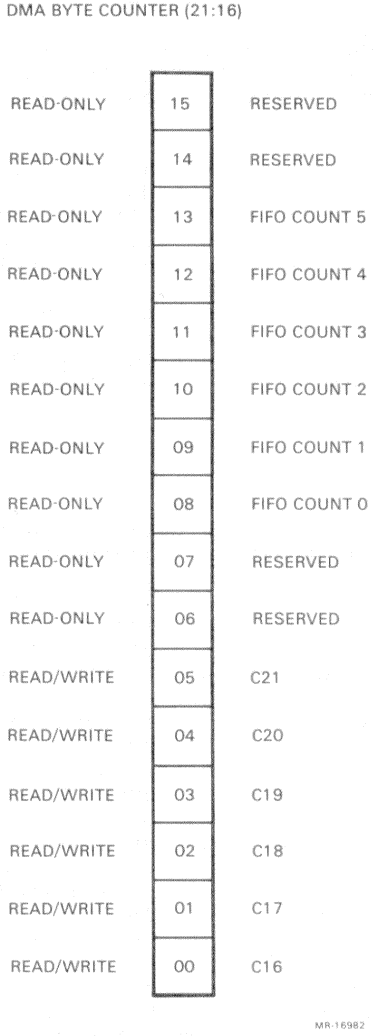


Figure 3-53 DMA Byte Counter Register 4 Bitmap

3.5.6.6 FIFO Register (Register 5) -- The FIFO implemented by the DMA gate array uses the first 64 word locations of the template RAM. The gate array maintains two pointers into the RAM that control the loading and unloading process. The MicroVAX CPU can access the FIFO by addressing register 5 (FIFO register). When the MicroVAX CPU accesses the FIFO register, the appropriate pointer (read or write) is incremented and used as the address for the template RAM cycle.

FIFO data always passes through the DMA gate array. The MicroVAX CPU can determine how many words are in the FIFO by reading register 2 (DMA address counter <21:16>). Bits 08 through 13 of register 2 indicate the number of words in the FIFO (0 = empty, 63 = full). The MicroVAX CPU can access the FIFO locations directly by addressing the first 64 words in the template RAM. Accessing the FIFO by addressing the template RAM does not affect the FIFO pointers. The MicroVAX CPU should not access the FIFO register or modify any of the FIFO 3-112 locations by addressing the template RAM if a DMA process is active. Figure 3-54 shows the FIFO register bitmap.

3.5.6.7 Cursor X Position Register (Register 6) --

NOTE

The format of the data loaded into the cursor X position register is complex. Read this section carefully.

Bits 02 through 10 of the X position register are loaded into an up-counter by the first cursor clock pulse (signal pin CURCLK H) after the assertion of the signal pin CSYNC H (that is, during the horizontal sync interval). The second CURCLK H pulse completes the synchronous load of the 2's complement value into the X counter. Beginning with the third CURCLK H pulse, the X counter is incremented (effectively by 4 since the X counter begins with bit 02) by CURCLK H (cursor clock). CURCLK H is one-fourth the pixel frequency. The cursor data is output when all bits of the X counter are set (bits 02 through 10 are all 1). Because the X counter begins counting during the horizontal retrace period, an offset must be included in the data loaded into the cursor X position register in order to position the cursor at the left edge of the active video line. Refer to the algorithm in Figure 3-55.

The data loaded into the X counter (bits 02 through 10 of the cursor X position register) must be 2's complement data. The two least significant bits of the cursor X position register cannot be 2's complement data.

FIFO REGISTER — READ/WRITE

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00

MR-16983

Figure 3-54 FIFO Register Bitmap

CURSOR X POSITION REGISTER — WRITE-ONLY

15	12	11	10	09	08	07	06	05	04	03	02	01	00
		X11	X10	X09	X08	X07	X06	X05	X04	X03	X02	X01	X00

PIXEL ALIGNMENT DIAGRAM

X01	X00	CUR0	CUR1	CUR2	CUR3	CUR0	CUR1	CUR2	CUR3	CUR0	CUR1	CUR2	CUR3	CUR0	CUR1	CUR2	CUR3	CUR0	CUR1	CUR2	CUR3
0	0	0	0	0	0	D000	D001	D002	D003	D004	D005	D006	D007	D008	D009	D010	D011	D012	D013	D014	D015	0	0	0	0
0	1	0	0	0	0	0	D000	D001	D002	D003	D004	D005	D006	D007	D008	D009	D010	D011	D012	D013	D014	D015	0	0	0
1	0	0	0	0	0	0	0	D000	D001	D002	D003	D004	D005	D006	D007	D008	D009	D010	D011	D012	D013	D014	D015	0	0
1	1	0	0	0	0	0	0	0	D000	D001	D002	D003	D004	D005	D006	D007	D008	D009	D010	D011	D012	D013	D014	D015	0

CURSOR ENABLED

X00 = BIT 00 OF THE CURSOR X POSITION REGISTER
X01 = BIT 00 OF THE CURSOR X POSITION REGISTER

CUR0 = THE CURA0 H OR CURB0 H OUTPUTS OF THE GATE ARRAY
CUR1 = THE CURA1 H OR CURB1 H OUTPUTS OF THE GATE ARRAY
CUR2 = THE CURA2 H OR CURB2 H OUTPUTS OF THE GATE ARRAY
CUR3 = THE CURA3 H OR CURB3 H OUTPUTS OF THE GATE ARRAY

D0, ... , D15 = BIT 0, ... , 15 OF THE CURSOR A OR B DATA READ FROM THE TEMPLATE RAM.

MR-16984

Figure 3-55 Cursor X Position Register Bitmap and Algorithm

The two least significant bits of the cursor X position register shift the cursor data so that it is pixel-aligned. (Refer to Figure 3-55.) The cursor X position register should be modified only during the vertical sync interval, or when the control and status register cursor enable bit (02) is reset. After modifying the cursor X position register, at least one CSYNC H pulse must occur prior to the start of the first active display line. (Equalization pulses are required unless a least one blank line occurs after the release of the vertical sync pulse.) The cursor X position register cannot be read. Writing bits 12 through 15 has no effect.

3.5.6.8 Cursor Y Position Register (Register 7) -- The cursor Y position register is loaded into an up-counter on the second horizontal sync pulse during the vertical retrace period. (BLANK H remains asserted for at least two horizontal sync periods.) The Y counter is incremented by the trailing edge of BLANK H (the start of an active video line). The contents of the Y counter are checked on the leading edge of CSYNC H (during the horizontal retrace preceding the next scan line). If control and status register bit 02 is set, and all except the four least significant bits of the Y counter are set, the cursor is enabled for output on the current display line, and two cursor data words (16 bits) are read from the template RAM. The cursor Y position register should be modified only during the vertical sync period, or when the cursor enable bit (02) in the CSR is reset (cursor is disabled). After modifying the cursor Y position register, at least two CSYNC H pulses must occur prior to the deassertion of BLANK H (the end of the vertical sync blanking interval). The cursor Y position register cannot be read. Writing bits 12 to 15 has no effect. Figure 3-56 shows the cursor Y position register bitmap.

3.5.6.9 Interrupt Register (Register 8) -- The interrupt register provides interrupt status to the MicroVAX CPU. When read, the status bits are latched by the assertion of the signal pin DIN H and remain latched until DIN H is removed. By writing the interrupt register, the base for the interrupt vector output by the DMA gate array during interrupt acknowledge cycles can be selected. (Refer to Paragraph 3.5.3.) Each bit is described in Figure 3-57.

Bit 15: Latched IRQ (Read-Only)

The latched IRQ bit (15) is set whenever the latched DMA IRQ bit (12), latched IRQ1 bit (13), or latched IRQ2 bit (14) is set. Writing bit 15 has no effect. Asserting the signal pin QINIT H resets the latched IRQ bit (15).

Bit 14: Latched IRQ2 (Read-Only)

The latched IRQ2 bit (14) is set whenever the control and status register external interrupt enable bit (02) is set and the signal pin IRQ2 L becomes asserted (the adder chip is requesting interrupt service). Writing bit 14 has no effect. Asserting the signal pin QINIT H resets the latched IRQ2 bit (14).

CURSOR Y POSITION REGISTER WRITE-ONLY

15 12	11	10	09	08	07	06	05	04	03	02	01	00
	Y11	Y10	Y09	Y08	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00

MR-16985

Figure 3-56 Cursor Y Position Register Bitmap

READ-ONLY	15	LATCHED IRQ
READ-ONLY	14	LATCHED IRQ2
READ-ONLY	13	LATCHED IRQ1
READ-ONLY	12	LATCHED DMA IRQ
READ-ONLY	11	RESERVED
READ-ONLY	10	RESERVED
READ-ONLY	09	RESERVED
READ/WRITE	08	VECTOR BASE 08
READ/WRITE	07	VECTOR BASE 07
READ/WRITE	06	VECTOR BASE 06
READ/WRITE	05	VECTOR BASE 05
READ/WRITE	04	VECTOR BASE 04
READ-ONLY	03	INTERRUPT VECTOR 03
READ-ONLY	02	INTERRUPT VECTOR 02
READ-ONLY	01	RESERVED
READ-ONLY	00	RESERVED

MR-16986

Figure 3-57 Interrupt Register Bitmap

Bit 13: Latched IRQ1 (Read-Only)

The latched IRQ1 bit (13) is set whenever the control and status register external interrupt enable bit (02) is set and the signal pin IRQ1 L becomes asserted (the communications controller is requesting interrupt service). Writing bit 13 has no effect. Asserting the signal pin QINIT H resets the latched IRQ1 bit (13).

Bit 12: Latched DMA IRQ (Read-Only)

The latched DMA IRQ bit (12) is set whenever the control and status register DMA interrupt enable bit (01) is set and a DMA interrupt occurs. The DMA interrupt status can be read from the control and status register (register 0). Writing bit 12 has no effect. Asserting the signal pin QINIT H resets the latched DMA IRQ bit (13).

Bit 11: Reserved (Read-Only)

When read, bit 11 is always reset. Writing bit 11 has no effect.

Bit 10: Reserved (Read-Only)

When read, bit 10 is always reset. Writing bit 10 has no effect.

Bit 09: Reserved (Read-Only)

When read, bit 09 is always reset. Writing bit 09 has no effect.

Bit 08: Vector Base 8 (Read/Write)

Bit 08 of the interrupt vector output by the DMA gate array is selected by writing this bit. The vector base 8 bit (08) is reset when the signal pin QINIT H is asserted.

Bit 07: Vector Base 7 (Read/Write)

Bit 07 of the interrupt vector output by the DMA gate array is selected by writing this bit. The vector base 7 bit (07) is reset when the signal pin QINIT H is asserted.

Bit 06: Vector Base 6 (Read/Write)

Bit 06 of the interrupt vector output by the DMA gate array is selected by writing this bit. The vector base 6 bit (06) is reset when the signal pin QINIT H is asserted.

Bit 05: Vector Base 5 (Read/Write)

Bit 05 of the interrupt vector output by the DMA gate array is selected by writing this bit. The vector base 5 bit (05) is reset when the signal pin QINIT H is asserted.

Bit 04: Vector Base 4 (Read/Write)

Bit 04 of the interrupt vector output by the DMA gate array is selected by writing this bit. The vector base 4 bit (04) is reset when the signal pin QINIT H is asserted.

Bit 03: Interrupt Vector 3 (Read-Only)

The interrupt vector 3 bit (03) is priority encoded and is determined by the status of the latched DMA IRQ bit (12), latched IRQ1 bit (13), and the latched IRQ2 bit (14), as shown in Table 3-6. Writing bit 03 has no effect.

Bit 02: Interrupt Vector 2 (Read-Only)

The interrupt vector 2 bit (02) is priority encoded and is determined by the status of the latched DMA IRQ bit (12), latched IRQ1 bit (12), and the latched IRQ2 bit (14), as shown in Table 3-6. Writing bit 02 has no effect.

Bit 01: Reserved (Read-Only)

When read, bit 01 is always reset. Writing bit 01 has no effect.

Bit 00: Reserved (Read-Only)

When read, bit 00 is always reset. Writing bit 00 has no effect.

Table 3-6 Interrupt Vector 3 Determination

Latched IRQ2	Latched IRQ1	Latched DMA IRQ	Interrupt Vector 3	Interrupt Vector 2
X	X	0	0	0
X	0	1	0	1
0	1	1	1	0
1	1	1	1	1
1	1	1	1	1

3.5.6.10 Memory Base Address Register -- A power reset disables the address decoder in the DMA gate array. To enable the address decoder, a value must be written to the I/O page control and status register (CSR) and decoded external to the gate array by logic on the base module. When this address is detected, the external decoder asserts the CONSEL L signal pin on the gate array. If the cycle is a write cycle, the DMA gate array writes the bus data into its memory base register, asserts the signal pin RPLYO H, and enables its address decoding logic. If the cycle is a read cycle, the gate array asserts RPLYO H, and the contents of the external I/O page CSR are output. (The gate array does not drive the data lines.)

The value written into the memory base register selects the starting address, or base address, of a 56-Kbyte block of addresses that are decoded by the DMA gate array. The base address is always on a 64-Kbyte boundary in the Q22-Bus memory address space, as selected by bits 00 through 05 of the memory base register. Bits 05 through 00 are compared with the signal pins A(21:16) H during the address portion of Q22-Bus memory cycles. Figure 3-58 shows the memory base register bitmap.

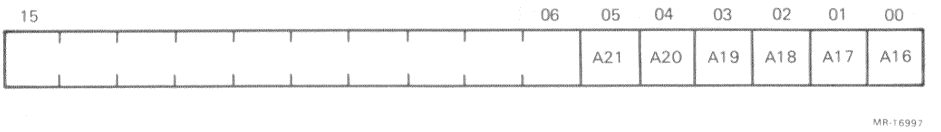


Figure 3-58 Memory Base Register Bitmap

3.5.7 Signal Description

The signal pins on the DMA gate array are listed in Table 3-7.

3.5.7.1 Q22-Bus Interface -- The signals described in this section are provided by the DMA gate array for interfacing to the Q22-Bus.

AD(15:00) H -- Address/Data (Input/Output, Tri-state)

These multiplexed address and data lines are used for both master and slave cycles on the Q22-Bus. During the address portion of a DMA transfer (master) cycle, the DMA gate array outputs the lower 16 bits of the DMA destination address on AD(15:00) H. When the gate array is a Q22-Bus slave device, it latches the address information received on AD(15:00) when SYNC H is asserted.

Table 3-7 DMA Gate Array Signals

Signal	Description
Q22-Bus Interface	The signals needed to interface to the Q22-Bus, both as a slave and master device.
Address Decodes	The signals decoded from the Q22-Bus address lines whenever the DMA gate array is not the bus master; can be used for selecting devices that need to be interfaced to the Q22-Bus.
Interrupt Inputs	The inputs that may be connected to interrupt sources external to the DMA gate array.
Address Processor Chip Interface	The address and control signals needed to interface to the address processor chip.
Template RAM Interface	The address and control signals for the template RAM.
Private Data Bus	A 16-bit bidirectional data bus that connects the DMA gate array, the template RAM, and the address processor chip.
Cursor Signals	The cursor data outputs and the signals that control the timing of the cursor logic.
Miscellaneous Signals	The signals common to the entire gate array but not associated with any one group of signals.

During the data portion of a Q22-Bus cycle, these lines contain read or write data, depending on the type of transfer. Bits AD(15:08) are defined as the high byte and bits AD(07:00) are the low byte of data. If the DMA gate array is the bus master, it outputs data on AD(15:00) during data-out cycles (DOUT H asserted) and receives data on AD(15:00) during data-in cycles (DIN H asserted). If the template RAM, the address processor chip, or a gate array register is being addressed during a slave cycle, the gate array receives data on AD(15:00) during data-out cycles (DOUT H asserted) and drives data on AD(15:00) during data-in cycles (DIN H asserted).

A(17:16) H -- Address/Parity (Input/Output, Tri-state)

During the address portion of Q22-Bus cycles, these two lines are used as address lines. If the gate array is not the bus master (not performing DMA transfers), A(17:16) H are compared with bit 01 and bit 00, respectively, of the memory base register. If the bits are not equal when SYNC H is asserted, the address decode logic is disabled. When it is the bus master (during DMA transfers), the DMA gate array drives addresses on these lines only during the address portion of cycles.

When the gate array is the bus master performing a data-out cycle (DOUT H asserted), A(17:16) H are driven to the deasserted state (all zeros) during the data portion of the cycle. When the gate array is a Q22-Bus slave device responding to a data-in cycle (DIN H asserted), it drives A(17:16) H to the deasserted state. (This disables memory parity detection.) The gate array receives these lines at all other times.

When the DMA gate array is performing a DMA read cycle (DIN H asserted), A(17:16) H are used to check for slave memory parity errors during the data portion of the cycle. A slave device asserts bit A17 H to inform the gate array (the bus master) that it has parity error detection logic. If a parity error occurs on the slave memory device, the slave asserts bit A16 H to indicate the error.

A(21:18) H -- Address (Input/Output, Tri-state)

During the address portion of Q22-Bus cycles, these lines are used as address lines. When it is the bus master (during DMA transfers), the DMA gate array outputs addresses on these lines only during the address portion of cycles. When the gate array is the bus master performing a data-out cycle (DOUT H asserted), A(21:18) H are driven to the deasserted state (all zeros) during the data portion of the cycle. (This disables bus parity detection.) When the gate array is a Q22-Bus slave device responding to a data-in cycle (DIN H asserted), it drives A(21:18) H to the deasserted state. The gate array receives these lines at all other times.

If the gate array is not the bus master (not performing DMA transfers), A(21:18) H are compared with bits 05, 04, 03, and 02, respectively, of the memory base register. If the bits are not equal when SYNC H is asserted, the address decode logic is disabled.

BS7 H -- Bank Select 7 (Input/Output, Tri-state)

During the address portion of Q22-Bus cycles, BS7 H is asserted if the Q22-Bus I/O page is being addressed.

During the data portion of Q22-Bus DATBI cycles, BS7 H is asserted to indicate to the slave device that the bus master wants to perform a block-mode cycle. The Q22-Bus specification does not require the assertion of BS7 during the data portion of Q22-Bus DATBO cycles. The DMA gate array asserts BS7 H when it is the bus master (performing DMA) during the data portion of both DATBI and DATBO cycles. The gate array receives BS7 H at all other times.

If the gate array is not the bus master (not performing DMA transfers), and BS7 H is asserted when SYNC H is asserted, the address decode logic is disabled (that is, the address decode logic in the gate array is disabled whenever the Q22-Bus I/O page is being addressed).

WTBT H -- Write/Byte (Output, Tri-state)

The DMA gate array asserts WTBT H only when it is the bus master. WTBT H is asserted by the bus master during the address portion of a cycle to indicate that an output cycle (DATO, DATOB, DATBO) rather than an input cycle, is to follow. WTBT H is asserted by the bus master during the data portion of a DATOB or DATIOB bus cycle to indicate a byte transfer, rather than a word transfer, is to take place.

NOTE

As a slave, the DMA gate array is word-addressable only. As a master, the gate array can perform byte transfers.

SYNC H -- Synchronize (Input/Output, Tri-state)

The bus master asserts SYNC H to indicate that it has placed an address on the bus. The transfer is in process until SYNC H is negated. When doing block moves, SYNC H remains asserted until the last transfer is completed. SYNC H is not asserted during interrupt acknowledge cycles.

The DMA gate array drives SYNC H only when it is the bus master. SYNC H is an input at all other times.

DIN H -- Data Input (Input/Output, Tri-state)

If SYNC H is not asserted while DIN H is asserted, the cycle is an interrupt acknowledge cycle. The gate array latches the interrupt request status on the leading edge of DIN H.

If DIN H is asserted while SYNC H is asserted, a Q22-Bus read cycle is being performed. The addressed slave device drives the bus with data, and the master inputs the data. When the gate array is a Q22-Bus slave device, it receives DIN H and asserts RPLYO H to complete the cycle.

The DMA gate array outputs DIN H when it is the bus master (and is performing a read cycle). The slave device must respond by asserting RPLYI H to complete the read cycle.

DOUT H -- Data Output (Input/Output, Tri-state)

DOUT H is asserted by the bus master during data output (write) cycles. The bus master asserts DOUT H after it outputs the data to the bus. The slave device must respond by asserting RPLYI H to complete the write cycle.

The DMA gate array outputs DOUT H when it is the bus master (and is performing a write cycle). When the gate array is a Q22-Bus slave device, it receives DOUT H and asserts RPLYO H to complete the cycle.

RPLYO H -- Reply Output (Output, Tri-state)

When the DMA gate array is selected as the bus slave, it asserts RPLYO H in response to DIN H or DOUT H. RPLYO H indicates to the bus master that the gate array has placed its data on the bus (when DIN H is asserted), or has accepted data from the bus (DOUT H is asserted).

If the DMA gate array is asserting IRQ H when IAKI H is received, it places the appropriate interrupt vector on the bus and asserts RPLYO H to indicate it has accepted the interrupt acknowledge.

RPLYO H is never asserted when the DMA gate array is the bus master.

RPLYI H -- Reply Input (Input)

When the DMA gate array is the bus master, RPLYI H is monitored to determine when the selected bus slave is ready to complete the bus cycle. RPLYI H indicates to the gate array that the slave device has placed its data on the bus (when DIN H is asserted), or has accepted data from the bus (DOUT H is asserted).

DMRO H -- Direct Memory Access Request Output (Output, Tri-state)

The DMA gate array asserts DMRO H to request bus mastership of the Q22-Bus. After bus mastership is granted to the DMA gate array, the gate array asserts SACK H and deasserts DMRO H.

DMRI H -- Direct Memory Access Request Input (Input)

When the DMA gate array is the bus master, it monitors DMRI H to determine if another DMA device is requesting bus mastership. Under certain conditions the gate array gives up bus mastership to allow another DMA device to acquire it.

DMGI H -- Direct Memory Access Grant Input (Input)

The MicroVAX CPU asserts DMGI H to indicate it is granting bus mastership to a Q22-Bus device. If the DMA gate array is asserting DMR H (requesting the bus) when DMGI H becomes asserted, it waits for the deassertion of SYNC H and RPLYI H, asserts SACK H, and removes DMRO H. (The gate array becomes the bus master.)

If the gate array does not assert DMRO H when it is received, the gate array asserts DMGO H and passes the DMA grant to the next Q22-Bus device.

DMGO H -- Direct Memory Access Grant Output (Output, Tri-state)

The DMA gate array asserts DMGO H, if the DMGI H signal is asserted, and not asserting DMRO H (not requesting bus mastership).

SACK H -- Signal Acknowledge (Output, Tri-state)

SACK H is asserted by the DMA gate array when it has acquired bus mastership of the Q22-Bus. The signal indicates that the gate array is accepting bus mastership. The gate array keeps SACK H asserted until it gives up bus mastership.

The DMA gate array requests bus mastership by asserting DMRO H. When DMGI H is received and DMRO H is asserted, the gate array waits for the deassertion of SYNC H and RPLYI H, asserts SACK H, and removes DMRO H. (The gate array becomes the bus master.)

SACK H also controls the direction of the transceiver for the Q22-Bus signals SYNC H, DIN H, DOUT H, WTBT H, and BS7 H. When SACK H is asserted, the VCB02 base module transmits the control signals. When SACK H is deasserted, the base module receives the control signals.

REF H -- Refresh (Input)

As bus master, the DMA gate array monitors this signal to determine if the slave device is a block-mode device. Block-mode slave devices assert this line along with RPLYI H, if they are able to do subsequent data transfers. Slave devices that do not support block mode do not assert this line.

NOTE

The DMA gate array is a block mode master, not a block mode slave device.

IRQ H -- Interrupt Request (Output, Tri-state)

The DMA gate array asserts this line when it requires an interrupt on the Q22-Bus. Two bits (01 and 02) in the control and status register are provided to enable/disable the internally connected DMA interrupt and the external interrupts IRQ1 H and IRQ2 H (from the address processor chip and communications controller).

IRQ H deasserts when the interrupt request is removed or IAKI H is received acknowledging the interrupt.

The interrupt priority level of the DMA gate array is position-dependent. (See Appendix A.)

IAKI H -- Interrupt Acknowledge Input (Input)

The MicroVAX CPU asserts IAKI H to acknowledge an interrupt request. If the DMA gate array is asserting IRQ H when it receives IAKI H, it outputs an interrupt vector and clears the highest priority interrupt. If IAKI H is received by the gate array when IRQ H is not asserted, the gate array outputs IAKO H to pass the interrupt acknowledge to the next Q22-Bus slot.

The interrupt priority level of the DMA gate array is position-dependent. (See Appendix A.)

IAKO H -- Interrupt Acknowledge Output (Output, Tri-state)

If the DMA gate array is not requesting an interrupt and receives IAKI H, the gate array asserts IAKO H. The assertion of this signal propagates the interrupt acknowledge to the next Q22-Bus slot.

If the DMA gate array receives IAKO H and is requesting an interrupt, the gate array does not assert IAKO H.

The interrupt priority level of the DMA gate array is position-dependent. (See Appendix A.)

QINIT H -- Initialize (Input)

QINIT H is used by the DMA gate array as a software reset. When QINIT H is asserted, the following states are effected in the gate array.

1. The address decoder is disabled. The memory base register has to be rewritten before accessing any locations other than the VCB02 subsystem's I/O page control status register.

2. Control and status register bit 15 is set. All other control and status register bits are reset.

3. The DMA logic in the gate array is reset, causing any DMA transfers in progress to be aborted.

4. The 22 bits of the DMA address counter are reset.

5. The 22 bits of the DMA byte counter are reset.

6. The FIFO pointers are reset. (The FIFO is empty.)

7. The template address counter is reset.

8. All interrupts are disabled.

9. The hardware cursor is disabled.

10. The cursor X position register is reset.

11. The cursor Y position register is reset.

POK L -- Power Okay (Input)

POK L must be asserted to enable the DMA gate array to request bus mastership. When POK L is deasserted, the DMRO H output is disabled. (The gate array cannot request bus mastership.) If POK L is deasserted while the DMA gate array is the bus master, it does not abort the DMA process. If POK L remains deasserted, the gate array is prevented from acquiring bus mastership again after it completes the current DMA process.

BTIMEO L -- Bus Timeout (Open Collector Output, Tri-state)

BTIMEO L is used to discharge a timing capacitor external to the gate array (connected through a current limit resistor).This capacitor is used to generate a bus timeout error when the slave device fails to assert REPLYI H in response to a DIN H or DOUT H data strobe within 10 μs (approximate time constant of charging circuit). If REPLYI H is asserted before the 10 μs limit, the gate array deasserts the data strobe (DIN H or DOUT H) and asserts BTIMEO L, which discharges the capacitor (preventing a bus timeout error).

BTIMEI H is used to monitor the voltage on the capacitor. When BTIMEO L is deasserted, the capacitor begins to charge (begins timing period). If the capacitor voltage exceeds the logic high input threshold of the BTIMEI H input, a bus timeout error occurs.

When the gate array is a slave device, BTIMEO L is always asserted. When the DMA gate array is the bus master, BTIMEO L is asserted (when both DIN H and DOUT H are not asserted).

BTIMEI H -- Bus Time Input (Input, Schmitt Trigger)

BTIMEI H is used to detect a Q22-Bus timeout when one occurs. When the DMA gate array is bus master and asserts DIN H or DOUT H, the gate array stops driving BTIMEO L, which allows an external timing capacitor to charge. If the slave device has not asserted RPLYI H by the time this input exceeds the logic high input threshold, a bus timeout error occurs, and the bus timeout bit (05) and DMA error bit (07) in the control and status register (CSR) are set. If the DMA interrupt enable bit (01) in the CSR is set, the DMA gate array asserts IRQ H (request interrupt service).

DTIMEO L -- DMA Delay Timeout (Open Collector Output, Tri-state)

DTIMEO L is used to discharge a timing capacitor external to the gate array (connected through a current limit resistor). This capacitor is used to generate a delay between requests for bus mastership by the DMA gate array. DTIMEO L is asserted whenever SACK H is asserted. (The gate array is the bus master.) When the gate array gives up bus mastership (SACK H is deasserted), DTIMEO L deasserts and the timing capacitor begins to charge (begins timing the delay).

DTIMEI H is used to monitor the voltage on the capacitor. Until the capacitor voltage reaches the logic high threshold for DTIMEI H, the DMA gate array cannot request bus mastership (DMRO H is disabled). When the capacitor voltage exceeds the logic high threshold of DTIMEI H, DMRO H is enabled and the gate array can request bus mastership again.

DTIMEI H -- DMA Delay Time Input (Input, Schmitt Trigger)

DTIMEI H monitors the voltage on an external timing capacitor. When DTIMEI H is low, the DMA gate array does not request bus mastership. (DMRO H is disabled.) The external capacitor is discharged by DTIMEO L whenever the gate array is the bus master. When the gate array gives up bus mastership, the capacitor begins to charge. After a delay, the capacitor charges to a voltage that asserts DTIMEI H, and enables the gate array to request bus mastership again.

XVREN0 L -- Transceiver Enable 0 (Output, Tri-state)

XVREN0 L is the enable for the transceiver for the Q22-Bus control signals SYNC H, DIN H, DOUT H, WTBT H, and BS7 H. XVREN0 L disables the transceiver when the DMA gate array transitions from slave to master and master to slave.

XVREN1 H -- Transceiver Enable 1 (Output, Tri-state)

XVREN1 H is the enable for the address and data transceivers for the Q22-Bus. XVREN1 H disables the transceivers when at the leading and trailing edges of DIN H (except for IAK cycles). XVREN1 H also disables the transceivers when the DMA gate array transitions from slave to master and master to slave.

XVRRCV L -- Transceiver Receive (Output, Tri-state)

XVRRCV L controls the direction of the Q22-Bus transceivers for address and data. When XVRRCV L is asserted, the VCB02 base module receives data. When XVRRCV L is deasserted, the base module transmits data. XVRRCV L is asserted/deasserted only when XVREN1 H is deasserted (except for IAK cycles).

3.5.7.2 Address Decodes -- This section describes the signals provided by the DMA gate array for decoding Q22-Bus addresses.

CONSEL L -- Console Select (Input)

CONSEL L is an external Q22-Bus address decode for the VCB02 subsystem's I/O page control and status register (CSR). If the slave cycle accessing the I/O page CSR is a write cycle, the DMA gate array writes the data into its memory base address register and enables the address decoder. If the slave cycle accessing the I/O page CSR is a read cycle, the gate array does not drive the AD(15:00) signals. The DMA gate array responds by asserting the RPLYO H line for both read and write cycles when CONSEL L is asserted.

IOENB L -- I/O Enable (Output, Tri-state)

When the gate array is not the bus master, it asserts IOENB L when it decodes an address in the range VCB02 memory base address + C400 (hex) to VCB02 memory base address + DFFF (hex). IOENB L is used as an enable for an external decoder on the base module that segments the 7-Kbyte address block to provide register space for the communications controller, the memory page control status register, and the color maps (red, green, and blue). The DMA gate array does not drive the data bus lines when IOENB L is decoded.

ROMENB L -- ROM Enable (Output, Tri-state)

ROMENB L is an address decode for slave cycles that access the VCB02 base module's ROM. The address range for the ROM is from VCB02 memory base address + 0000 (hex) to VCB02 memory base address + 7FFF (hex).

3.5.7.3 **Interrupt Inputs** -- The interrupt inputs connected to the address processor chip interrupt output and the communications controller interrupt output are described in this section. An additional interrupt is provided internally to the DMA gate array for the DMA logic. Refer to Paragraph 3.5.7.1 for details on the signals IRQ H, IAKI H, and IAKO H.

IRQ(2:1) L -- Interrupt Request (2:1) (Input)

IRQ2 L is connected to the communications controller on the VCB02 base module. IRQ1 L is connected to the address processor chip on the VCB02 base module. The interrupt request signals are used by the external devices to indicate to the DMA gate array that service by the MicroVAX CPU is required. IRQ2 L and IRQ1 L are asynchronous inputs that are low-level sensitive. Request for service must be held low until the interrupt has been acknowledged. An additional DMA interrupt is provided internal to the gate array.

The interrupts are priority encoded, as shown in Table 3-8. During an interrupt acknowledge cycle, only the highest priority interrupt is cleared. IRQ(2:1) L may be masked by the external interrupt enable bit (02) in the control status register. The DMA interrupt may be masked by the DMA interrupt enable bit (01) in the control and status register.

Table 3-8 DMA Interrupt Priority Encoding

Interrupt Priority	Level	Type
Highest	IRQ0	Internal DMA interrupt
	IRQ1	Address processor chip interrupt
Lowest	IRQ2	Communications interrupt

3.5.7.4 **Address Processor Chip Interface** -- The signals described in this section are the control signals between the DMA gate array and the address processor chip.

AA(7:1) H -- Address Processor Address (Output, Tri-state)

The VCB02 uses only AA(6:1) H. AA(7) H is reserved by Digital for future use. AA(6:1) H select the address processor chip register being accessed.

ADDAS L -- Address/Data Strobe (Output, Tri-state)

ADDAS L is a combination of address strobe and data strobe for the address processor chip. The assertion of ADDAS L indicates that a

valid address is present on the AA(7:1) H lines, and that the signal ADDWR L is valid. For write cycles to the address processor, this indicates that valid data is on the PB(15:00) H lines. On read cycles from the address processor, ADDAS L enables data onto the data bus from the address processor. The address processor chip is word-addressable only.

ADDWR L -- Address Processor Write (Output, Tri-state)

This line indicates a read or write operation to the address processor chip. If the signal is asserted when ADDAS L is asserted, the gate array is writing the address processor. If the line is deasserted when ADDAS L is asserted, the gate array is reading from the address processor.

ADDRQ L -- Address Processor DMA Request (Input)

ADDRQ L is a status line from the address processor chip that indicates the chip is ready for a secondary DMA transfer. The ADDRQ L signal is controlled by writing an enable mask to the address processor's request enable register. If the logical AND of the enable mask and the address processor's status register results in any bit being set, the ADDRQ L is asserted. ADDRQ L is used to synchronize secondary DMA transfers with the DMA gate array.

3.5.7.5 **Template RAM Interface** -- The signals described in this section are the control signals for the template RAM generated by the DMA gate array.

TA(13:01) H -- Template Address (Output, Tri-state)

The VCB02 uses only TA(11:01) H. TA13 H and TA12 H are reserved for future use by Digital. TA(11:01) H are address lines for the template RAM. The template RAM is word-accessible only.

RAMWR L -- RAM Write (Output, Tri-state)

RAMWR L is the write strobe for the template RAM. The data on PB(15:00) is written to the template RAM on the trailing edge of RAMWR L.

RAMOE L -- RAM Output Enable (Output, Tri-state)

RAMOE L is the template RAM output enable control line. RAMOE L is asserted when the DMA gate array is performing a read cycle from the template RAM. RAMOE L is used to tri-state the template RAM's output drivers when it is not being read.

3.5.7.6 **Private Data Bus** -- The signals described in this section are the data lines that connect the DMA gate array to the address processor chip and the template RAM.

PB(00:15) H -- Private Bus (Input/Output, Tri-state)

These signals provide the data path between the DMA gate array, the template RAM, and the address processor chip. Any of these devices may receive or transmit data on these signal lines. All transfers on this bus are controlled by the gate array.

3.5.7.7 **Cursor Signals** -- The signals described in this section are used to generate a 2-plane cursor that can be integrated into the video data output logic.

CURA(3:0) H -- Cursor A Plane Data (3:0) (Output)

CURA(3:0) are the plane "A" cursor data outputs. This parallel data is output at one-fourth the frequency of the VCB02 subsystem's video pixel clock, and is connected to an external shift register that generates a single bit stream at the full VCB02 video pixel clock rate. CURCLK H is the clocking signal for CURA(3:0) H.

CURB(0:3) H -- Cursor B Plane Data (3:0) (Output)

CURB(3:0) are the plane "B" cursor data outputs. This parallel data is output at one-fourth the frequency of the VCB02 subsystem's video pixel clock, and is connected to an external shift register that generates a single bit stream at the full VCB02 video pixel clock rate. CURCLK H is the clocking signal for CURA(3:0) H.

CURCLK H -- Cursor Clock (Input)

CURCLK H is the clock used by all the cursor logic. This 50 percent duty cycle clock is one-fourth the frequency of the VCB02 subsystem's video pixel clock.

CSYNC L -- Composite Sync (Input)

CSYNC L is used to generate horizontal and vertical load pulses for the cursor X and cursor Y counters. CSYNC L must have horizontal sync pulses during the vertical sync interval. CSYNC L is connected to the address processor chip's CMPSYN output. CSYNC L is clocked into the gate array on the rising edge of CURCLK H.

BLANK H -- Video Blanking (Input)

BLANK H is used to generate vertical load and count pulses for the cursor Y counter, and load pulses for the cursor X counter. BLANK H is clocked into the gate array on the rising edge of CURLCK H. BLANK H must remain asserted during the vertical sync interval. BLANK H does not disable the CURA(3:0) or CURB(3:0) outputs. BLANK H is connected to the address processor chip's BLANK output.

3.5.7.8 **Miscellaneous Signals** -- The signals described in this section are common to the entire gate array but not associated with a particular signal group.

CLOCK H -- Clock (Input)

This is the system clock for the DMA gate array with a 50 percent duty cycle. The rising edge of CLOCK H is used to generate all timing in the gate array except the cursor timing. In the VCB02 subsystem, CLOCK H is one-fourth the video pixel clock frequency.

GOZ L -- Go Tri-state (Input)

When this line is driven low, all tri-state outputs are disabled (go tri-state). This pin is for manufacturing testing only.

3.5.8 **Physical Description**
The following sections describe the package pin numbering and signal pin assignments of the DMA gate array. The package is a ceramic 120 PGA (pin grid array).

3.5.8.1 **Package Pin Numbering** -- Figure 3-59 identifies the DMA gate array's pin numbers from the top view. Figure 3-60 shows the pin numbering from the bottom view. The top view is used when looking at the hole pattern for the gate array on the component side of the printed circuit board. The bottom view is used when looking at the pin side of the gate array package, or when looking at the hole pattern for the gate array on the etch side of the printed circuit board.

120	117	116	113	111	108	107	104	101	99	96	93	90	A
3	2	119	115	112	109	105	103	100	97	95	92	87	B
6	5	1	118	114	110	106	102	98	94	91	89	86	C
9	7	4	+						*	88	85	83	D
11	10	8	*							84	82	81	E
14	13	12								80	79	78	F
17	15	16				120-PIN GRID ARRAY				76	75	77	G
18	19	20								72	73	74	H
21	22	24								68	70	71	J
23	25	28	*						*	64	67	69	K
26	29	31	34	38	42	46	50	54	58	61	65	66	L
27	32	35	37	40	43	45	49	52	55	59	62	63	M
30	33	36	39	41	44	47	48	51	53	56	57	60	N
1	2	3	4	5	6	7	8	9	10	11	12	13	

NOTE:
THE LETTERS ALONG THE SIDE AND THE NUMBERS ALONG THE BOTTOM OF THE DIAGRAM IDENTIFY THE GRID COORDINATE OF THE PIN.

THE NUMBERS IN THE SQUARES IDENTIFY THE PIN NUMBER.

THE + SYMBOL INDICATES AN EXTRA PIN FOR KEYING THE PACKAGE ORIENTATION.

THE * SYMBOL INDICATES AN INSULATED STANDOFF FOR HOLDING THE PACKAGE UP OFF THE BOARD SURFACE.

MR-16998

Figure 3-59 DMA Gate Array Pin Diagram (Top View)

A	90	93	96	99	101	104	107	108	111	113	116	117	120
B	87	92	95	97	100	103	105	109	112	115	119	2	3
C	86	89	91	94	98	102	106	110	114	118	1	5	6
D	83	85	88	*						+	4	7	9
E	81	82	84							*	8	10	11
F	78	79	80								12	13	14
G	77	75	76				120-PIN GRID ARRAY				16	15	17
H	74	73	72								20	19	18
J	71	70	68								24	22	21
K	69	67	64	*						*	28	25	23
L	66	65	61	58	54	50	46	42	38	34	31	29	26
M	63	62	59	55	52	49	45	43	40	37	35	32	27
N	60	57	56	53	51	48	47	44	41	39	36	33	30
	13	12	11	10	9	8	7	6	5	4	3	2	1

NOTE:
THE LETTERS ALONG THE SIDE AND THE NUMBERS ALONG THE BOTTOM OF THE DIAGRAM IDENTIFY THE GRID COORDINATE OF THE PIN.

THE NUMBERS IN THE SQUARES IDENTIFY THE PIN NUMBER.

THE + SYMBOL INDICATES AN EXTRA PIN FOR KEYING THE PACKAGE ORIENTATION.

THE * SYMBOL INDICATES AN INSULATED STANDOFF FOR HOLDING THE PACKAGE UP OFF THE BOARD SURFACE.

MR-17015

Figure 3-60 DMA Gate Array Pin Diagram (Bottom View)

3.5.8.2 Pin Signal Assignments -- Table 3-9 lists the signal assignments of each pin in numerical order. The package pin grid coordinate, signal name, and the gates used for the receiver and/or driver are given for each pin.

Table 3-9 DMA Gate Array Pin Signal Assignments

Pin Number	Grid Number		Signal Name	Gate Used For Receiver	Gate Used For Driver
1	C	3	VCC		
2	B	2	PB10 H	TLCHTI	BTS7L
3	B	1	PB09 H	TLCHTI	BTS7L
4	D	3	PB08 H	TLCHTI	BTS7L
5	C	2	PB07 H	TLCHTI	BTS7L
6	C	1	PB06 H	TLCHTI	BTS7L
7	D	2	PB05 H	TLCHTI	BTS7L
8	E	3	PB04 H	TLCHTI	BTS7L
9	D	1	PB03 H	TLCHTI	BTS7L
10	E	2	PB02 H	TLCHTI	BTS7L
11	E	1	PB01 H	TLCHTI	BTS7L
12	F	3	PB00 H	TLCHTI	BTS7L
13	F	2	RAMWR L		BTS1
14	F	1	RAMOE L		BTS1
15	G	2	GND		
16	G	3	VCC		
17	G	1	TA13 H		BTS1
18	H	1	TA12 H		BTS1
19	H	2	TA11 H		BTS1
20	H	3	TA10 H		BTS1
21	J	1	TA09 H		BTS1
22	J	2	TA08 H		BTS1
23	K	1	TA07 H		BTS1
24	J	3	TA06 H		BTS1
25	K	2	TA05 H		BTS1
26	L	1	TA04 H		BTS1
27	M	1	TA03 H		BTS1
28	K	3	TA02 H		BTS1
29	L	2	TA01 H		BTS1
30	N	1	GND		
31	L	3	VCC		
32	M	2	A21 H	TLCHTI	BTS7L
33	N	2	A20 H	TLCHTI	BTS7L
34	L	4	A19 H	TLCHTI	BTS7L
35	M	3	A18 H	TLCHTI	BTS7L
36	N	3	A17 H	TLCHTI	BTS7L
37	M	4	A16 H	TLCHTI	BTS7L
38	L	5	AD15 H	TLCHTI	BTS7L
39	N	4	AD14 H	TLCHTI	BTS7L
40	M	5	AD13 H	TLCHTI	BTS7L
41	N	5	AD12 H	TLCHTI	BTS7L
42	L	6	AD11 H	TLCHTI	BTS7L
43	M	6	AD10 H	TLCHTI	BTS7L

Table 3-9 DMA Gate Array Pin Signal Assignments (Cont)

Pin Number	Grid Number	Signal Name	Gate Used For Receiver	Gate Used For Driver
44	N 6	AD09 H	TLCHTI	BTS7L
45	M 7	AD08 H	TLCHTI	BTS7L
46	L 7	GND		
47	N 7	VCC		
48	N 8	AD07 H	TLCHTI	BTS7L
49	M 8	AD06 H	TLCHTI	BTS7L
50	L 8	AD05 H	TLCHTI	BTS7L
51	N 9	AD04 H	TLCHTI	BTS7L
52	M 9	AD03 H	TLCHTI	BTS7L
53	N 10	AD02 H	TLCHTI	BTS7L
54	L 9	AD01 H	TLCHTI	BTS7L
55	M 10	AD00 H	TLCHTI	BTS7L
56	N 11	BS7 H	TLCHTI	BTS7L
57	N 12	WTBT H		BTS1
58	L 10	SYNC H	TLCHTI	BTS7L
59	M 11	DIN H	TLCHTI	BTS7L
60	N 13	GND		
61	L 11	VCC		
62	M 12	DOUT H	TLCHTI	BTS7L
63	M 13	RPLYI H	TLCHT	
64	K 11	RPLYO H		BTS1
65	L 12	SACK H		BTS1
66	L 13	XVREN1 H		BTS1
67	K 12	XVREN0 L		BTS1
68	J 11	XVRRCV L		BTS1
69	K 13	DMGI H	TLCHT	
70	J 12	DMGO H		BTS1
71	J 13	REF H	TLCHT	
72	H 11	DMRI H	TLCHT	
73	H 12	DMRO H		BTS1
74	H 13	IOENB L		BTS1
75	G 12	ROMENB L		BTS1
76	G 11	GND		
77	G 13	CONSEL L	TLCHT	
78	F 13	QINIT H	TLCHT	
79	F 12	POK L	TLCHT	
80	F 11	GOZ L	TLCHT	
81	E 13	DTIMEO L		B1OD
82	E 12	DTIMEI H	SCHMDT1	
83	D 13	CLOCK H	TLCHT	
84	E 11	BTIMEO L		B1OD
85	D 12	BTIMEI H	SCHMDT1	
86	C 13	IRQ H		BTS1
87	B 13	IAKO H		BTS1
88	D 11	IAKI H	TLCHT	
89	C 12	IRQ2 L	SCHMDT1	
90	A 13	IRQ1 L	SCHMDT1	
91	C 11	VCC		
92	B 12	CURB3 H		B1

Table 3-9 DMA Gate Array Pin Signal Assignments (Cont)

Pin Number	Grid Number	Signal Name	Gate Used For Receiver	Gate Used For Driver
93	A 12	CURB2 H		B1
94	C 10	CURB1 H		B1
95	B 11	CURB0 H		B1
96	A 11	CURA3 H		B1
97	B 10	CURA2 H		B1
98	C 9	CURA1 H		B1
99	A 10	CURA0 H		B1
100	B 9	CURCLK H	TLCHT	
101	A 9	BLANK H	TLCHT	
102	C 8	CYSNC H	TLCHT	
103	B 8	ADDRQ L	TLCHT	
104	A 8	ADDWR L		BTS1
105	B 7	VCC		
106	C 7	GND		
107	A 7	ADDAS L		BTS1
108	A 6	AA07 H		BTS1
109	B 6	AA06 H		BTS1
110	C 6	AA05 H		BTS1
111	A 5	AA04 H		BTS1
112	B 5	AA03 H		BTS1
113	A 4	AA02 H		BTS1
114	C 5	AA01 H		BTS1
115	B 4	PB15 H	TLCHTI	BTS7L
116	A 3	PB14 H	TLCHTI	BTS7L
117	A 2	PB13 H	TLCHTI	BTS7L
118	C 4	PB12 H	TLCHTI	BTS7L
119	B 3	PB11 H	TLCHTI	BTS7L
120	A 1	GND		

3.5.9 Input/Output Specifications

Table 3-10 lists the VCB02 I/O specifications.

Vdd = 4.75 V to 5.25 V
Ta = 0°C to 70°C

Vil = Low-level input voltage
Vih = High-level input voltage

Vt+ = Schmitt trigger positive edge threshold
Vt- = Schmitt trigger negative edge threshold

Iil = Low-level input current
Iih = High-level input current

Vol = Low-level output voltage
Voh = High-level output voltage

Ioz = Tri-state leakage output current

Table 3-10 Input/Output Specifications

Symbol	I/O Gate Type	Condition	Min.	Max.	Unit
Vil	TLCHT, TLCHTI		Vss	0.8	V
Vih	TLCHT, TLCHTI		2.0	Vdd	V
Vt+	SCHMDT1			4.0	V
Vt-	SCHMDT1		1.0		V
Iil	TLCHT	Vin = Vss	-0.5	-20	μA
	TLCHTI, SCHMDT1			-10	μA
Iih	TLCHT	Vin = Vdd		20	μA
	TLCHTI, SCHMDT1			10	μA
Vol	B1, B1OD, BTS1, BTS7L	Iol = 2.0 mA		0.4	V
Voh	B1, BTS1, BTS7L	Ioh = -1.6 mA	2.4		V
	B1OD (open collector)		N/A		V
Ioz	BTS1, BTS7L, B1OD		-10	+10	μA

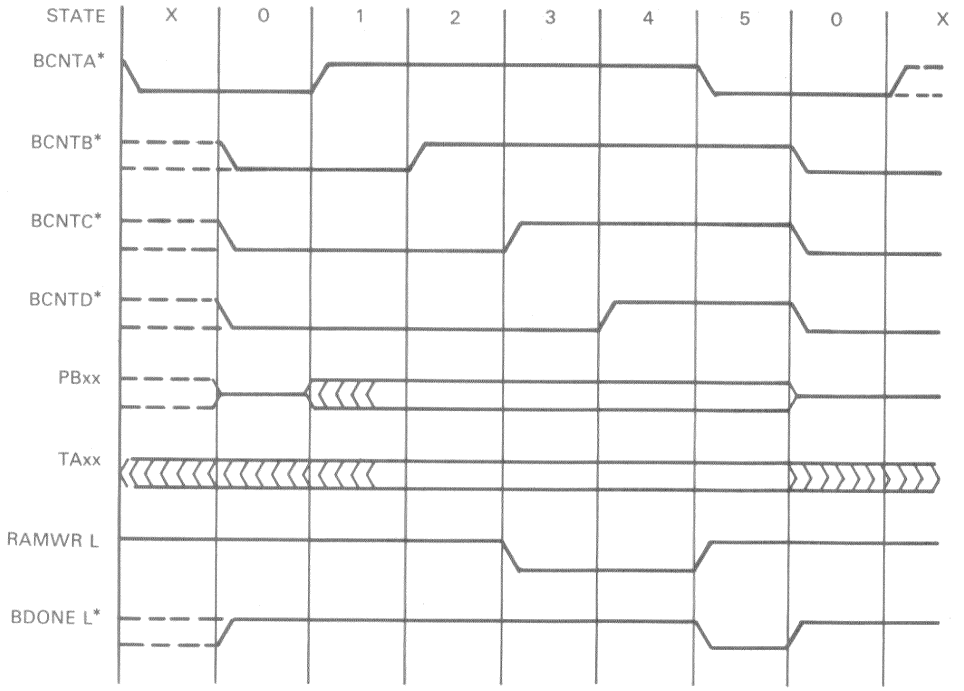
3.5.10 Timing Diagrams

The following paragraphs contain timing diagrams for selected DMA gate array signals.

3.5.10.1 Private Bus Timing -- The following paragraphs contain the timing diagrams for all cycles on the private bus. The "state" changes on the rising edge of CLOCK H.

Template RAM Write Cycle

Figure 3-61 shows the timing for write cycles to the template RAM caused by slave writes to the template RAM or FIFO register, or by DMA writes to the FIFO.

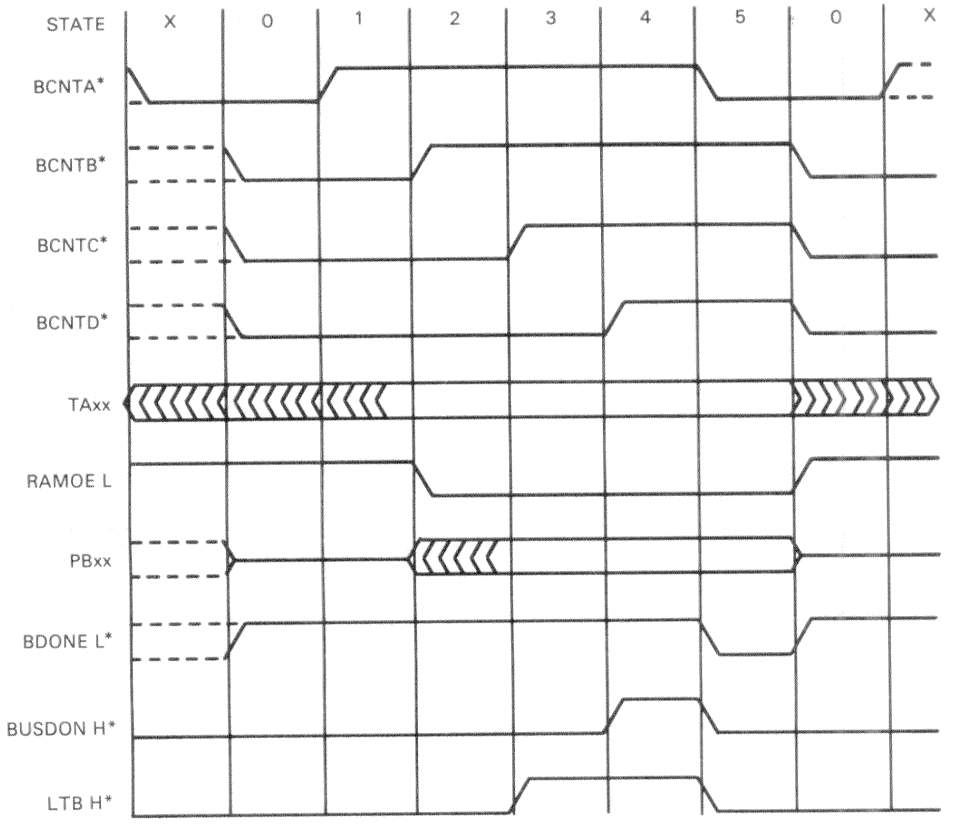


NOTES:
1. RAMOE L IS NOT ASSERTED DURING THIS CYCLE.
2. SIGNALS THAT ARE APPENDED WITH AN * ARE INTERNAL TO THE GATE ARRAY.

MR-16988

Figure 3-61 Template RAM Write Cycle Timing

Template RAM Read Cycle
Figure 3-62 shows the timing for read cycles from the template RAM caused by slave reads from the template RAM or FIFO register, or by DMA reads from the FIFO.



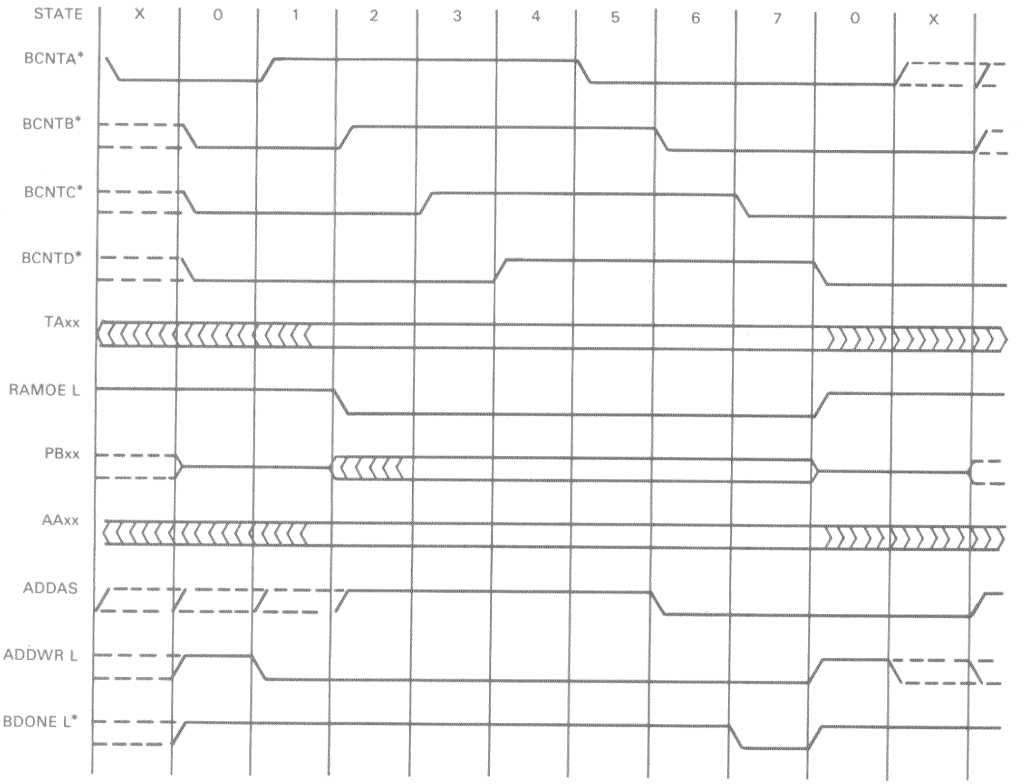
NOTES:
1. RAMWR L IS NOT ASSERTED DURING THIS CYCLE.
2. SIGNALS THAT ARE APPENDED WITH AN * ARE INTERNAL TO THE GATE ARRAY.

MR-16991

Figure 3-62 Template RAM Read Cycle Timing

Template RAM Read, Address Processor Write Cycle

Figure 3-63 shows the timing for cycles that read from the template RAM and write to the address processor chip (caused by the execution of display lists or processor-to-bitmap DMA).



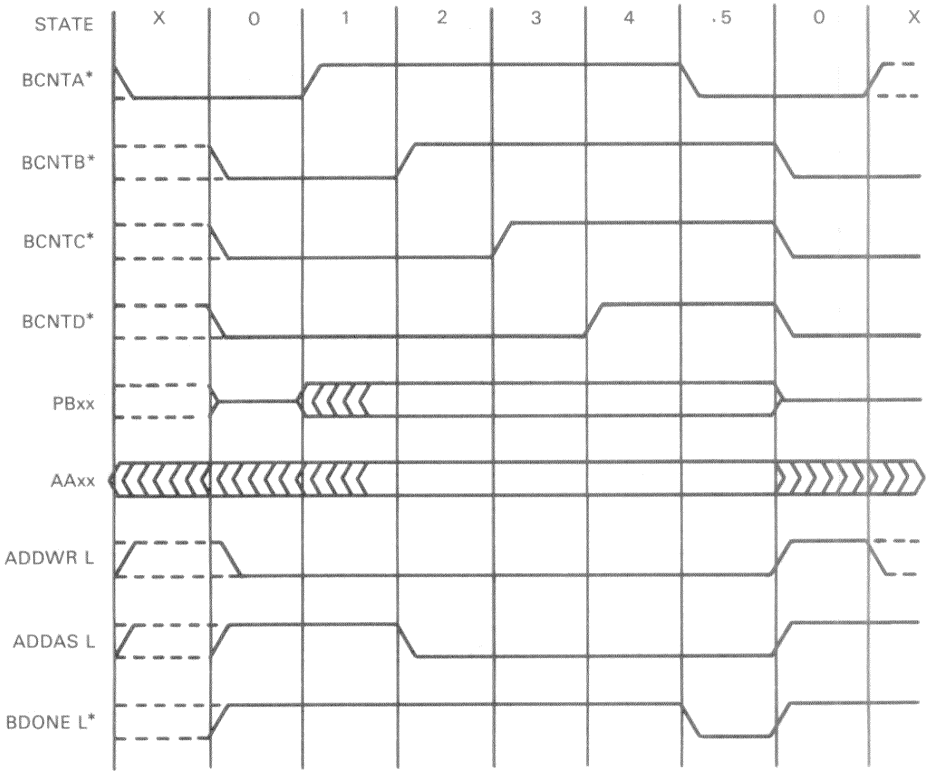
NOTES:
1. RAMWR L IS NOT ASSERTED DURING THIS CYCLE.
2. SIGNALS THAT ARE APPENDED WITH AN * ARE INTERNAL TO THE GATE ARRAY.

MR-16990

Figure 3-63 Template RAM Read, Address Processor Write Cycle Timing

Address Processor Write Cycle

Figure 3-64 shows the timing for write cycles to the address processor chip caused by slave writes.



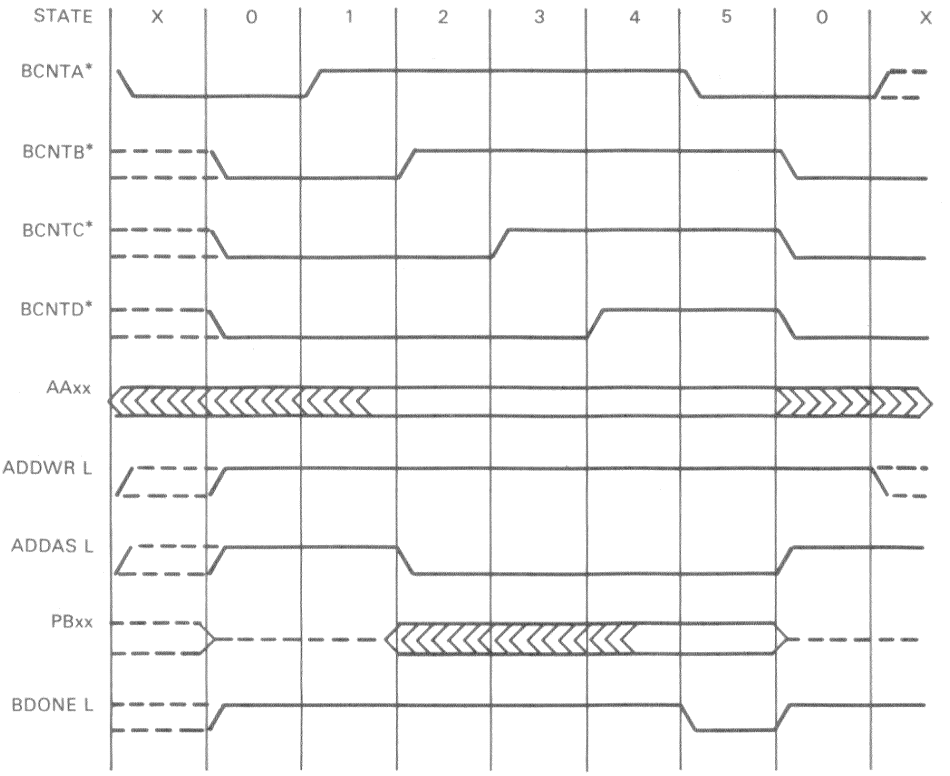
NOTES:
1. RAMOE L AND RAMWR L ARE NOT ASSERTED DURING THIS CYCLE.
2. SIGNALS THAT ARE APPENDED WITH AN * ARE INTERNAL TO THE GATE ARRAY.

MR-16992

Figure 3-64 Address Processor Write Cycle Timing

Address Processor Read Cycle

Figure 3-65 shows the timing for read cycles from the address processor chip caused by slave reads.

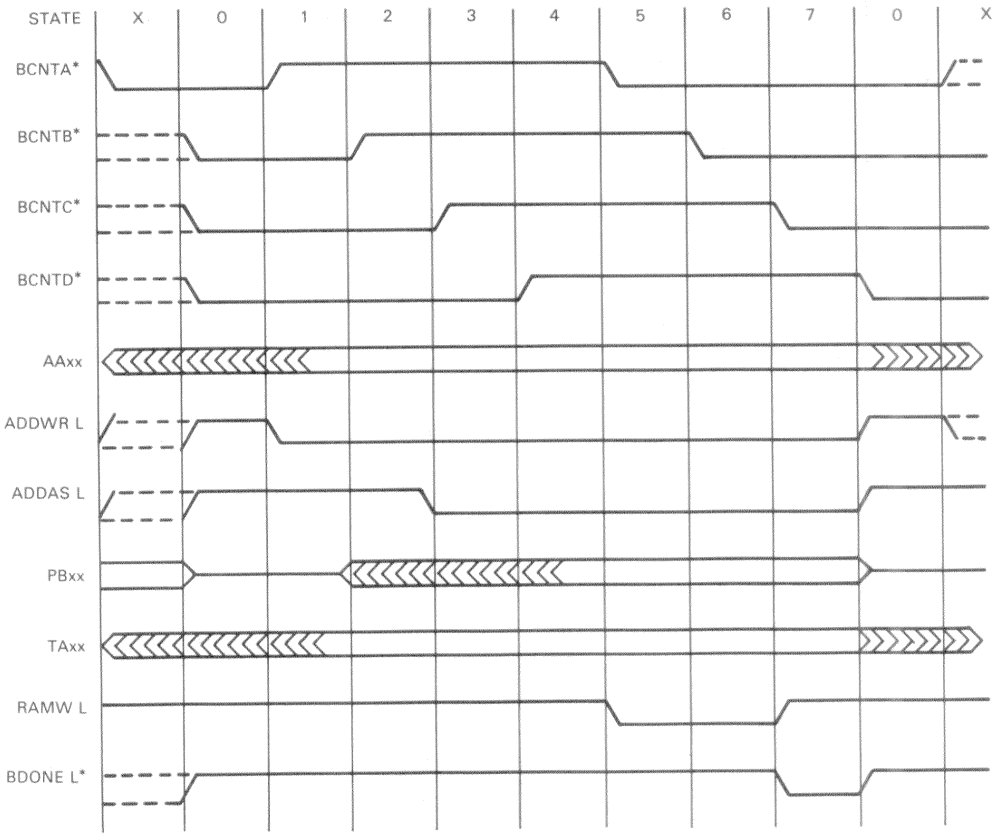


NOTES:
1. RAMOE L AND RAMWR L ARE NOT ASSERTED DURING THIS CYCLE.
2. SIGNALS THAT ARE APPENDED WITH AN * ARE INTERNAL TO THE GATE ARRAY.

MR-16989

Figure 3-65 Address Processor Read Cycle Timing

Address Processor Read, Template RAM Write Cycle
Figure 3-66 shows the timing for cycles that read from the address processor chip and write to the template RAM (caused by bitmap-to-processor DMA).



NOTES:
1. RAMOE L IS NOT ASSERTED DURING THIS CYCLE.
2. SIGNALS THAT ARE APPENDED WITH AN * ARE INTERNAL TO THE GATE ARRAY.

MR-16987

Figure 3-66 Address Processor Read, Template RAM Write Cycle Timing

3.5.10.2 Q22-Bus Timing -- The following paragraphs contain the timing diagrams for DMA gate array Q22-Bus master cycles. The "state" changes on the rising edge of CLOCK H. Timing diagrams for the slave-to-master and master-to-slave transceiver enable and direction signals are also provided.

Slave-to-Master Transition
Figure 3-67 shows the slave-to-master transition.

Master-to-Slave Transition
Figure 3-68 shows the master-to-slave transition.

Master DIN Cycle
Figure 3-69 shows the master DIN cycle.

1. For block-mode transfers, the DMA machine state changes from state 23 to state 7. For non-block-mode transfers, the DMA machine state changes from state 23 to state 0.

2. If REF H is asserted by the slave device during any DMA cycle, the DMA gate array attempts 16 transfers, provided the DMA byte count is greater than or equal to 31. If REF H is not asserted by the slave device during any transfers, the DMA gate array attempts a maximum of 8 transfers, provided the DMA byte count is greater than or equal to 7.

3. BUSRQ H and BUSDON H are internal signals and are presented on this timing diagram for reference only.

Master DOUT Cycle
Figure 3-70 shows the master DOUT cycle.

1. For block-mode transfers, the DMA machine state changes from state 23 to state 7. For non-block-mode transfers, the DMA machine state changes from state 23 to state 0.

2. If REF H is asserted by the slave device during any DMA cycle, the DMA gate array attempts 16 transfers, provided the DMA byte count is greater than or equal to 31. If REF H is not asserted by the slave device during any transfers, the DMA gate array attempts a maximum of 8 transfers, provided the DMA byte count is greater than or equal to 7.

3. BUSRQ H and BUSDON H are internal signals and are presented on this timing diagram for reference only.

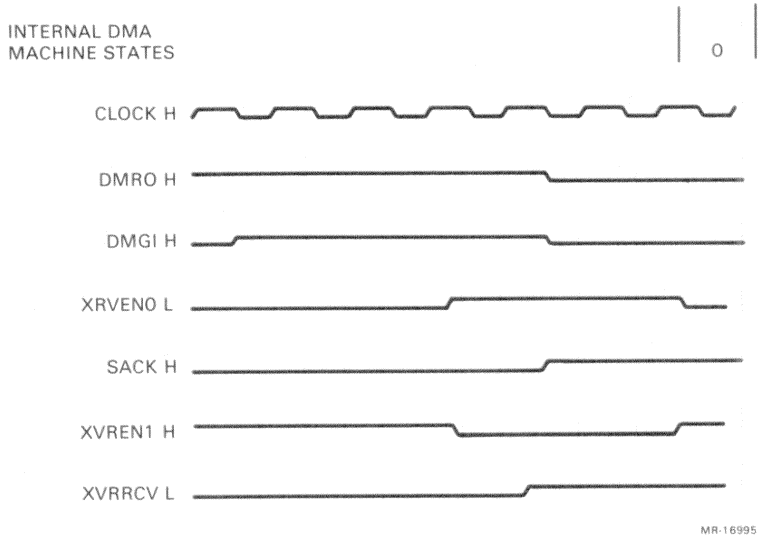


Figure 3-67 Q22-Bus Timing, Slave-to-Master Transition

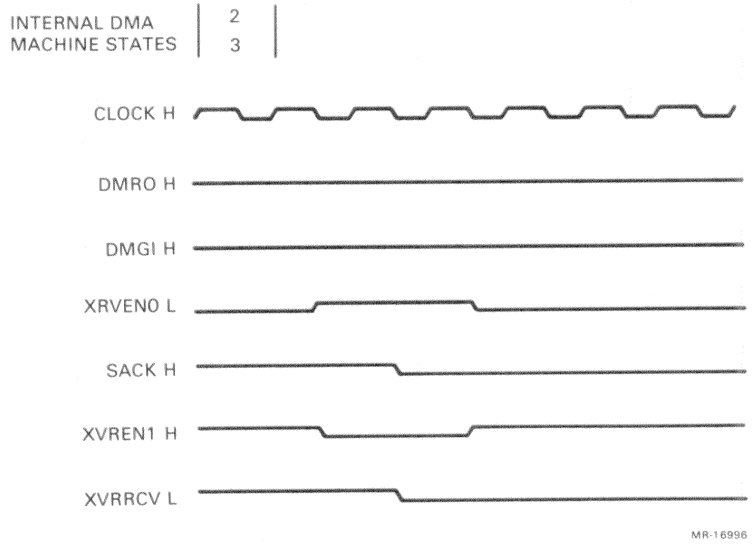


Figure 3-68 Q22-Bus Timing, Master-to-Slave Transition

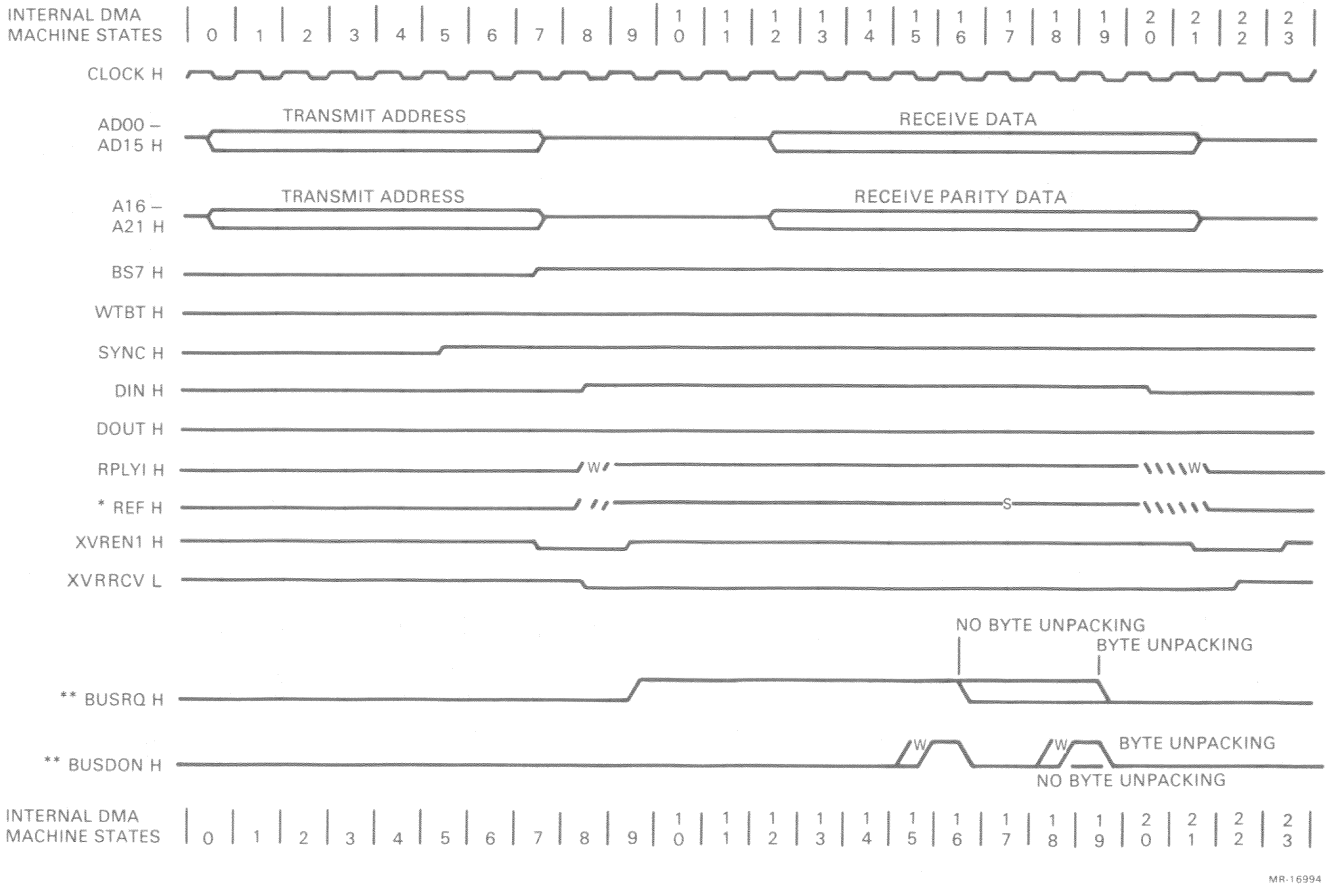


Figure 3-69 Q22-Bus Timing, Master DIN Cycle

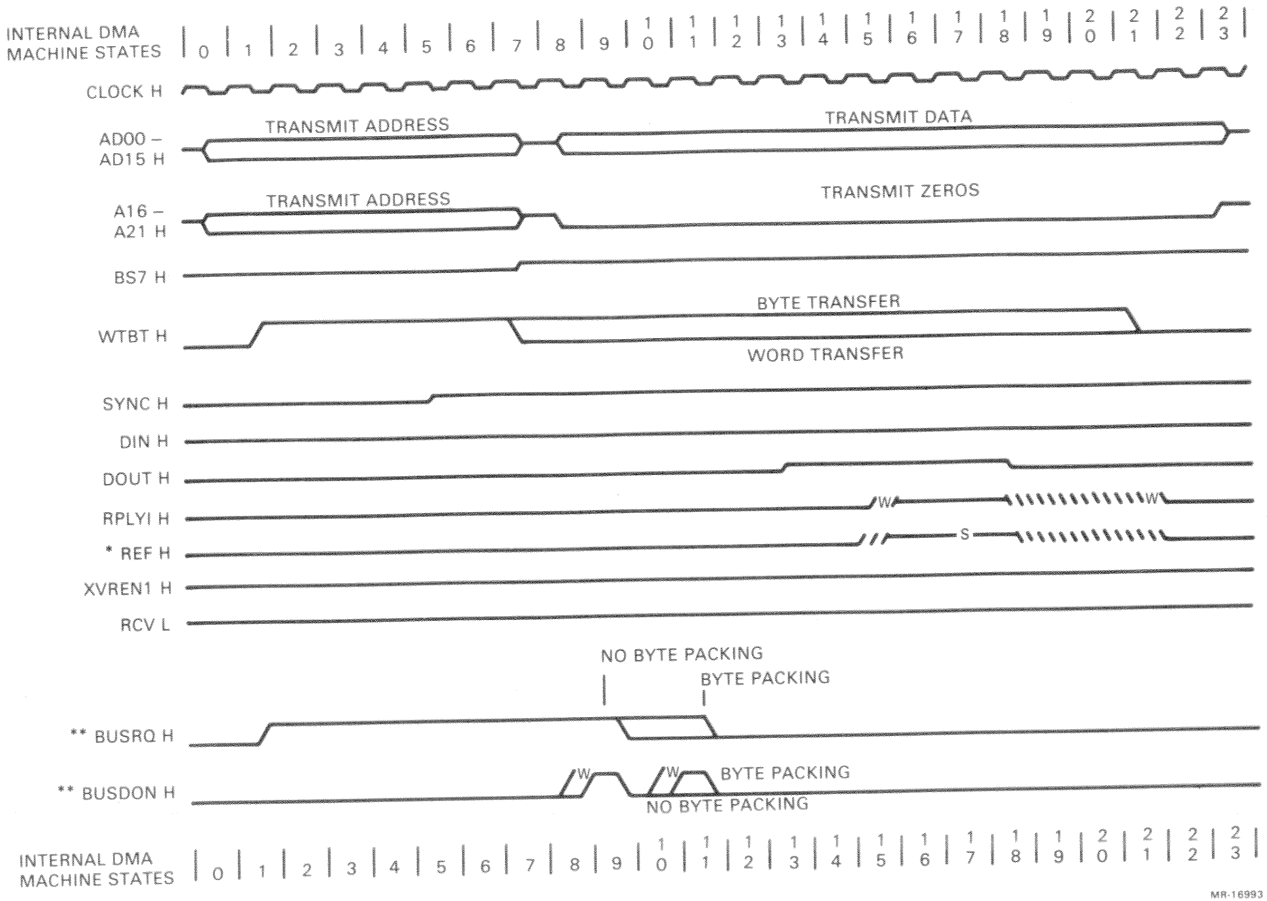


Figure 3-70 Q22-Bus Timing, Master DOUT Cycle

4.1 PROGRAMMING THE VCB02 VIDEO SUBSYSTEM

4.1.1 Hardware Support

The VCB02 video subsystem hardware supports the following features:

1. The ability to transform, manipulate, and create images through a set of bitmap operations called raster operations (rasterops) at rates between 0.5 million and 8 million pixels/s.

2. A dedicated parallel data path processor for each plane of physical memory, resulting in a maximum bandwidth of 256 million bits/s.

3. Viewports, by providing clipping, along with horizontal and vertical smooth scrolling within a rectangular region.

4. A text-writing rate of up to 20 thousand characters/s.

5. Tiling, pattern, and half-tone generation.

6. Polygon fill.

7. Z-axis read and write operations, to set the value of a pixel in a single operation.

8. DMA of address processor commands from MicroVAX memory to the address processor.

9. Storage and execution of address processor command "subroutines" in the VCB02 template RAM.

Figure 4-1 is a simplified block diagram of the VCB02 video subsystem. The essential subsystem components are:

1. Four or eight planes of bitmap memory. The bitmap is not directly accessible from the MicroVAX.

2. One video processor (data path) chip for each plane of bitmap memory. This chip receives input data from the bitmap or I/D interconnect, operates on it as directed by its control registers, and makes the result available for screen refresh and rewrite into the bitmap.

3. Chip select hardware that determines which set of video processor chips (bitmap planes) will participate in a scrolling or bitmap update operation.

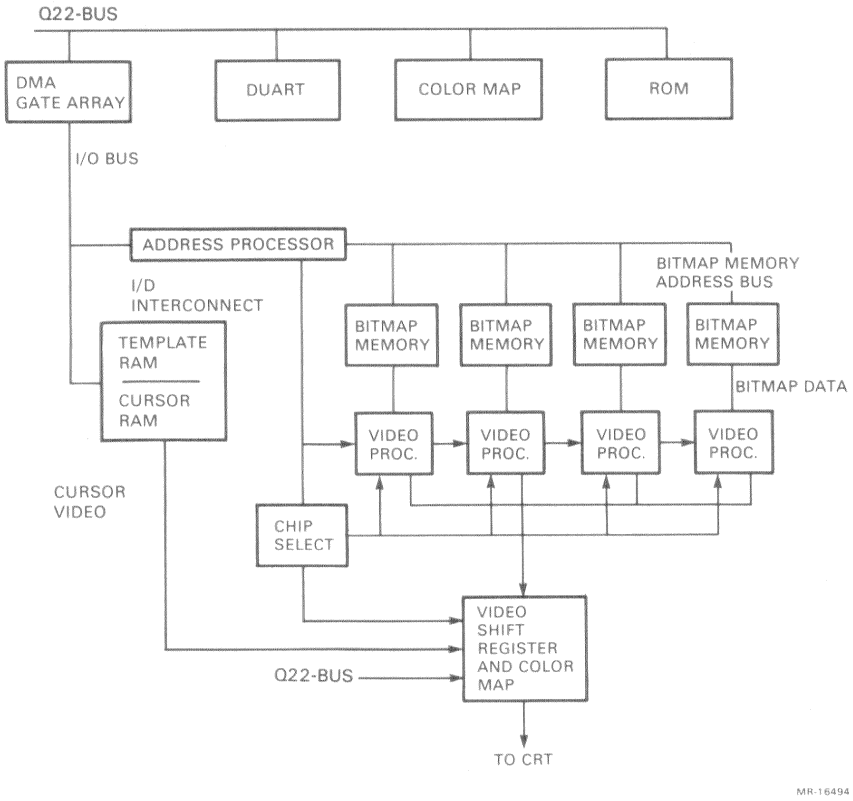


Figure 4-1 VCB02 Video Subsystem

4. The address processor chip (microsequencer); a custom LSI chip that implements the rasterop and scrolling algorithms. The outputs of this chip are memory addresses and commands to the video processor chips to perform rasterops, scrolling, and screen refresh.

5. The I/D interconnect connecting the address processor chip to all video processor chips. This interconnect is used to broadcast commands and data to all video processor chips, to transfer bitmap data from the video processor chips to the address processor chip, and to load the update and scroll chip select registers.

6. The DMA gate array, to accelerate the transfer of data and commands between the MicroVAX CPU and the address processor.

The VCB02 hardware executes a set of raster operations to manipulate and create images, and performs horizontal or vertical scrolling of any rectangular region on the screen. The VCB02 subsystem has a 2-plane programmable cursor, a 256 X 12 bit color map, support for a keyboard and a mouse, and a DMA interface.

4.1.2 Bitmap Memory Organization

Figure 4-2 shows each 2-dimensional plane of bitmap memory under control of a separate video chip.

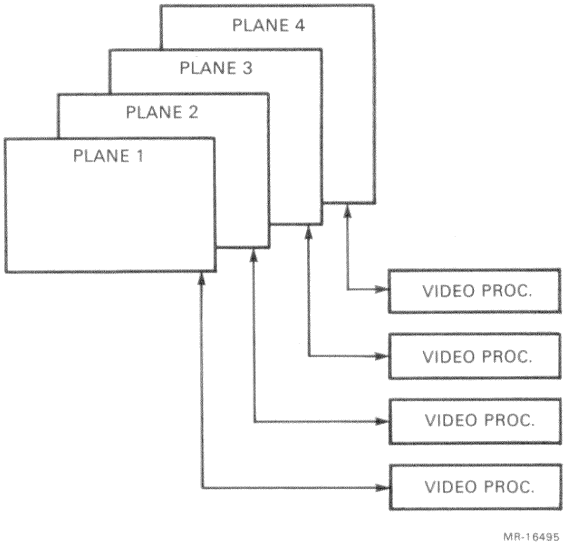


Figure 4-2 Multiplane Bitmap Memory Organization

A pixel is addressed by its Cartesian coordinates referenced to the upper left corner of the memory space. The X coordinate increases to the right; the Y coordinate increases downward. A bit within a plane of memory is addressed by its Cartesian coordinates and bitplane number.

As Figure 4-3 shows, the bitmap is divided into the screen refresh area and off-screen memory (which can be used for font and image storage). Each plane consists of 1024 X 2048 pixels; of this, 1024 X 864 are visible on the screen. The remainder is available for fonts and other off-screen images.

In addition to delineating the part of memory visible on the CRT, the screen refresh boundaries also mark that part of memory that can participate in scrolling. The actual scrolling boundaries are set in the address processor's scroll boundary registers. The hardware achieves the visual effect of scrolling by either moving data within the scroll boundaries, or in the case of down-scrolling, moving the data outside the scroll boundaries up and shifting the screen down with respect to the memory origin. In either case, the area outside the boundaries remains stationary, while the area within the boundaries scrolls.

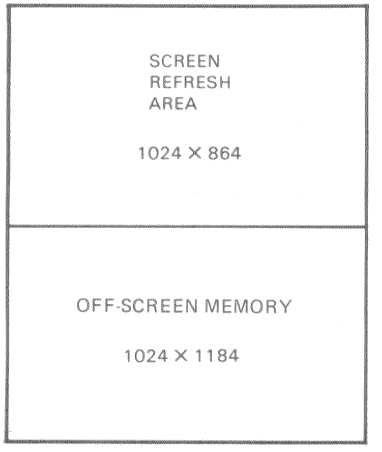


Figure 4-3 Off-Screen and Visible Memory

4.1.3 Overview of Rasterops

In the VCB02 video subsystem, all image processing in the bitmap is done by means of rasterops. A raster is a set of pixels within the bitmap that is enclosed by a parallelogram of any orientation. A raster operation (rasterop) transforms a rectangular source raster, aligned with the X- and Y-axis, into a destination raster. As shown in Figure 4-4, the transformation consists of both a geometric and a logical component. The geometric transformations consist of rotation, scaling, and translation. These are performed by the address processor chip. The logical component consists of masking, clipping, and combining logic, using pixels in the destination. These operations are performed by each video processor chip using codes placed in control registers by the MicroVAX CPU.

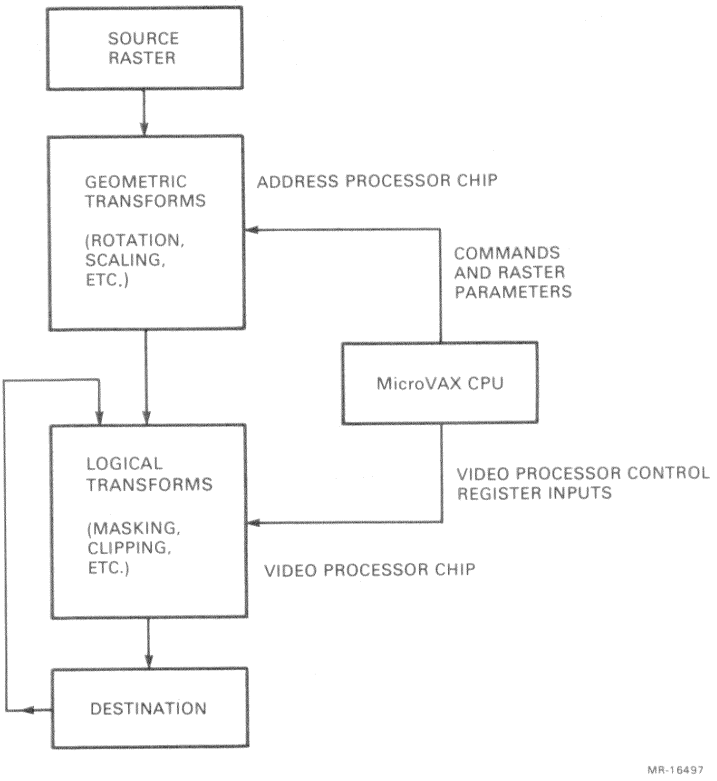


Figure 4-4 Rasterop Functional Model

In the general case shown in Figure 4-5, a rasterop is characterized by two source operands, a destination, a transformation between sources and destination, and a clipping rectangle applied to the destination raster. All operands are optional and all can be used in one of the following ways.

- As a source of pixels to be logically combined with the destination
- As a mask applied to other operands
- As a source of data to be broadcast on the I/D interconnect

In its simplest form, a rasterop copies a rectangular array of pixels from one location in the bitmap to another. For example, text may be displayed by selecting the appropriate raster from a set of rectangular font patterns in an off-screen portion of the bitmap and moving it to the location defined by the cursor position. In this case, the source is an array of pixels in the

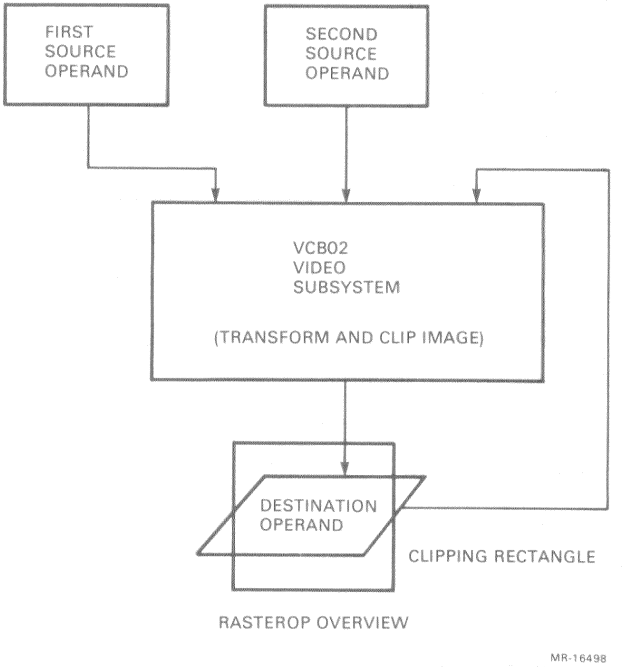


Figure 4-5 Rasterop Operand Flow

font table, the destination is a rectangular area located at the cursor position, and the transformation is a replacement of the destination with a copy of pixels from the source.

Other, more complex, transformations involve rotation, scaling, and some logical combination of source and destination pixels. The use of these additional features allows the rasterop facility to be used for vector generation and a large class of text and graphics operations.

The clipping rectangle facilitates the implementation of viewports. It is a mask that overlays the destination, preventing pixels outside the boundaries from being overwritten. The results of the clipping algorithm are available to the MicroVAX CPU to aid in operations such as selecting an object or a menu item. Resolution of the clipping boundaries is up to 1 bit vertically and 4 bits horizontally.

The performance of a raster operation depends largely on the orientation of the destination raster. With one edge aligned to the X direction, data is written into the destination in up to 16-bit increments, resulting in a maximum rate of 8 million pixels/s. If neither edge is aligned with the X-axis, data must be written one pixel at a time. Thus, the overall rate decreases to 1/16 of the fast mode, or 0.5 million pixels/s.

4.1.4 Overview of Scrolling

In the VCB02 video subsystem, horizontal and vertical scrolling is accomplished by high-speed logic within the address processor and video processor chips. The logic can move an entire 1024 X 905 (screen height plus blanking time) scroll region upward or horizontally in a single frame time. The scrolling boundaries may be set on any vertical pixel boundary and on a horizontal 4-pixel boundary. Up-scrolling within a rectangular region simply involves using the scrolling hardware to move the contents of the region upward by the scroll increment. Conversely, since the scrolling logic does not shift the region downward, the hardware creates the appearance of down-scrolling by leaving the data inside the scrolling region stationary and shifting the surrounding data upward. The illusion of down-scrolling is produced by offsetting the start of screen refresh in the Y direction to compensate for the physical movement of the "stationary" data.

One way to visualize down-scrolling is to consider a matte enclosed in a picture frame. If the matte is placed over a photo, it would outline some portion of the image. The image can be made to move downward with respect to the matte by either moving the photo downward or the matte upward. If the matte is moved, the appearance of the matte within the frame is held constant by shifting the matte and frame together. This is analogous to the way the VCB02 video subsystem performs down-scrolling. Relocating the start of the screen corresponds to shifting the picture frame.

In addition to scrolling, this logic may be used to "drag" a region across the screen. Dragging is performed by elongating the scrolling boundary in the direction the image is to be moved, leaving the fill color in its wake. Taking the previous example, it is analogous to moving the matte and photo together within the frame boundary. The horizontal scrolling rate may vary from 1 to 15 pixels/frame; the vertical rate may vary from 1 to 905 pixels(lines)/frame. Thus, smooth or jump scrolling may be performed in either the X or Y direction.

4.1.5 Coordinate Systems and Mapping

Images in the VCB02 video subsystem can be referenced by their Cartesian coordinates relative to:

- The origin of bitmap memory
- The origin of the screen
- Any arbitrary point in the bitmap (usually a viewport or scrolling region)
- The origin of the destination raster.

As a rule, applications will want to specify the location of objects in a region relative to the region origin. Visible objects outside a region will be specified in screen coordinates. Objects in off-screen memory will be located with respect to memory coordinates. This section explains how the address processor chip performs coordinate conversions between these frames of reference in the absence of scrolling.

Figure 4-6 shows the relationship of these spaces. Logically, the screen and memory coordinate systems overlay one another and have the same extent. The physical bitmap appears as a large rectangular coordinate system that is subdivided into the screen refresh area and off-screen memory. The mapping hardware makes it seem as if the origin of the screen is always coincident with the bitmap.

Because of vertical scrolling, the actual position of the screen refresh area is displaced in the Y direction from the memory origin, as shown in Figure 4-7. The extent of this area is defined by the contents of X and Y limit registers in the address processor chip. The shaded area indicates the visible portion of memory. The unshaded part is the off-screen area reserved for scrolling. Down-scrolling is actually performed by displacing the screen origin relative to the start of physical memory. This vertical displacement is contained in the Y offset register in the address processor chip.

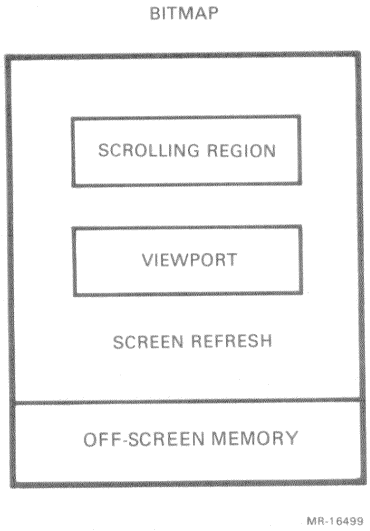


Figure 4-6 Viewport and Scrolling Regions

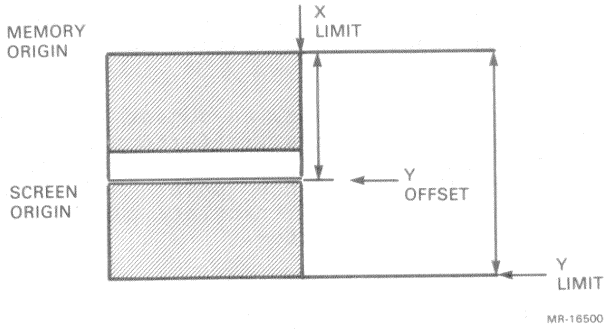


Figure 4-7 Screen-to-Memory Mapping

During screen refresh, or when a rasterop address inside the screen refresh limits is generated, the address is automatically mapped to physical memory by the following algorithm.

```
ROUTINE SCREEN_TO_MEMORY_TRANSFORM (X,Y; X_MEMORY,Y_MEMORY)

!
! Screen to memory coordinate transformation
!
! X and Y are addresses generated by either the screen
! refresh or rasterop processes.
!
! X_MEMORY and Y_MEMORY are the physical memory coordinates
Y_MEMORY = Y
X_MEMORY = X
IF ((X is less than or equal to X_LIMIT) AND
(Y is less than or equal to Y_LIMIT)) THEN
Y_MEMORY = (Y + Y_OFFSET) MODULO (Y_LIMIT)

RETURN
```

Note the cylindrical screen-to-memory mapping that results from the use of modulo arithmetic.

Figure 4-8 shows the relationship of the scrolling region to the screen coordinate system.

The address processor chip locates the scrolling region with respect to the screen origin by the four X and Y coordinates shown. The clipping region is specified to the scrolling region similarly.

To support viewports and windows, an object (raster) can have an address relative to an arbitrary reference point in the screen coordinate system. The address processor can conditionally apply the coordinates of the reference point and the X and Y index values (shown in Figure 4-9) to translate a destination or first source address to screen coordinates.

If the translated value is inside the refresh area, the Y offset is automatically applied to convert the address to memory coordinates. As the following algorithm shows, operand indexing and the index values are specified by the MicroVAX CPU.

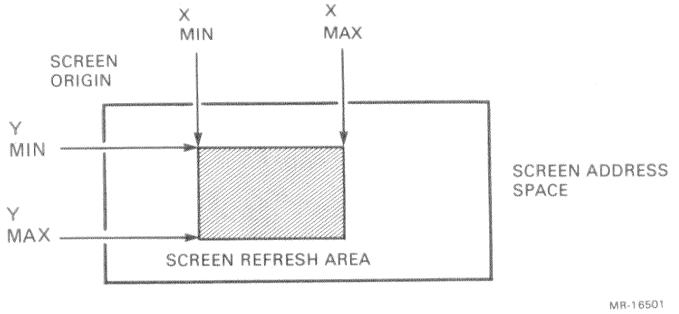


Figure 4-8 Scrolling Region

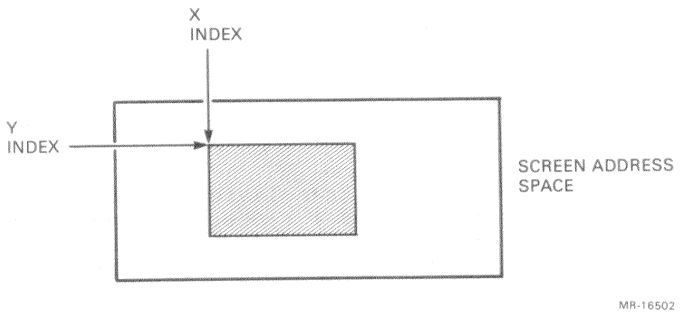


Figure 4-9 Address Indexing

```
ROUTINE REGION_TO_MEMORY_TRANSFORM (X, Y; X_MEMORY, Y_MEMORY)

!
! X and Y are the object coordinates relative to the
! start of the scrolling region.
!
! First assume that indexing is disabled
!
X_SCREEN = X
Y_SCREEN = Y
IF (Indexing is enabled for this raster operand) THEN BEGIN
X_SCREEN = X + X_INDEX
Y_SCREEN = Y + Y_INDEX
END

!
! If the result is inside the screen refresh area, apply
! the screen to memory coordinate transform shown
! earlier.
!
SCREEN_TO_MEMORY_TRANSFORM (X_SCREEN,Y_SCREEN; X_MEMORY,Y_MEMORY)
RETURN
```

Indexing may be selectively enabled only for the source 1 and destination operands.

Figure 4-10 summarizes address translation for the screen refresh process, destination, and source 1 raster operands. The output of the rasterop computation is an address in screen, memory, or region coordinates. The interpretation is specified, in part, by the MicroVAX CPU, which selectively enables indexing for either the first source or destination operands. Region coordinates cannot be used for the second source operand. The result, after indexing, may be further mapped by the address processor, if it is inside the screen refresh area.

When tiling is disabled, the origin of the destination raster defines the coordinate system for the second source operand. In this case, the MicroVAX CPU specifies the second source operand in terms of X and Y offsets from the destination. Figure 4-11 shows the computation of the second source physical memory address.

4.1.5.1 Interactions between Rasterops and Scrolling -- The effects of concurrent scrolling and rasterops are twofold. First, the process of down-scrolling may alter the screen-to-memory mapping for data that appears stationary on the screen. Second, memory references by the rasterop process may "collide" with the scrolling process. These interactions are explained below.

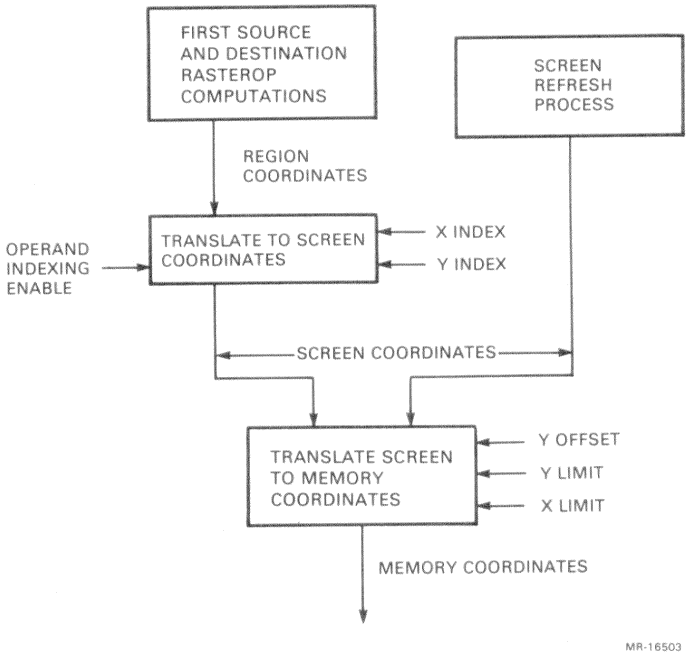


Figure 4-10 Address Translation

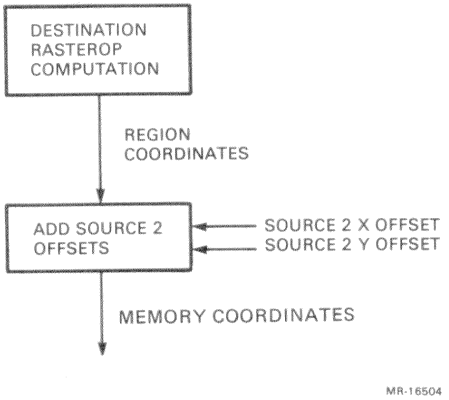


Figure 4-11 Source 2 Address Computation (Tiling Disabled)

Effect of Scrolling on Screen and Region Mapping

The bitmap is divided into the screen refresh area, the scrolling region, viewport area, and off-screen memory. The address processor chip performs the following coordinate transformations.

- Screen-to-memory mapping, using the Y offset to map visible objects to physical memory.
- Region-to-screen mapping, using the X and Y index values.

If scrolling is ignored, the coordinate conversions can be performed by simply adding the appropriate constants. The effect of scrolling is threefold. First, scrolling or dragging a viewport effectively changes its location with respect to the screen origin; therefore, the index values must be adjusted accordingly. Second, down-scrolling requires adjustment of the screen-to-memory Y offset. Third, when rasterops and scrolling are performed concurrently, the use of a single Y offset or set of indexes for coordinate mapping no longer works.

To see why a single coordinate mapping fails during scrolling, recall that scrolling involves the movement of some portion of the visible data within the time required to scan one complete frame. During that time, the scrolling hardware makes a sweep of the bitmap in the downward direction, accessing the data to be moved one scan line at a time. At any instant, then, the screen is divided into two parts: the part above the current scan, consisting of data already moved; and the part below the scan, consisting of data not yet moved. Thus, during this time, two mappings exist between the screen and the bitmap: one for shifted data above the scan, and another for unshifted data. Therefore, without the features described below, "scrambling" would occur due to pixels being placed incorrectly in the destination raster or fetched incorrectly from the source.

The hardware eliminates this scrambling effect by providing an extra set of Y offset and index registers. These extra registers contain the new mapping for shifted data. They are automatically referenced by the address processor chip when an address is generated in the scrolled part of the screen refresh area. The old mapping is automatically used for the unscrolled portion.

The task of updating these registers is left to the MicroVAX CPU. To allow a full frame time for this operation, a third set of index and offset registers is provided to hold a set of pending values. The index values are in the order X coordinate, Y coordinate. At the completion of a frame, the address processor chip automatically advances the register values. Since all data has been scrolled, the "new" values now represent the mapping for unscrolled data, and are therefore copied to the "old" registers. The pending values become the new values for scrolled data. The MicroVAX CPU has a full frame time to compute and set up a new set of pending values for the next frame. The flow of data for this pipeline is shown in Figure 4-12.

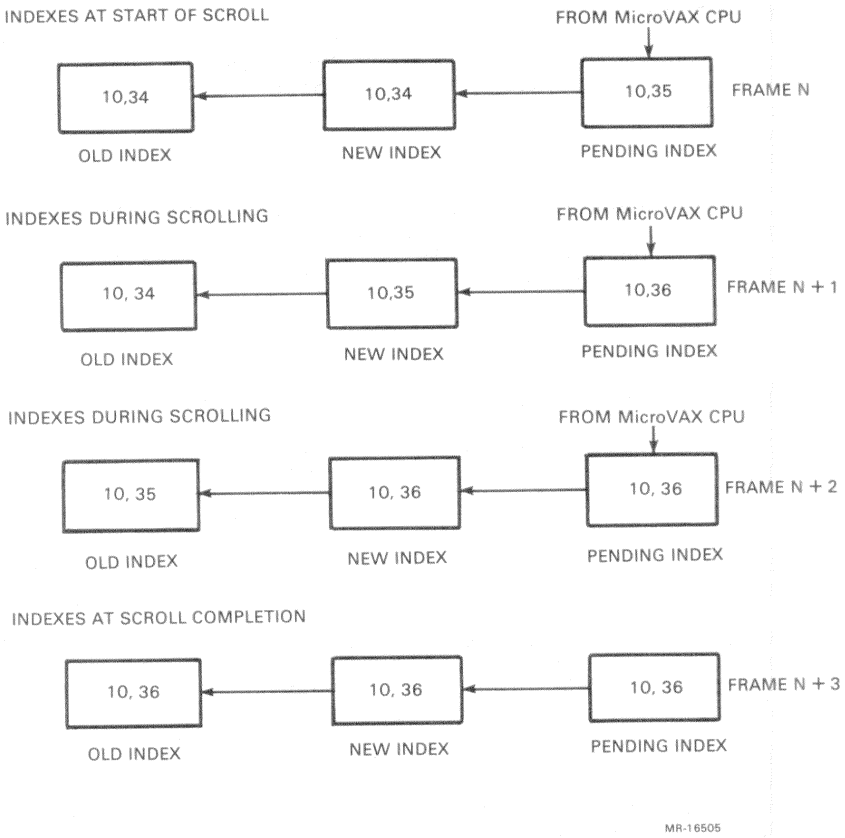


Figure 4-12 Index Registers for 2-Pixel Down-Scrolling

Note that at the completion of scrolling, the pending index value is simply allowed to propagate to the "old" and "new" register sets. The index represents the screen coordinates of the upper left corner of a region. Scrolling or dragging a viewport effectively changes its location with respect to the screen origin; therefore, the index values must be adjusted accordingly. Conversely, if the viewport is not being scrolled, the index values are unchanged.

Collisions between Rasterops and Scrolling

A collision occurs whenever a rasterop and the scroll process try to read or write the same memory locations in the bitmap. The hardware resolves collisions by suspending the rasterop until the scrolling process has moved on. Since scrolling accesses memory from the top of the screen to the bottom, the worst case for collisions occurs when the processing of one or more raster operands scans the screen refresh area in the downward direction. In this case, rasterop memory references, trailing behind the scroll process, repeatedly collide with scroll memory accesses. A way to limit such collisions is to specify the raster operand so that it is scanned in the upward direction. In this way, only one collision per frame can occur during a rasterop.

4.1.6 Additional Operations on Bitmap Data

In addition to conventional raster operations and scrolling, the following functions are supported by the hardware.

- Polygon fill, to fill the area between two vectors with any source of pixels, such as a solid color, tile pattern, or image.

- Transferring data between the MicroVAX CPU and one plane in the bitmap, to load a font or save and restore an occluded viewport.

- A set of Z-axis operations, for transferring one pixel value between the MicroVAX CPU and the bitmap in a single operation. This function is used to facilitate pixel operations that are not supported by the hardware, such as flooding or painting within the boundaries of a region having an arbitrary shape. In this case, the VCB02 hardware maps a pixel vector into a single 16-bit word to be read or written into the MicroVAX's address space. The VCB02 can map two pixel vectors into one 16-bit word when doing DMA.

The operations above are all implemented as variations on the basic rasterop function.

4.2 VIDEO PROCESSOR AND ADDRESS PROCESSOR CHIP ARCHITECTURE

The VCB02 video subsystem architecture divides all display operations into two orthogonal functions: the generation of addresses for bitmap data, and logical operations on the data. Address generation is performed by the address processor chip. Logical operations on the data are performed by the video processor chip.

The functional relationship between the address processor and video processor chips is shown in Figure 4-13. The address processor chip broadcasts memory addresses to the bitmap commands and data to each video processor chip on the I/D interconnect. Each video processor chip, running in lockstep, processes the bitmap data as directed by instructions on the I/D interconnect, control signals, and instructions in its control registers.

The I/D interconnect protocol allows any device to become bus master. Therefore, during raster operations, the video processor chips can serve as sources or destinations for operands. This is useful for broadcasting common data, such as character fonts, to all the bit planes. Of course, only one video processor chip can serve as the source of data at any one time, while any number of chips can receive data.

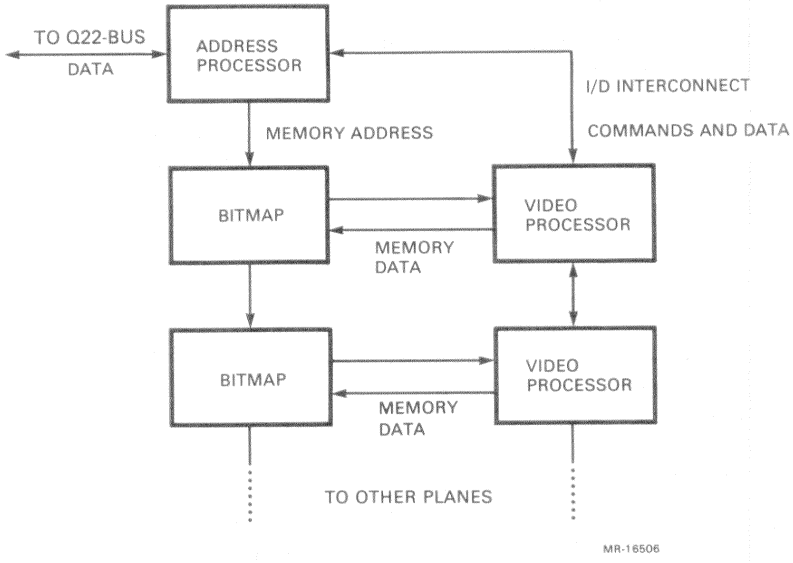


Figure 4-13 Address Processor and Video Processor Chips, Functional Diagram

4.2.1 Address Processor Chip Architecture

The address processor chip logically supports three separate and concurrent processes:

1. CRT screen refresh
2. Scrolling
3. Raster operation and processor-to-bitmap transfers.

Figure 4-14 shows the relationships of these processes.

Each process, running concurrently, generates a bitmap address and an I/D interconnect command. The arbitrator resolves contention for access to shared resources, such as the I/D interconnect and bitmap, based on the following requirements.

- Screen refresh cannot be interrupted.
- Scrolling must not be visually disturbed.
- Race conditions must be eliminated between the scroll and rasterop processes.

Both screen refresh and scrolling share memory access on a "round-robin" basis, with the remaining time allocated for rasterops. During simultaneous raster operations and scrolling, the arbitrator blocks the rasterop process when its memory accesses conflict with those of the scrolling process.

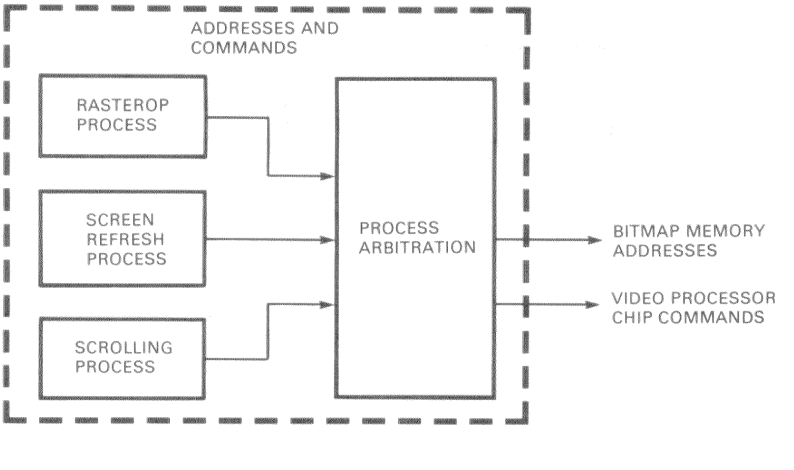


Figure 4-14 Address Processor Chip Data Flow

4.2.2 Address Processor Chip Interface to the Q22-Bus

The address processor chip is the gateway to the video processor chip, bitmap, and chip select registers. As such, it provides access to this hardware through a set of register-level interfaces for commands and the interchange of data and status via the Q22-Bus. The register format and addressing is also designed to be an efficient interface to the DMA gate array.

4.2.3 Video Processor Chip Architecture

The function of the video processor chip is to process data appearing at its input pins as directed by instructions in its control registers, commands on the I/D interconnect, and hard-wired signals from the address processor chip. Logically, the video processor chip has two data paths: one for rasterops, the other for scrolling and screen refresh.

4.2.3.1 Data Flow for Scrolling and Screen Refresh -- Figure 4-15 shows data flow through the video processor chip for scrolling and refresh.

The video processor chip reads 128 bits from the bitmap quickly, storing the 128 bits in its internal FIFO. This allows the video processor to output the data on its video bus at a slower rate, while the memory bus is being used for updates and scrolling, until another 128 bits is required.

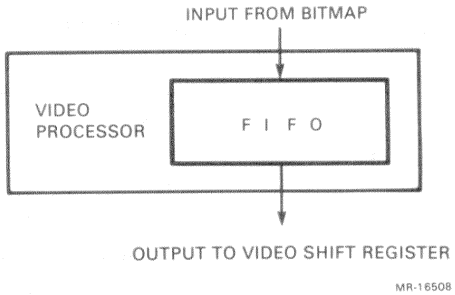


Figure 4-15 Video Processor Data Flow (Screen Refresh/Scrolling)

4.2.3.2 **Rasterop Data Flow** -- Figure 4-16 shows the data flow for a rasterop. Operands are read and processed as directed by internal control registers by the following functional units.

- Operand fetch logic
- Combinatorial logic
- Masking and clipping logic

Each rasterop cycle consists of the following events.

1. The first and second source operands are read in sequence and routed to the appropriate functional unit, as directed by the control store RAM.

2. The destination is read and processed, also; it automatically becomes the second operand to the logic unit.

3. The logic function specified by the video processor control registers is performed.

4. The output of the logic unit is masked, clipped, and written into the destination.

Steps 2, 3, and 4 are actually executed in a single read-modify-write memory cycle.

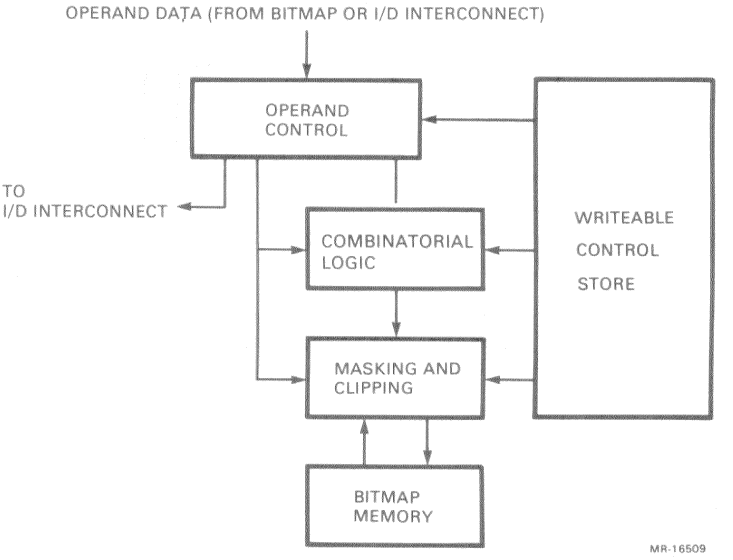


Figure 4-16 Rasterop Data Flow

Video Processor Chip Operand Fetch

The operand fetch logic routes the operand to the appropriate functional logic within the video processor chip, and optionally, to broadcast data on the I/D interconnect.

Figure 4-17 shows how operands are processed. An operand fetch cycle is actually divided into two microcycles. During the first, internal input data from the bitmap is processed. During the second, external data from the I/D interconnect is read, and optionally, a copy of the internal operand can be gated onto the interconnect.

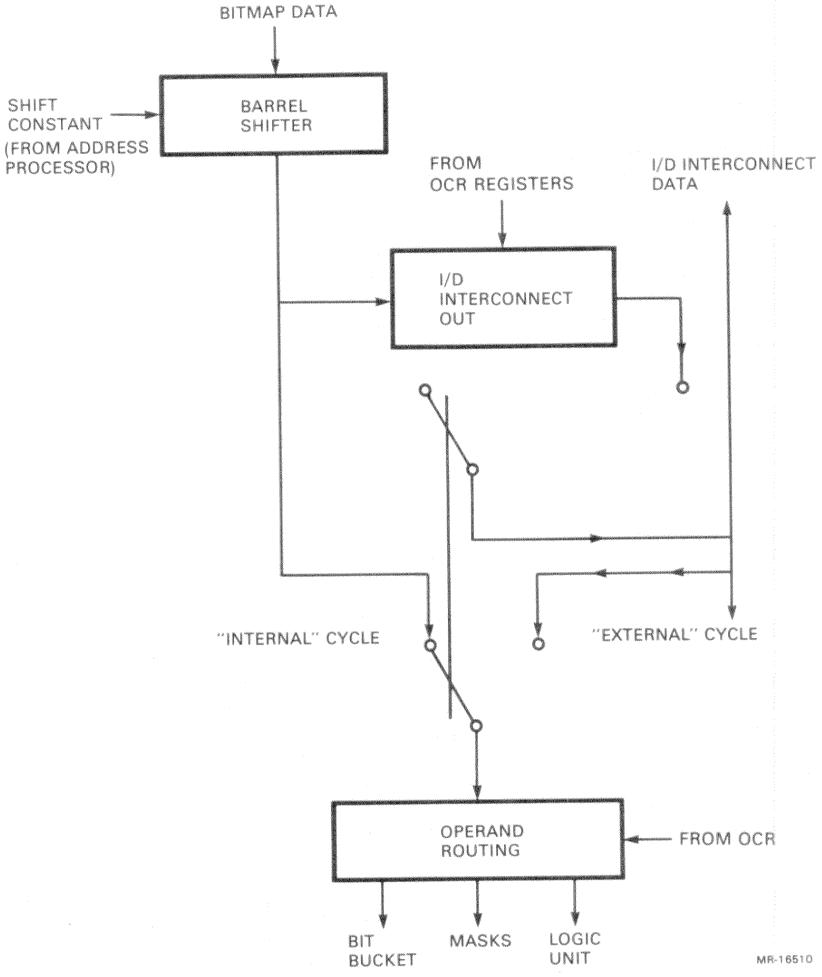


Figure 4-17 Operand Routing Functions

In Figure 4-17, the ganged switches indicate the data paths for each microcycle. The switch state shown is for the internal cycle. During this time, data from the bitmap is read, rotated into alignment with the destination raster, and directed to a functional unit.

During the external cycle, data from the I/D interconnect is also read and passed, without rotation, to the destination. Concurrently, a copy of the aligned internal operand can be output to the I/D interconnect. When this occurs, the output becomes the external operand for all video processor chips, including the chip sourcing the data.

The control store RAM specifies:

- The destination of the internal operand
- The destination of the external operand
- Whether or not a copy of the internal operand will be placed on the I/D interconnect.

The Logic Unit

The logic unit is shown in Figure 4.18. The foreground, background, and source registers are loaded by the MicroVAX CPU. The source register can also be loaded with bitmap data by the video processor. The logic unit combines data from the bitmap with data from the source register in a way determined by the logic function registers. The output of the logic unit is used to select either a foreground or background value. An output of 1 causes the selection of a foreground value (color), an output of 0 causes the selection of a background value (color). The selected foreground or background value is further manipulated by the clipping and masking logic.

Masking and Clipping Logic

The masking and clipping section of the chip is shown in Figure 4-19.

The mask registers can be set up with operands during each fetch microcycle by programming the appropriate operand destinations into the control store RAM. In most cases, both mask registers will require the same value; therefore, sending an operand to mask 1 automatically causes a copy to be written into mask 2. To set different mask values, the control store RAM is programmed to write mask 1, followed by mask 2. Each mask is also controlled by a register that allows the mask register, or its complement, to be used to form the mask value.

A mask value of 0 in a given bit position specifies that the corresponding bitmap location is to be unmodified. A 1 indicates that the location is to be replaced with the output of the logic unit.

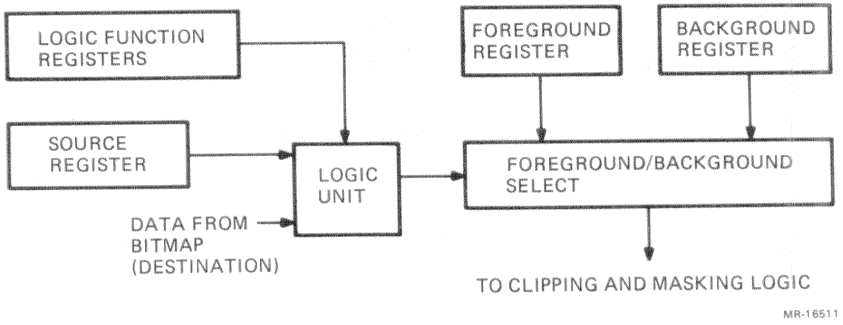


Figure 4-18 Logic Unit

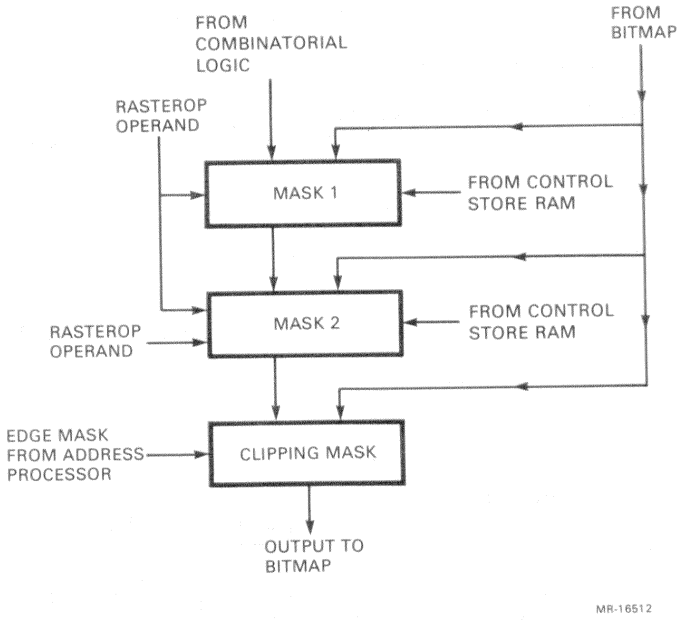


Figure 4-19 Masking and Clipping

Video Processor Chip Control Registers

The video processor chip contains two sets of registers that can be programmed by the MicroVAX CPU to control the execution of raster operations:

- Control store RAM (CSRs) that specify how operands will be transferred
- Logic and mask control registers that determine the function of the logic unit and the polarity of the mask register output.

Control Store RAM Functions

The operand fetch logic is controlled by a set of six control store RAMs divided into two banks of three locations. Within each bank, one location controls each operand. Every time a rasterop is performed, the MicroVAX system must specify the bank of control store RAM to be used. In this way, the software can preset two individual banks, which permits operations requiring two rasterop steps without need of reloading RAM locations.

The following is a brief description of the control fields.

- Internal cycle destination -- Specifies one of the following destinations for the internal operand:
  - The bit bucket
  - The source register
  - Both mask 1 and mask 2
  - Mask 2 only.
- External cycle destination -- Specifies one of the following destinations for the external operand:
  - The bit bucket
  - The source register
  - Both mask 1 and mask 2
  - Mask 2 only.
- I/D interconnect output control -- If set, places a copy of the aligned internal operand on the I/D interconnect during the external operand microcycle.

Note that every operand fetch cycle involves the execution of an internal and an external microcycle. Therefore, for all CSRs referenced by an operation, the disposition of data during both microcycles must be specified explicitly. If the data from either cycle is unwanted, the appropriate destination field to place the result in the bit bucket must be set.

Logic Unit Function Registers

The logic function control registers determine the operation performed by the logic unit and the polarity of the mask register output applied to the result. Four registers are available, and can be preset by the MicroVAX CPU, but only one is used for a given operation. The register number is a parameter of the rasterop command.

The logic unit operates on the output of the background/foreground select register and the contents of the destination raster. The masks applied to the output determine which pixels in the destination raster will be modified.

The logic function control registers have the following fields.

- Logic unit function -- One of the following functions can be programmed.

  - Output = Zeros
  - Output = NOT (Destination OR SOURCE)
  - Output = NOT (Destination) AND SOURCE
  - Output = NOT (Destination)
  - Output = Destination AND NOT (SOURCE)
  - Output = NOT (SOURCE)
  - Output = Destination XOR SOURCE
  - Output = NOT (Destination AND SOURCE)
  - Output = Destination AND SOURCE
  - Output = NOT (Destination XOR SOURCE)
  - Output = SOURCE
  - Output = NOT (Destination) OR SOURCE
  - Output = Destination
  - Output = Destination OR NOT SOURCE
  - Output = Destination OR SOURCE
  - Output = Ones

- Mask register 1 control -- If set, use the complement of mask register 1.

- Mask register 2 control -- If set, use the complement of mask register 2.

- Source register control -- If clear, use the complement of the source register.

- Resolution mode logic for source register control -- If clear, enables resolution mode logic for the source register.

4.2.4 I/D Interconnect Protocol

The I/D interconnect is a multidrop interconnect that can be used by the address processor chip and any video processor chip to send and receive data. Update and scroll ship select registers are connected to the interconnect, and specific address processor commands are available for sending data to them.

The role of the address processor in an I/D interconnect transaction is implicitly determined by the function being performed. During a processor-to-bitmap transfer, for instance, the address processor chip transmits data from the I/D interconnect to one or more video processor chips. During a bitmap-to-processor transfer, it acts as a receiver. Otherwise, unless specified explicitly, the address processor chip neither sends, or receives data.

The role of each video processor chip is determined by the contents of its control store RAM and logic function registers. During a processor-to-bitmap transfer, for example, at least one video processor chip must be programmed to receive external data during the appropriate operand cycle. During a full-plane bitmap-to-processor transfer, only one video processor chip will be programmed to source data on the I/D interconnect. During a Z-axis transfer, when part or all of a pixel is being sent to the MicroVAX CPU, each video processor chip will be programmed to "source" its portion of the pixel value at the appropriate bit position on the interconnect. This is the only case where multiple sources are allowed (with the restriction that no two video processor chips attempt to drive the same bit position simultaneously). Broadcasting data from one video processor chip to others during a rasterop requires the following actions.

1. The processor selects a memory plane and its associated video processor chip to serve as the source of data for broadcast to other video processor chips during a raster operation.

2. The appropriate operands are defined, enclosing the data to be broadcast (first source, second source, or destination).

3. For the chip broadcasting the data, the CSRs associated with the selected operands are programmed to place a copy of the data on the interconnect during the external microcycle.

4. For each chip receiving the data, the CSRs associated with the selected operands are programmed to receive data during the external microcycle.

For a given raster operation, the configuration of receivers and transmitter can be different for each operand. For example, in a system with four planes, numbered 0 through 3, the configuration for each operand cycle might be as follows.

Cycle	Transmitter	Receiver
Source 1	Video processor chip 0	Video processor chips 1, 2
Source 2	Video processor chip 2	Video processor chip 0

NOTE
The hardware does not prevent multiple writers on the I/D interconnect.

The programmer must ensure that multiple writers do not attempt to place data on the I/D interconnect "at the same time"; that is, during the same operand cycle. If this occurs, the contents of the interconnect will be garbled, and electrical failure of the input/output circuits is possible.

4.2.5 Rasterop Process

This section discusses the algorithms the address processor chip uses to generate operand addresses.

4.2.5.1 Raster Scanning Algorithm -- Figure 4-20 shows a raster defined by an origin and two vectors that enclose an area containing the pixels to be processed.

The algorithm for generating pixel addresses within the raster involves moving one vector along a path traced out by the other vector. The vector defining the path is called the "slow" vector; it is traversed only once per scan. The other edge vector, called the "fast" vector, is traversed at each point along the path traced out by the slow vector. Points along each vector are computed using Bresenham's algorithm.

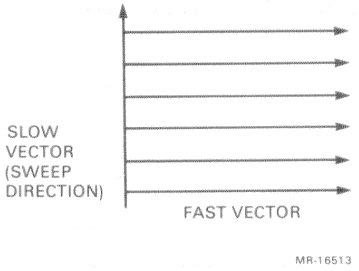


Figure 4-20 Raster Scanning

To allow an image to be created with multiple raster operations, the vector computation excludes the last point. This feature is intended to support "complement mode" writing, in which an image is erased by writing its complement back into the bitmap. For this to work, all pixels in an image must be written an odd number of times. If vector generation included the last point, the first point of the adjoining raster would trace over the boundary a second time, leaving a residue of unerased pixels.

For an ideal scanning algorithm, each pixel in the raster would be addressed only once during a raster operation, regardless of the orientation of the edge vectors. In fact, under certain conditions, one of the following will occur.

1. A pixel within the raster boundaries will be addressed more than once during a rasterop.

2. A pixel within the raster boundaries will not be addressed at all during a rasterop.

These conditions never happen when a vector is parallel to the X- or Y-axis, or at right angles to the other edge vector.

Failure to address a pixel will leave holes in the resulting image. The appearance of the image can be partially corrected by a hardware feature that, optionally, fills such holes. Hole fill is particularly useful when text is written into the bitmap by rotating and transforming a rectangular font.

Addressing a pixel more than once can affect the process of erasing an image by writing its complement into the bitmap. In such cases, it is advisable to first create the image off screen, then manipulate it with a rectangular rasterop.

Other kinds of distortion arise when a rectangular raster is scaled, rotated, or transformed into a parallelogram with angles other than 90 degrees. In these cases, information may be lost because the number of pixels in the destination raster is not the same as in the source.

4.2.5.2 **Specification of Operands for a Rasterop** -- A rasterop may have the following operands.

- A destination raster, with fast and slow vectors of any size and orientation.

- A primary source raster, called source 1, that can be used for spatial transformations on a rectangular raster (rotation, scaling, etc.) and linear pattern generation. The fast vector is always parallel to the Y-axis.

- A secondary source raster, called source 2, that can be used to copy a raster of any shape or orientation, or generate a tile pattern.

All operands are optional. The address processor chip allows the generation of a specific operand raster to be suppressed, when it is not required for a given function. In this case, the contents of the associated control store RAM locations in the video processor chip are not referenced, and therefore, need not be set up by the MicroVAX CPU.

Destination Operand
The destination operand is specified by the following parameters.

- The coordinates of the origin.
- The coordinate system used (either screen or region coordinates).
- The slope and extent (DX and DY) of the slow vector.
- The slope and extent (DX and DY) of the fast vector.

Region coordinates may be used when a raster is within a region or viewport. In this case, before executing the rasterop, the MicroVAX CPU must set up the address processor chip index registers with the values required to translate the region coordinates to screen coordinates.

First Source Operand
The first source operand is a rectangle with edges parallel to the X- and Y-axis. The fast vector is in the Y direction. The raster is defined by the following parameters.

- The location of the origin.
- The direction of the fast and slow vectors (+ or -).
- The coordinate system used (either region or screen coordinates).
- A scale factor relating the length of fast source to fast destination vectors.
- A scale factor relating the length of slow source to slow destination vectors.
- When linear patterns are generated, the length of the pattern is in the X and Y directions.

Mapping and Scaling the Source 1 Operand

During each rasterop cycle, the address processor chip generates an address in the source 1 raster, followed by an address for the destination. Within that cycle, the video processor chip combines the pixels at the source and destination as directed by its control registers. The path traced by the source and destination address pairs over time determines the source-to-destination mapping. Figure 4-21, with pixels scaled in dimension by a factor of a thousand or so, shows this relationship for the simple case of two rectangular rasters having a width of one pixel and a scale factor of one. The letters indicate pixels being copied from the source to the destination.

The numbers indicate the time sequence in which each pair of source and destination addresses is computed. At a scale factor of 1, a new source address is generated in step with each new destination address.

Figure 4-22 shows the mapping between the first source and a destination vector at some angle with respect to the X-axis.

Observe that rotation has introduced an aberration in the length of the destination vector. This distortion can be be partially corrected by introducing a scale factor between the source and destination.

To clarify the scaling process, recall that at a scale factor of 1, precisely 1 new point in the source is sampled for each new point in the destination. That is, the source and destination addresses both have the same rate of change. A change in scaling is produced by increasing or decreasing the rate of change between the source and destination addresses.

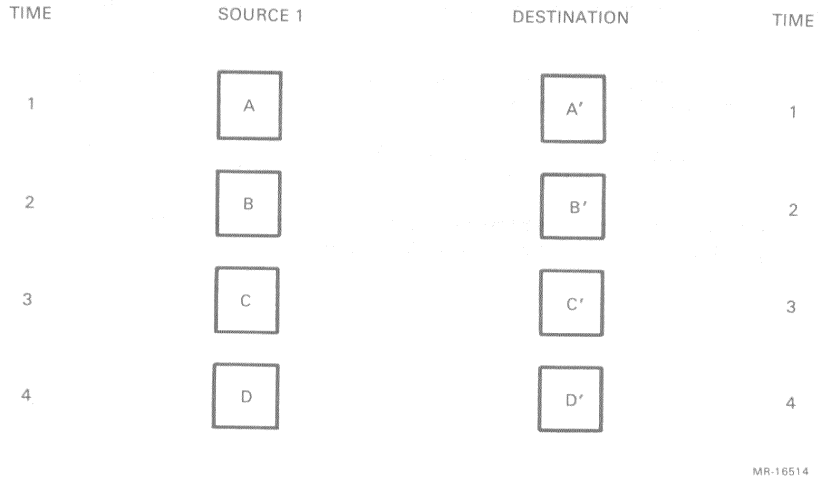


Figure 4-21 Simple Rasterop

Figure 4-23 shows the variation of source and destination addresses with time when the source is scaled up by a factor of 2. The numbers indicate the time sequence of each sample. A new source sample is taken on every other destination sample, effectively stretching each source pixel by the scale factor.

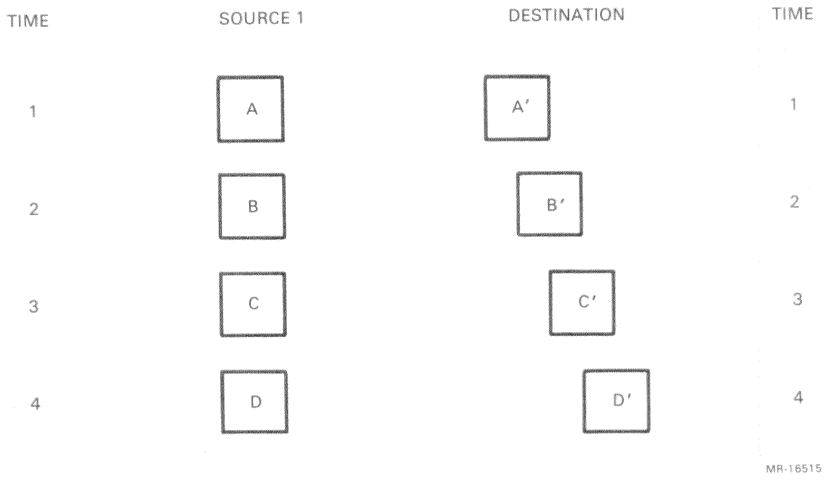


Figure 4-22 Rasterop with Rotation

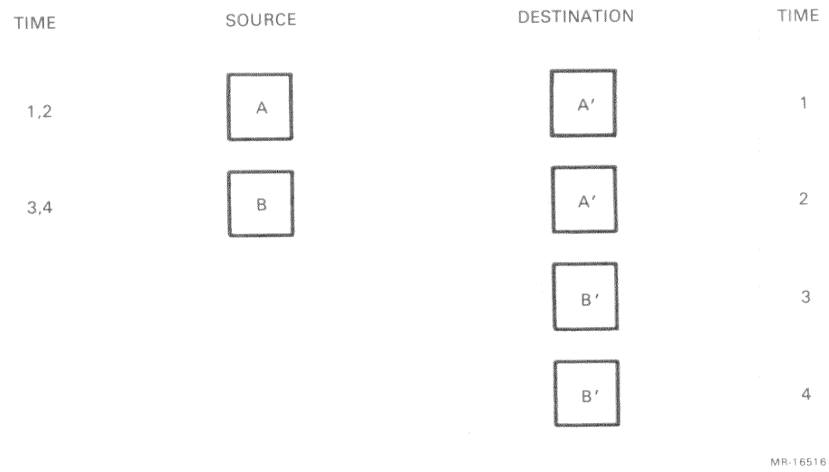


Figure 4-23 Source Scaled Up by a Factor of Two

Figure 4-24 shows what happens when the source is scaled down by a factor of 2. At time 1, the address of the source points to pixel "A". It is copied to the destination as shown. At time 2, the source address is incremented to pixel "B". The destination address is unchanged. The copy function overwrites the destination with "B'". Because the destination address changes at half the rate of the source, two source pixels are compressed into one destination pixel as shown.

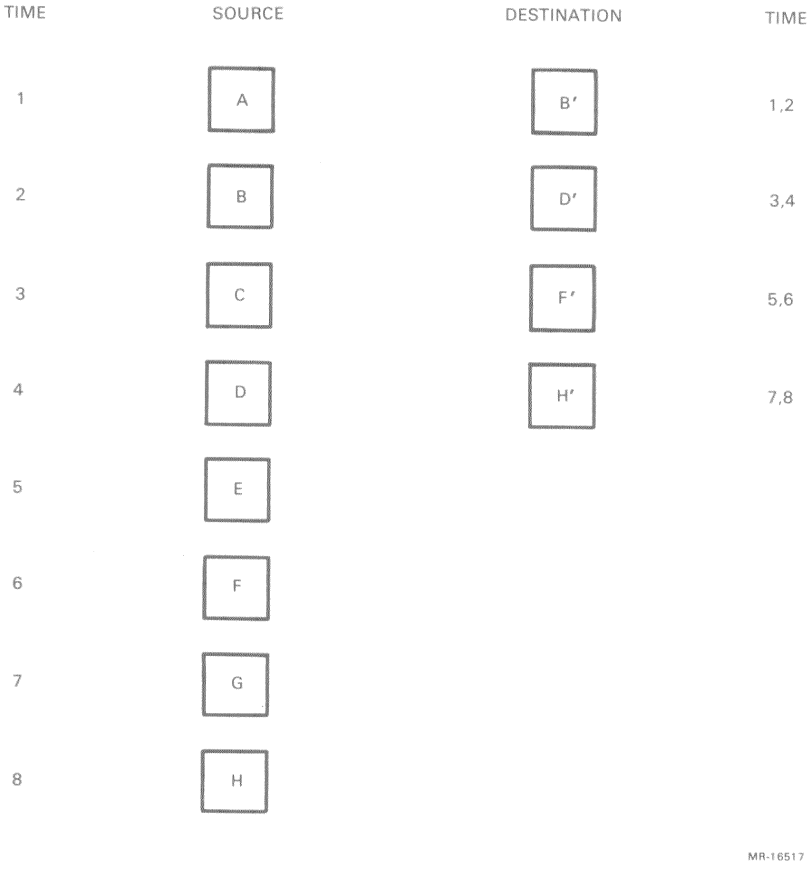


Figure 4-24 Source Scaled Down by a Factor of Two

A side-effect of down-scaling is that "complement" writing will not properly erase an image, since a point in the destination is addressed more than once. This problem can be solved by manipulating the image in a staging area, then using a rectangular rasterop to update the screen refresh area. As the previous figures illustrated, regardless of the scale factor, the extent of a rasterop is governed by the dimensions of the destination raster. That is, the rasterop process completes when one pass over the destination raster is made.

Linear Pattern Generation

This section discusses how the linear pattern mode in the address processor chip can be used to generate a pattern aligned with the destination raster boundaries. The source 1 raster is a rectangle whose extent is usually proportional to the size of the destination fast and slow vectors. In this case, a rasterop simply transforms the source into a scaled and rotated copy in the destination. In linear pattern mode, the source 1 raster contains a pattern that is scaled and replicated within the destination raster limits. The parameters of the pattern are the coordinates of the source 1 origin and the source 1 extent in the X and Y directions. The process for generating a pattern is shown in Figure 4-25.

Basically, linear pattern generation is produced by a simple modification of the source 1 raster algorithm, which resets a source edge vector when the appropriate pattern limit is reached. All other aspects of source 1 operand address computation are unchanged. Therefore, the pattern can be scaled by applying the scale factors.

The pattern repeat cycle is determined by the extent of the source DX and DY vectors and the scale factor. In Figure 4-25, a scale factor of 1 is used.

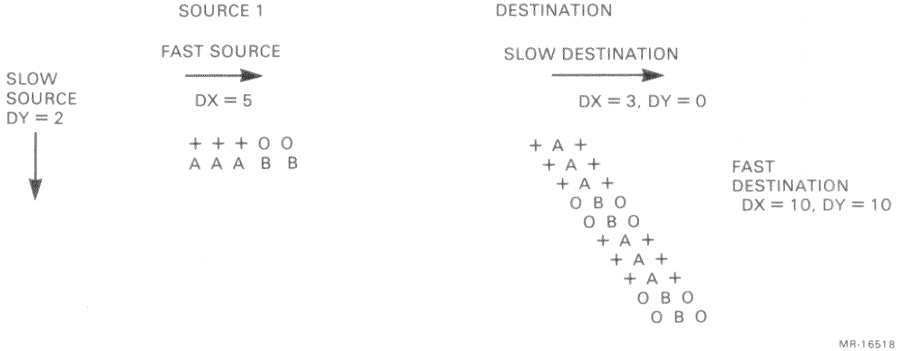


Figure 4-25 Linear Pattern

During a fast scan, source addresses are incremented in the X direction at a rate determined by the scale factor. When the DX limit is reached, the source address is reset to the beginning, and the source fast vector is scanned again to create the pattern effect along the destination fast vector.

Similarly, during a slow scan step, the slow source vector is incremented until the DY limit is reached. At that time, the slow vector is reset and the pattern is scanned again from the origin.

Second Source Operand

The second source operand can be used in one of the following ways.

- To generate a tile pattern from a rectangular raster.
- To copy pixels from a source raster having the same shape as the destination.

When used to generate tile patterns, this operand is defined by the following parameters.

- The origin of the pattern in memory coordinates
- The length and width of the pattern

When used to copy a raster, the source origin is specified by an offset relative to the unindexed coordinates of the destination. The fast and slow source vectors have the same length and orientation as the destination vectors, and there is no scaling between source and destination.

Figure 4-26 shows the mapping of source 2 to destination pixels for a destination raster having a width of 1.

During each rasterop address cycle, the hardware computes the source 2 operand address by applying the offset to the unindexed destination address.

Figure 4-26 Mapping of Source 2 to Destination

Tiling
Unlike a linear pattern, a tile is always aligned with the X- and Y-axis. The tiling logic generates a textured surface that has an origin coinciding with the coordinate system of the screen. In this way, the tiling of overlapping images is always in registration. Tiling an image is equivalent to uncovering the surface, revealing the tiled pattern underneath.

The tile dimensions are independent of each other and are restricted to powers of two, ranging from 2^2 to 2^9. The tile width may never be less than the memory bus width, but smaller tile elements can be defined by repeating the tile elements within the cell. For proper alignment of adjoining tiles, the dimensions of the embedded pattern must be multiples of two.

The key to tile pattern generation is the use of modulo arithmetic in the tile address computation. The tile pattern is generated as follows.

1. The destination address is computed using the raster scan algorithm.

2. The destination address is reduced to a modulus of the tile size.

3. The result of step 2 is added to the source 2 origin.

Note that tiling is disabled by simply omitting step 2.

4.2.5.3 **Model of Raster Operations** -- The model shown in Figure 4-27 summarizes the rasterop capabilities described in previous sections.

The operands in Figure 4-27 are defined as follows.

Operand A : Operand B
Source 1 raster or linear pattern
Source 2 raster or tile
Destination raster
Constant
I/D interconnect data

Operand B, Operand C : Source 1 raster or linear pattern
Source 2 raster or tile
Destination raster
Constant
I/D interconnect data

The functional units consist of:

- The map function, which includes the logic register and foreground/background selection

- Mask logic functions that can form the mask from the operand or its complement.

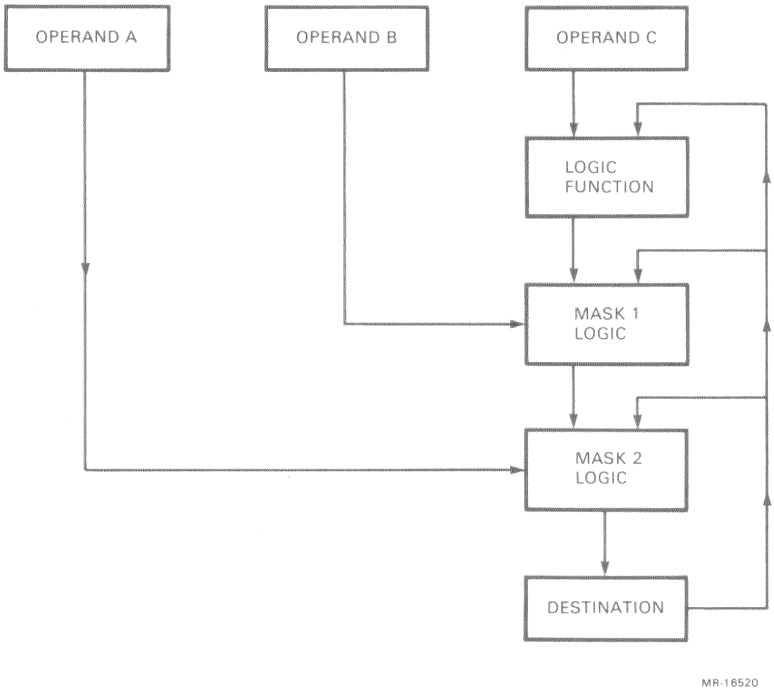


Figure 4-27 Simplified Rasterop Model

4.2.6 Polygon Fill

The polygon fill feature uses the rasterop logic to generate a sequence of addresses that cover the space between two edge vectors. The enclosed space can be filled with a solid color, tile pattern, or image. A fill operation is defined by the two edge vectors and a fill vector that is parallel to the X- or Y-axis. The first edge vector, called the A vector, is defined by the parameters normally used to specify the origin and slope of the slow destination vector. The second edge vector, called the B vector, is specified by the source 1 origin, DX and DY parameters. The fill vectors connect points on the A vector to corresponding B vector points as shown in Figure 4-28. To permit joining of fill areas, the fill stops one point shy of the endpoint.

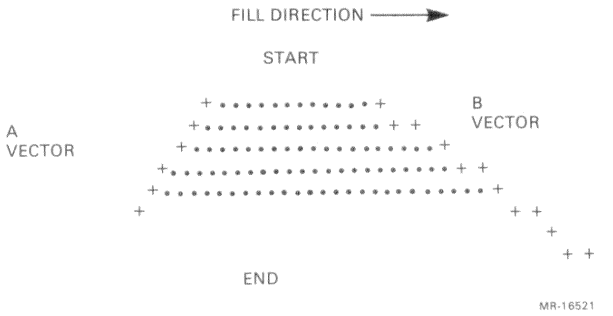


Figure 4-28 Polygon Fill

At completion, the address processor chip retains the final source and destination addresses. These can be used as the origins for a subsequent fill operation. In this case, only the new slopes would have to be specified.

The fill operation is defined by the following parameters.

- The fill direction, which is parallel to either the X- or Y-axis. When there is a choice, the X-axis should be selected for performance reasons.
- The A edge vector, specified using the destination slow vector origin, DX and DY.
- The B edge vector, specified using the source 1 origin, DX and DY. The source origin need not be coincident with the destination. However, it must be possible to connect the source and destination origins with the first fill vector. Also, the edge vectors may intersect. When this occurs, filling continues, but the direction of the fill vector is reversed.

The destination pixels are within the trapezoid enclosed by the A and B vectors. The source pixels are determined by the second source operand specified.

A special case of the fill operation allows a polygon to be filled from a baseline that is parallel to the X- or Y- axis. In this instance, only the X or Y component of the source 1 vector must be specified, depending on the fill baseline.

The fill operation is a variant of the basic raster operation and shares most of the rasterop logic. Therefore, the following parameters should be set as follows.

- Address Processor Chip

Source 1 operand:	Disabled.
Scale factors:	Set to 1.
Source 2 operand:	As required, can be disabled, set for tiling or image modes.
Clipping mask:	As required.

- Video Processor Chip

Destination control store RAM:	Set up to process the pixels addressed by the address processor chip parameters.
Source 2 control store RAM:	Required if the source 2 operand is used for tiling or image generation.

4.2.6.1 Polygon Fill Model

The model shown in Figure 4-29 summarizes the polygon fill functions described above. The operands in Figure 4-29 are defined as follows.

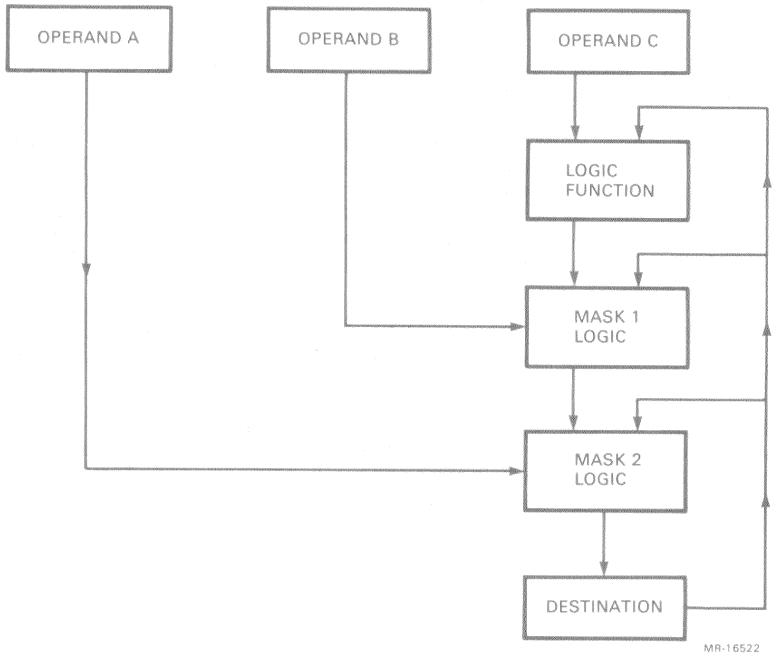


Figure 4-29 Polygon Fill Model

Operand A: Operand B
Source 2 raster or tile
Destination trapezoid
Constant
I/D interconnect data

Operand B, Operand C: Source 2 raster or tile
Destination trapezoid
Constant
I/D interconnect data

4.2.6.2 **Side Effects of the Fill Algorithm** -- When filling a polygon, the program must account for the following side effects.

1. Filling does not include the last point on the edge vector. Therefore, the firmware must account for any unfilled vertices that result.

2. When regions are abutted or the edge vectors intersect, interior points may be addressed twice with the usual adverse effect on complement writing.

To understand the problem with adjoining areas, a closer look at the algorithm is required. In minimizing the occurrence of doubling, the fill algorithm attempts to draw only one fill vector between edge vectors at a given X or Y coordinate. Figure 4-30 shows successive points along the unfilled edge vectors. Because of the finite resolution, any vector-approximation algorithm will occasionally generate two points in succession that have the same X or Y coordinate. To ensure that only one fill vector is drawn between edges, the address processor chip connects only the outermost points along the fill direction.

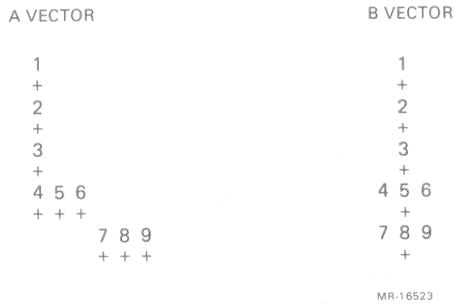


Figure 4-30 Edge Vectors Before Fill

Figure 4-31 replicates Figure 4-30, but with the fill vectors drawn in. The X marks the last point on each edge vector.

If the fill is continued from the end points shown in Figure 4-31, the next fill vector will be coincident with the last vector drawn. This is shown in Figure 4-32. The "doubled" points are marked with an asterisk.

Since only the outermost points are connected, doubling will not occur when the edge vectors are outbound. This is shown in the Figures 4-33 and 4-34, which illustrate successive fill operations with diverging edge vectors.

In Figure 4-33, point 6 is the A vector end point. The fill operation terminates one point before this, at point 5. Since this is not the outermost point along the fill axis, no fill vector is drawn. Continuing the fill as shown in Figure 4-34 does not result in doubling.

In general, joining two fill areas results in doubling any time at least one of the previous edge vectors approaches the opposite vector with a slope of less than 1:2 (with the longer component in the fill direction). One way to compensate for doubling in complement mode is to write the doubled pixels an "odd" number of times. A partially effective algorithm involves preceding the next area fill with a fill command, using a unit vector inbound, followed by a unit vector outbound. The inbound vector draws a single fill. The outbound vector repositions the edge vector without drawing a fill vector. The first fill vector drawn by the area fill command will address the doubled points for the third time. This approach fails when the edge vector has a slope that results in three or more points along the same fill vector at the endpoint. It is recommended that you avoid complement mode writing for polygon fill.

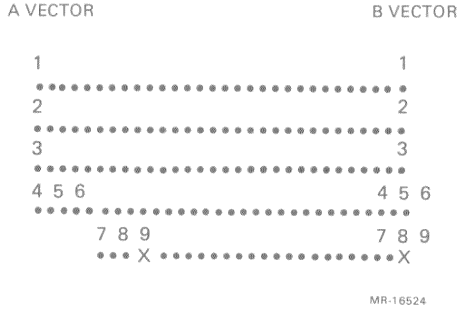


Figure 4-31 Filled Area

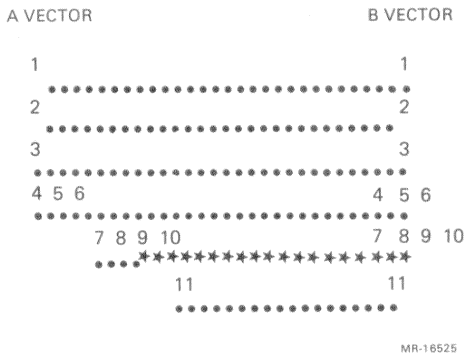


Figure 4-32 Filled Area Continuation Showing Doubling

Figure 4-33 Diverging Edge Vectors

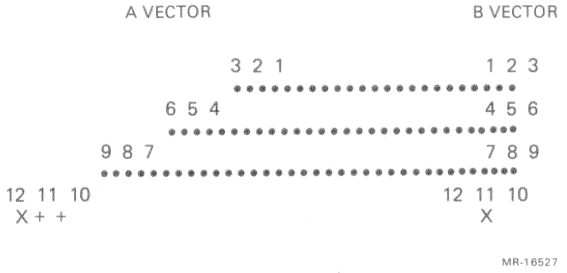


Figure 4-34 Filled Area with Diverging Edges

4.2.7 Processor/Bitmap Transfers

Processor/bitmap transfers allow the MicroVAX CPU to copy the contents of a rectangular raster, aligned with the X- and Y-axis, to or from system memory. These operations are similar to rasterops, except that the source or destination is the Q22-Bus. The following kinds of operations can be performed.

- Transferring the portion of a raster that resides in a single plane of memory.
- Transferring the value of each pixel in a raster.

Single-plane transfers provide a bit-efficient way to save and restore an image. Pixel transfers are efficient when an image is to be processed in the MicroVAX CPU, as would be the case for a flood operation. Like rasterops, the actual logical operations are determined by a control register in the video processor chip. Also, data written into the bitmap will be masked by the clipping rectangle.

4.2.7.1 Single-Plane Bitmap-to-Processor Transfers -- A single-plane bitmap-to-processor transfer moves data from a raster in the bitmap to the MicroVAX CPU by way of the address processor chip. This operation is defined by the following parameters.

- Origin -- Source 1 operand coordinates.
- Fast vector -- Destination fast vector aligned with the X-axis.
- Slow vector -- Destination slow vector aligned with the Y-axis.
- Destination origin -- Determines the alignment of data placed on the I/D interconnect for transmission to the MicroVAX CPU.
- Video processor chip control store RAM location to be used (can be CSR 1 or 2 in either bank).

Alignment of the fast and slow vector is required to allow the MicroVAX CPU to easily compute the space required for raster storage.

Although the destination parameters are specified, no data is actually written into the bitmap. Instead, the destination origin is used simply to establish the correspondence between bits in memory and data on the I/D interconnect.

For the video processor chip transmitting the data, the control store RAM location specified in the address processor chip must have the I/D interconnect output enabled. The corresponding CSR in all other video processor chips must have the I/D interconnect output disabled. Otherwise, the other video processor chips can be deselected by means of the chip select register.

The series of raster addresses are generated by traversing the fast and slow destination vectors. The source address is computed by adding the offsets from the destination raster to the source 1 origin. Each time the fast vector is exhausted, a step along the slow vector is taken.

As mentioned earlier, the mapping between data in the bitmap and the contents of the I/D interconnect is determined by the origin supplied for the destination raster. To see how this works, realize that the smallest unit of transfer between the bitmap and video processor chip is a single 16-bit word whose bitmap address is a multiple of 16. These limits are termed "natural memory boundaries." For a normal rasterop, the video processor chip must align the source operand so that the position of a bit relative to the raster boundaries is preserved when the data is copied to the destination. This almost always results in a difference between the location of the bit relative to the natural boundaries of the source and destination.

The effect is shown in Figure 4-35. The vertical lines denote word boundaries in the bitmap.

As indicated in Figure 4-35, the alignment of raster contents can be preserved by shifting the data relative to the natural word boundaries. The shift amount is equal to the difference between the X coordinate of the source and the X coordinate of the destination, converted to a modulus of 16. The resulting I/D interconnect data word is a copy of the source, after it has been aligned with the corresponding word in the destination raster. Because the destination is not written, the destination address can be used to simply align the raster within the I/D interconnect data word. In some cases, it is desirable to align the Y edge of the raster with bit 0; therefore, a destination origin of 0,0 is used, and destination indexing is disabled.

To further show the effect of shifting, Figure 4-36 illustrates how a source raster is packed into system memory for various destination raster X addresses. Note that increasing pixel addresses in X correspond to increasing bit positions. Each character represents the state of one bit in refresh memory.

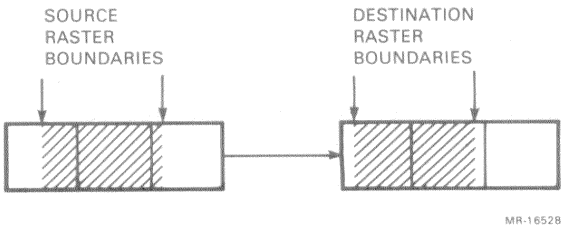


Figure 4-35 Source-to-Destination Alignment

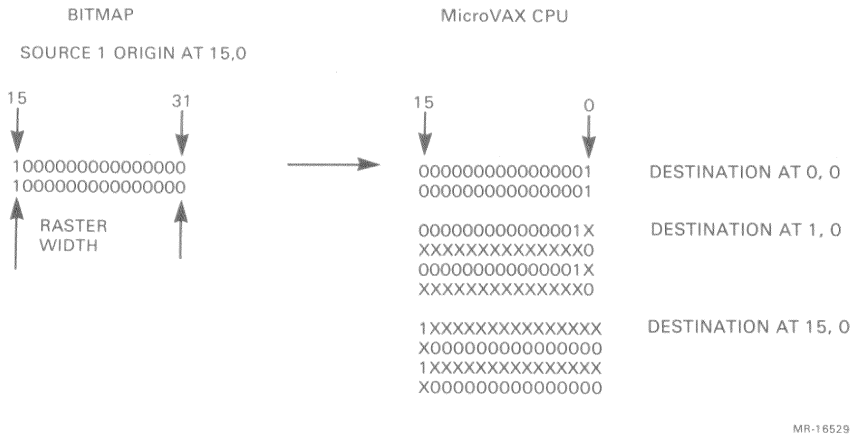


Figure 4-36 Bitmap-to-Processor Mapping (Simple Case)

The X symbols in Figure 4-36 represent bits read in from areas to the left or right of the source due to the shifting needed to align the source raster with the destination. Observe how the destination alignment effects the number of MicroVAX memory locations required to store the data. When the left edge of the source raster is aligned with bit 0 of the destination, 16-bits of source data are contained in a single 16-bit word in the destination. When the left edge is not aligned, the source data spans two 16-bit words in local memory.

Figure 4-37 shows a more general case, where the source raster is larger than the memory bus width and local memory word size.

From the example shown in Figure 4-37, it is clear that the space to store a raster in local memory depends partly on the alignment of the destination raster origin. The space computation is performed by:

1. Determining the number of natural words in the bitmap that would be spanned by the destination fast vector

2. Multiplying this value by the height of the destination raster.

Based on the above, the following procedure can be derived.

```
!
! Compute the number of natural words spanned by the
! fast vector. Only integer division is used.
!
! Truncate the starting address to a word boundary.
!
```

```
X1 := (DESTINATION_X/16) * 16

!
! Compute the endpoint in the X-axis and round up to
! the nearest word boundary.
!
! Compute the endpoint coordinate.
!

X2 := DESTINATION_X + DESTINATION_X_LENGTH

!
! Round up to a multiple of the word size.
!

X2 := ((X2 + 16)/16 ) * 16

!
! Compute the number of words spanned by the X vector.
!

X_WORDS = (ABS (X2 - X1))/16

!
! Compute the total number of words required.
!

N_WORDS = X_WORDS * ABS (DESTINATION_Y_LENGTH)
```

Of course, all of the above can be simplified by suitable optimization.

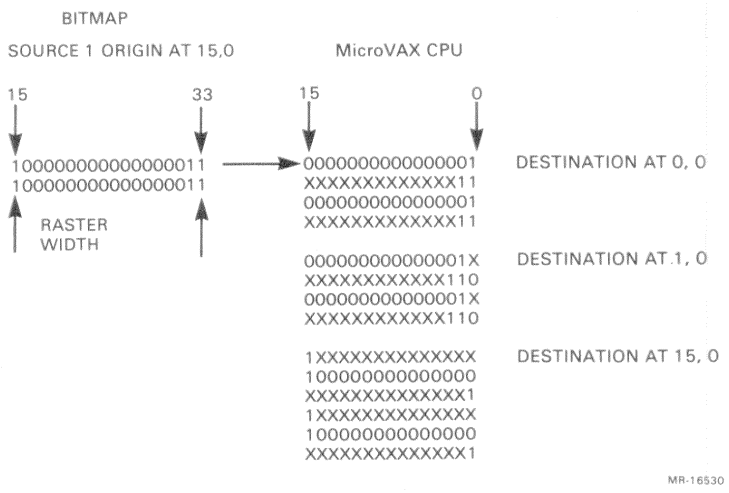


Figure 4-37 Bitmap-to-Processor Mapping (General Case)

4.2.7.2 Single-Plane Processor-to-Bitmap Transfers -- The following parameters are required for a single-plane processor-to-bitmap transfer.

- Origin and extent of the destination raster defining the pixels to be transferred. The fast vector must be aligned with the X-axis. The slow vector is aligned with the Y-axis.
- The clipping rectangle.
- The logic function register to be used.
- The logic function programmed appropriately.
- A video processor chip control store RAM location set up as follows.

Internal cycle destination field	As required by the function.
External cycle destination field	Set up to receive data from the I/D interconnect and process it as required by the function.
I/D interconnect output	Disabled.

All other video processor chips can be excluded from the transfer by either programming the CSR or update chip select register accordingly.

Data supplied from the MicroVAX CPU is placed on the I/D interconnect, and from there is written into the bitmap without rotation. The sequence of bitmap locations is determined by the destination raster scanning algorithm.

The mapping from host memory or I/D interconnect to bitmap memory is shown in Figure 4-38.

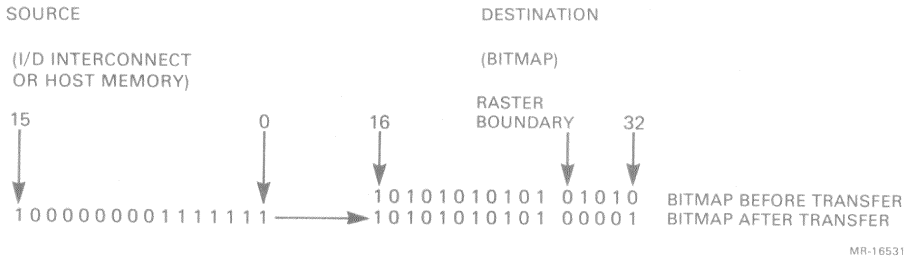


Figure 4-38 Processor-to-Bitmap Mapping

The portion of the data word that is outside the raster boundaries is masked out. Since the output is not rotated, it is the responsibility of the local processor to ensure that the data is aligned properly before being written to the bitmap.

To visualize the problem, consider the example shown in Figure 4-39. The short vertical lines mark addresses in the bitmap that are multiples of the memory bus width.

In Step 1 of Figure 4-39, data is transferred from the bitmap that results in each 16-bit bitmap segment being shifted into alignment with bit 0 in local memory. In Step 2 the contents of local memory is later written back into the bitmap at the same location from which it was read, but without the shifting required for proper alignment.

Since each memory word is written into the bitmap, so that it is aligned with the natural boundaries, the image is shifted left with respect to its original position. The part outside the raster limits is actually masked, and the part of the image adjacent to the right boundary is replaced with the random data that was in the high-order bits of local memory.

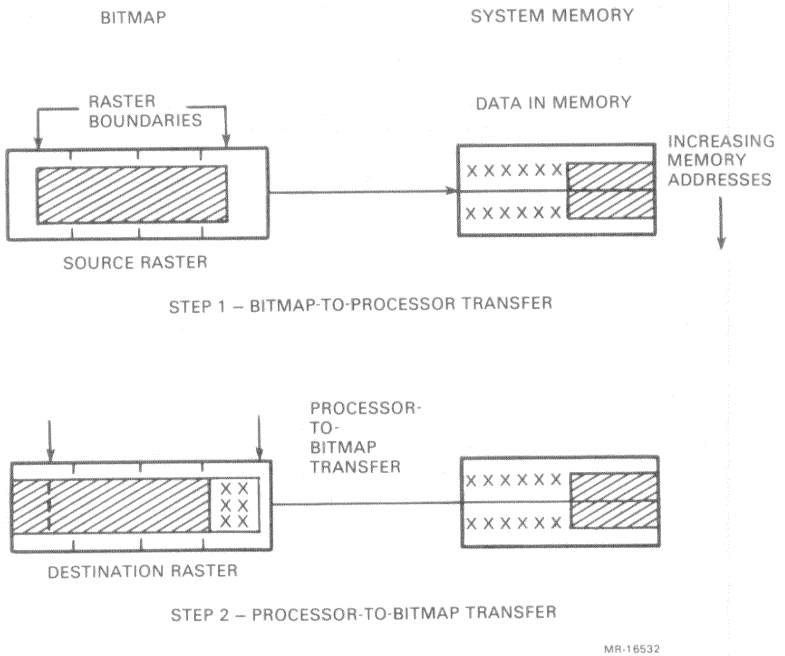


Figure 4-39 Alignment Error Example

The corrective alignment of output data can be accomplished by:

1. Aligning the data on input from the bitmap based on knowledge of where the output raster will be located

2. Emulating the video processor chip barrel shifter function on output

3. Forcing a raster to be aligned on natural bus-width boundaries.

The first method is suitable for saving and restoring an image in place of using a DMA device. In this case, the number of words to be transferred is computed as shown in the previous section. On the bitmap-to-processor transfer, the video processor chip automatically performs the required alignment. The resulting raster is then transferred to local memory, and the following parameters are saved.

- The parameters of the original raster, including the origin and dimensions

- The number of words transferred

To restore the image, the output raster is defined using the parameters saved earlier, and the DMA device is requested to transfer the specified number of words. The output is automatically masked to the raster boundaries.

In the second case, the goal is to emulate the video processor chip barrel shifter by shifting the data based on the difference between the X coordinate of the input and output rasters (modulo 16). During the bitmap-to-processor transfer, the data may be aligned in any convenient manner by specifying the destination raster address appropriately. On output to the bitmap, the shift is performed by the MicroVAX CPU, concatenating adjacent words as required. The net shift amount, between input and output, must equal the difference computed as described above. The disadvantage of this method is that, on output, it is not amiable to processing by a DMA engine.

The third method is also useful when the ultimate destination is not known. This procedure lends itself to a DMA processor, but incurs added overhead for an additional rasterop. In this case, the input raster parameters for the bitmap-to-processor transfer are modified by truncating the origin and rounding up the X dimension so that both are a multiple of 16. The raster is stored in system memory using the modified dimensions.

Restoring the image is a 2-step process. First, the image is transferred to a staging area in off-screen memory using the adjusted dimensions. It is then moved to the ultimate destination using the original raster dimensions. Data outside the raster limits is masked out (that is, the external area is not modified).

4.2.7.3 **Z-Axis Processor/Bitmap Transfers** -- Z-axis transfers move the contents of a raster, aligned with the X- and Y-axis, to or from local memory one pixel at a time. Each data word represents up to 8 bits of pixel value.

Since each transfer involves only one pixel, there are no requirements for shifting or alignment. Data is transferred by scanning the fast vector, which sweeps out a path defined by the slow vector. Like a rasterop, the scanning algorithm does not trace out the last point. Therefore, the raster parameters must be adjusted accordingly.

Z-Axis Bitmap-to-Processor Transfers
Z-axis transfers from the bitmap to the processor move the contents of a source raster that is aligned with the X- and Y-axis. The operation is defined by the following parameters:

- The raster size -- Defined by the magnitude of the fast and slow destination vectors. The fast vector must be parallel to the X-axis, and the slow vector parallel to the Y-axis.

- The raster origin -- Defined by the coordinates of the source 1 operand.

The number of words transferred is equal to the total number of pixels enclosed by the raster.

Z-Axis Processor-to-Bitmap Transfers
Z-axis transfers from the processor to the bitmap fill a destination raster with pixel values contained in local memory. The following parameter defines the transfer.

- A destination raster defined by the origin and extent of the destination operand

As for the bitmap-to-processor transfer, the number of pixels is defined by the total number of pixels enclosed by the raster.

4.3 ADDRESS PROCESSOR CHIP DESCRIPTION

The interface from the MicroVAX CPU to the video processor chips and bitmap is through a set of address processor chip registers that:

1. Set system and CRT timing
2. Hold rasterop and scrolling parameters
3. Control address processor chip operation.

Unless otherwise specified, all registers are write-only, and numerical registers contain a 14-bit, 2's complement value (including sign). The upper 2 bits are unused and must be set to zero.

In some cases, loading a register causes a specific system action to be taken. This is noted in the register's description.

4.3.1 Coordinate Systems

In the VCB02 video subsystem, images are referenced through two Cartesian coordinate systems:

1. Memory coordinates, referenced to the origin of the physical bitmap memory space

2. Screen coordinates, referenced to the upper left corner of the CRT.

In either case, X increases to the right and Y increases downward.

4.3.2 Address Processor Chip Commands

All address processor chip commands are written into one of the following registers.

- The COMMAND register, for update commands.

- The ID__SCROLL__COMMAND register, to load an I/D interconnect command while a rasterop is in progress. The contents of the high byte are ignored.

The scroll command port is provided for the video processor chip registers to perform loading of various scrolling functions without interfering with rasterops in progress. The sole function of this port is to issue I/D interconnect commands. Therefore, the contents of the high byte are ignored.

4.3.2.1 Rasterop Command --

Implicit Inputs:

The following video processor chip registers must be set up.

- For each enabled operand:

  The appropriate CSR in the selected bank. Within each bank, the address processor chip makes the following assignment of CSR's to operands.

  Bank 0 or 1, CSR 1 = Source 1
  Bank 0 or 1, CSR 2 = Source 2
  Bank 0 or 1, CSR 3 = Destination

- The logic function register specified in the command word.

The following address processor chip registers must be set up.

- Destination operand parameters:

DESTINATION_X	Destination X origin
DESTINATION_Y	Destination Y origin
FAST_DEST_DX	Destination fast DX
FAST_DEST_DY	Destination fast DY
SLOW_DEST_DX	Destination slow DX
SLOW_DEST_DY	Destination slow DY

- If the source 1 operand is enabled:

FAST_SOURCE_1_DX	Fast source 1 DX increment
SLOW_SOURCE_1_DY	Slow Source 1 DY increment
SOURCE_1_X	Source 1 X origin
SOURCE_1_Y	Source 1 Y origin
FAST_SCALE	Fast vector scale factor
SLOW_SCALE	Slow vector scale factor

- If the source 2 operand is enabled:

SOURCE_2_X	Source 2 X origin
SOURCE_2_Y	Source 2 Y origin
SOURCE_2_SIZE	Source 2 height, width and tiling enable flag

- MODE set as follows.

<1:0> Set to one of the following values.

00 = Normal mode. Source area is determined from the scaled destination area.
10 = Linear pattern mode.

<4> Hole fill enable, as required.

<5> First source indexing enable, as required. Will usually be set, if the first source is within a region.

<6> Destination indexing enable, as required. Will usually be set, if the destination is within a region.

<7> Pen down, as required. Set based on whether or not the destination raster is to be written.

Rasterop Protocol
The following protocol is used to initiate a rasterop.

```
ROUTINE NEW_RASTEROP (Video processor chip parameters,
                      address processor chip parameters)

! This entry point is called when starting a new rasterop.
! That is, when the last function issued through the command
! register is different from the one to be issued.

! Wait until it is safe to load all registers.
WAIT UNTIL STATUS <address output complete> IS SET
Load Video Processor chip parameters

Load Address Processor chip parameters

Load Rasterop Command

RETURN
```

The following protocol is used when continuing a rasterop. Continuation is defined as a repetition of the last rasterop command, modifying only the source 1 and destination operands.

```
ROUTINE CONTINUE_A_RASTEROP (COMMAND, SOURCE_1_X,SOURCE_1_Y,
                             DESTINATION_X,DESTINATION_Y)

! This routine is called when a rasterop command is to be
! continued with the only change being the origin of the first
! source and destination operands. This mode is useful when
! writing text into the bitmap.

! Wait until it is safe to reload the source 1 and destination
! origins.
WAIT UNTIL STATUS <Initialization complete> IS SET

LOAD Address Processor chip with SOURCE_1_X, SOURCE_1_Y,
DESTINATION_X, DESTINATION_Y

! COMMAND must be identical to the last command.
LOAD Address Processor chip with COMMAND

RETURN
```

The following protocol is used when any address processor chip rasterop parameter is to be modified. To improve throughput, it is executed while addresses from the previous rasterop are still in the pipeline. The following are the parameters that may be modified.

```
    FAST_SOURCE_1_DX
    FAST_SOURCE_1_DY
    SOURCE_1_X
    SOURCE_1_Y
    DESTINATION_X
    DESTINATION_Y
    FAST_DEST_DX
    FAST_DEST_DY
    SLOW_DEST_DX
    SLOW_DEST_DY
    FAST_SCALE
    SLOW_SCALE
    SOURCE_2_X
    SOURCE_2_Y
    SOURCE_2_SIZE
    RASTEROP_MODE

ROUTINE MODIFY_A_RASTEROP (COMMAND, Address Processor chip
parameters)

! This command allows the next rasterop to be started as soon as
! the address processor chip has completed address generation from
! the previous rasterop, but before all the previous addresses
! have been used by the video processor chip.
! The command word must be identical with the last command.

! Wait for rasterop address computation to finish.
WAIT UNTIL STATUS <rasterop complete> IS SET

LOAD address processor chip parameters

LOAD command register with next rasterop function (COMMAND)

RETURN
```

4.3.2.2 Processor/Bitmap Transfers -- The following commands transfer the contents of a single plane or a series of pixel values between the MicroVAX CPU and a raster in the bitmap.

Single-Plane Processor-to-Bitmap Command

Implicit Inputs:

The planes not receiving data are disabled, either through the chip select hardware or by programming the CSR in each video processor chip appropriately.

The following video processor chip registers must be set up.

- Control store RAM -- Set up to receive data from the I/D interconnect during the external cycle. Other fields are set up as required.

- Logic function control register

The following address processor chip parameters must be set up.

- MODE set as follows.

```
<1:0> Rasterop mode          = 00 (normal mode)
<4>   Hole fill enable       = 0
<6>   Destination indexing   As required.
<7>   Pen down               As required.

DESTINATION_X                X origin of output raster
DESTINATION_Y                Y origin of output raster
FAST_DEST_DX                 Length of output fast vector
FAST_DEST_DY                 Y component of fast vector = 0
SLOW_DEST_DX                 X component of slow vector = 0
SLOW_DEST_DY                 Length of output slow vector
FAST_SCALE = Unity
SLOW_SCALE = Unity
```

Single-Plane Bitmap-to-Processor Command

Implicit Inputs:

The planes not transmitting data are disabled, either through the chip select hardware or by programming the CSR in each video processor chip appropriately.

The following video processor chip register must be setup:

Control Store RAM - Set up to transmit data to the I/D interconnect during the external cycle. Other fields are ignored.

The following address processor chip parameters must be set up.

- MODE set as follows.

```
<1:0> Rasterop mode          = 00 (normal mode)
<4>   Hole fill enable       = 0
<5>   Source 1 indexing      As required.
<6>   Destination indexing   As required.

SOURCE_1_X          X origin of input raster
SOURCE_1_Y          Y origin of input raster
FAST_DEST_DX        Length of input fast vector
FAST_DEST_DY        Y component of fast vector = 0
SLOW_DEST_DX        X component of slow vector = 0
SLOW_DEST_DY        Length of output slow vector
SOURCE_1_DX         Must have the same sign as the destination
                    fast vector
SOURCE_1_DY         Must have the same sign as the destination
                    slow vector
FAST_SCALE = Unity
SLOW_SCALE = Unity
```

Z-Axis Processor-to-Bitmap Command

Implicit Inputs:

The following video processor chip register must be set up.

- Logic function control register 10 must be set up to copy the pixel value into the bitmap.

The following address processor chip parameters must be set up.

- MODE set as follows.

```
<1:0> Rasterop mode          = 00 (normal mode)
<4>   Hole fill enable       = 0
<6>   Destination indexing   As required.
<7>   Pen down               As required.

DESTINATION_X                X origin of output raster
DESTINATION_Y                Y origin of output raster
FAST_DEST_DX                 Length of output fast vector
FAST_DEST_DY                 Y component of fast vector = 0
SLOW_DEST_DX                 X component of slow vector = 0
SLOW_DEST_DY                 Length of output slow vector
FAST_SCALE = Unity
SLOW_SCALE = Unity
```

Z-Axis Bitmap-to-Processor Command

Implicit Inputs:

The following address processor chip parameters must be set up.

- MODE set as follows.

```
<1:0> Rasterop mode          = 00 (normal mode)
<4>   Hole fill enable       = 0
<5>   Source 1 indexing      As required.
<7>   Pen down               As required.

SOURCE_1_X                   X origin of input raster
SOURCE_1_Y                   Y origin of input raster
FAST_DEST_DX                 Length of input fast vector
FAST_DEST_DY                 Y component of fast vector = 0
SLOW_DEST_DX                 X component of slow vector = 0
SLOW_DEST_DY                 Length of input slow vector
FAST_SCALE = Unity
SLOW_SCALE = Unity
SOURCE_1_DX                  Must have the same sign as the input
                             fast vector
SOURCE_1_DY                  Must have the same sign as the input
                             slow vector
```

4.4 VIDEO PROCESSOR CHIP DESCRIPTION

4.4.1 Z-Axis Addressing Mode

The Z-axis addressing mode reduces and, in most cases, eliminates the adverse effect on performance caused by the addition of more bitmap memory planes. Certain graphics operations executed by the MicroVAX CPU, such as flooding or painting an area, require that the value of a pixel be processed in the MicroVAX system's memory. Without hardware assistance, assembling all the components of a pixel requires intervention by the MicroVAX CPU to:

1. Read the data in every plane or subplane
2. Perform the required shifting and masking functions to extract each component of the pixel
3. Combine the extracted pixel component with the result.

Once the operation is complete, a reverse procedure is needed to set the state of the bitmap. Obviously, therefore, system performance degrades as the number of memory planes increases.

With Z-axis addressing, on the other hand, logic within each video processor chip can be used to assemble or disassemble up to 16-bits of pixel value in a single transaction. A further performance improvement can be achieved by using the DMA support to perform the actual processor-to-bitmap transfer.

The video processor chip's Z-axis logic:

- Inserts bitmap data at the correct position in the word for a bitmap-to-processor transfer
- Extracts data from the correct position in the word for a processor-to-bitmap transfer.

Figure 4-40 shows the relationship of address to position for a 4-plane system.

Each video processor chip is numbered with its Z-axis address. As Figure 4-40 shows, bits 4 through 15 of the data word are unused.

NOTE

On a Z-axis bitmap to processor transfer, the value of unused bits are undefined.

The maximum value of a pixel is determined by the number of physical memory planes.

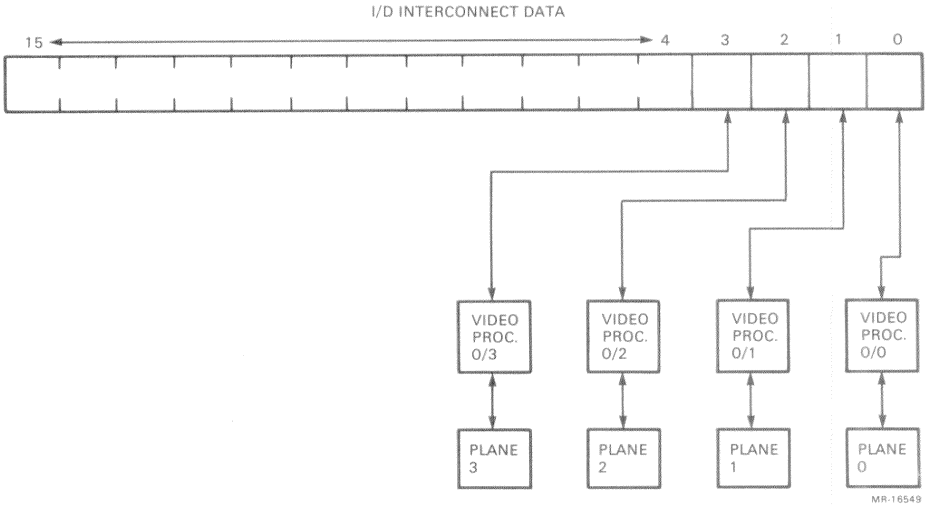


Figure 4-40 Z-Axis Addressing

Here are some things to remember about Z-axis processing:

1. A video processor chip can have any Z-axis address, if it does not overlap the address assigned to another video processor chip.

2. The states of unused bits are undefined. These bits must be set to some known state before referencing the pixel value.

4.4.2 Z-Axis Register Loads

The video processor chip foreground, background, source, and scroll fill registers are set to specific colors by the MicroVAX CPU. To facilitate this, the Z-axis register load function is provided to perform similarly to the Z-axis processor-to-bitmap transfers. Namely, in one I/D interconnect transaction, each video processor chip in a block extracts data from that portion of the I/D interconnect specified by the bit address. The bits of the selected register will either be set to all 1s or all 0s, corresponding to a 1 or 0 in its bit position in the data word.

CHAPTER 5
DIAGNOSTICS

5.1 OVERVIEW

This chapter contains information on diagnostic and console I/O firmware that will aid the user in isolating VCB02 module faults at the field-replaceable unit (FRU) level. The Q22-Bus VCB02 video subsystem is a multiplane bitmapped graphics controller that provides enhanced graphics processing capabilities to VAXstation II/GPX workstations. The hardware allows the user to install VCB02 modules of 4 or 8-planes where one specific address has been reserved as a secondary alternate console device.

The diagnostic VCB02 ROMs total 16 kilobytes and reside on the VCB02 base module. Power-up self-test and console I/O support are provided by the diagnostics. The power-up self-test is a concise test, invoked by the KA630 ROM, that quickly tests the VCB02 video subsystem. The power-up self-test verifies the VCB02 chip set, bitmap memory, video signals, interrupts, DMA, DUART, keyboard, and mouse or tablet.

The console I/O support provides sufficient functions to support the input and output for the console program residing in the KA630 ROM. Some of the functions include initialization, get character, put character poll, and put character.

5.2 OPERATION

When the KA630 CPU module is powered on, it begins executing the console program that resides in the KA630 ROM. Determining that it was invoked on power-up, the KA630 ROM performs the following steps before any video display device is referenced.

1. TOY chip verification. If the battery is too low or the battery backed-up RAM contains an invalid checksum, the clock is stopped and the time is zeroed. The console language and keyboard information is cleared, causing a language query later for the operator.

2. Each page of memory is checked until five good pages are found. These pages will be used for the console page, console stack, system memory bitmap, and diagnostic scratch space.

3. A diagnostic is run to verify that the Q22-Bus is arbitrating.

The next step is to determine what type of video display device is present. The types of video display devices include the primary alternate console display device (VCB01 - MicroVAX I/II) or the secondary alternate console device, such as the VCB02 (MicroVAX II/GPX workstations) or none at all. The console program reads address 20001E92H followed by address 20001F00H. If either address responds correctly, the console program loads the base address of the U.S. font table, the base address of the multinational font table, the address of the keycode translation routine, and the keyboard flag into the proper entries of the console page.

If address 20001E92H responded correctly, control passes to the VCB01 power-up self-test, which resides in the KA630 ROM. The test checks the VCB01 and updates the missing entries of the console page.

If address 20001E92H responded incorrectly and address 20001F00H responded correctly, the console program maps the VCB02 ROM, checksums the VCB02 ROM, finds and checksums the VCB02 power-up self-test file, and passes control to the VCB02 power-up self-test. The test checks the VCB02 and updates the missing entries of the console page. No other VCB02 in the system is affected.

After the console device has been selected and tested, the KA630 ROM uses that device for console input and output. Other software applications have the ability to use the console I/O functions indirectly through the KA630 ROM via specified KA630 ROM addresses or the console page. Specific applications include power-up, bug checks, and halt situations.

5.3 IMPLEMENTATION

5.3.1 Performance

The power-up self-test provides 90 percent fault detection/isolation to the FRU level. The VCB02 video subsystem's FRUs include the following.

- VCB02 base module
- First VCB02 4-plane module
- Second VCB02 4-plane module
- LK201 keyboard
- Three-button circular mouse
- Optional digitizing tablet

For an 8-plane video subsystem, the power-up self-test executes in under 30 seconds. The diagnostic code is contained in 16 kilobytes of ROM space.

5.3.2 Compatibility

The diagnostic code is compatible with the KA630-A processor, the KA630-A console program, the self-test responses specified for the keyboard/mouse/tablet, and the MicroVAX diagnostic ROM 1.0 format.

5.3.3 Hardware Failure

Failures may be nonfatal or fatal. A nonfatal error inhibits further testing of an operation or function of the device. This applies specifically to manufacturing, where external loopback connectors are attached in place of the respective manual input devices such as a keyboard or mouse. When a nonfatal error occurs, the power-up self-test saves the error information and continues testing the video subsystem. If no other error occurs, the power-up self-test updates the lights, and passes control to the KA630 ROM with the error information. A nonfatal error does not override a fatal error or a second nonfatal error.

A fatal error inhibits further testing of the device. Most of the power-up self-test's errors are categorized as fatal. When a fatal error occurs, the power-up self-test saves the error information, updates the VCB02 lights, and passes control to the KA630 ROM with the error information. Fatal errors override nonfatal errors.

5.3.4 Unexpected Traps

The power-up self-test treats unexpected traps as fatal errors.

5.3.5 Power Failures

The diagnostic does not handle power failures.

5.3.6 Restrictions

1. The diagnostic must be executed on a KA630 system. If it is attempted on any other system, the code may be improperly invoked, or not at all.

2. The diagnostic tests a single VCB02 video subsystem residing at the secondary alternate console address, 2001F000H. It does not test any additional VCB02 video subsystems residing at other addresses. However, the diagnostic is callable, so higher level software, such as MicroVMS and ULTRIX, can execute the self-test on any VCB02 video subsystem.

3. The power-up self-test does not isolate faults to the component level.

4. The diagnostic supports the character set supported by the KA630 ROM, which includes ASCII-printable characters, Digital's multinational-printable characters, carriage return, line feed, backspace, and bell. This excludes the remaining 7-bit control characters, 8-bit control characters, delete, and all escape sequences.

5.4 DIAGNOSTIC OPERATIONAL REQUIREMENTS

5.4.1 Hardware

The diagnostic operational requirements for hardware are:

1. KA630 system.
2. VCB02 base module, first VCB02 4-plane module, and optional second VCB02 4-plane module.
3. LK201 keyboard and optional VSXXX-AA mouse or optional VSXXX-BA tablet. External loopback connectors can be attached to both keyboard and mouse ports.
4. VR260 monochromatic or VR290 color monitor.

5.4.2 Software

The diagnostic operational requirements for software are:

1. The KA630 system lights are updated, except for the power-up self-test, which updates the VCB02 lights, only.
2. Memory must be configured as shown in Figure 5-1.

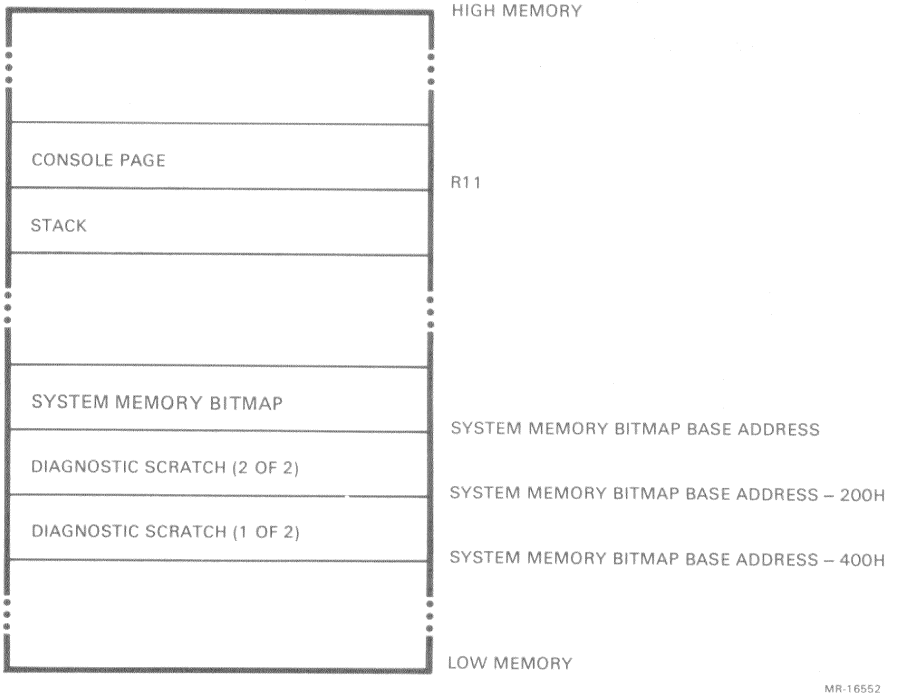


Figure 5-1 Memory Configuration Required for Power-Up Self-Test

3. The console page is configured as listed in Table 5-1.

4. R11 must contain the base address of the console page.

Table 5-1 Console Page Configuration

Address	Description
00H(R11)	Reserved for KA630 ROM
	.
	.
	.
18H(R11)	Reserved for KA630 ROM
1CH(R11)	Get character
20H(R11)	Put character poll
24H(R11)	Put character
28H(R11)	Console enable and bus reset
2CH(R11)	Console enable and reset
30H(R11)	Console reset N
34H(R11)	Keyboard put character
38H(R11)	Software service
3CH(R11)	VCB02 I/O space CSR
40H(R11)	US font table base address
44H(R11)	Multinational font table base address
48H(R11)	Keycode translate character
4CH(R11)	Current character row
4DH(R11)	Minimum character row value
4EH(R11)	Maximum character row value
4FH(R11)	Reserved
50H(R11)	Current character column
51H(R11)	Minimum character column value
52H(R11)	Maximum character column value
53H(R11)	Reserved
54H(R11)	Reserved for keyboard flags
55H(R11)	Reserved
56H(R11)	Reserved
57H(R11)	Reserved
58H(R11)	VCB02 memory space CSR address
5CH(R11)	Wait 1 millisecond
60H(R11)	Reserved
64H(R11)	Boot memory base address
68H(R11)	System memory bitmap base address
6CH(R11)	System memory bitmap length in bytes
70H(R11)	Reserved for KA630 ROM
	.
	.
	.
1FFH(R11)	Reserved for KA630 ROM

5. The VCB02 ROM must be properly mapped, and the diagnostic file must reside in the proper address space, and be verified by the checksum specified in the MicroVAX diagnostic ROM format 1.0 before executing the power-up self-test.

6. In kernel mode, all addresses must agree with the memory management state: if physical, all addresses must be physical; if virtual, all addresses must be virtual.

5.5 FUNCTIONAL DESCRIPTION

5.5.1 Power-Up Self-Test

5.5.1.1 **Parameters Passed and Returned** -- If the power-up self-test is invoked on power-up, no parameters are passed when the VCB02 ROM is executed on power-up. If the power-up self-test is invoked by software control, the following parameters must be passed: function code, console page base address, system control block address, bus adapter offset, device vector offset, bitmap Q22-Bus address, maximum interrupt priority level, minimum interrupt priority level, and a console page. The function code should be 1. Any other function code will execute "no operations" and return a successful status. Refer to the calling sequence for the proper method to pass the parameters.

NOTE

When invoking the power-up self-test by software control, it is important that the addresses agree with the memory management state. For example, if memory management is on, the addresses should be virtual.

If the power-up self-test passes, the procedure returns R0 with a value of 1, and R1 with a 0. If the power-up self-test fails, the procedure returns R0 with a value of 0, and R1 with an error code. If external loopback connectors are attached and the power-up self-test passes, the procedure returns R0 with a value of 2, and R1 with an error code. Refer to Table 5-2 for the error code definitions.

5.5.1.2 **VCB02 Diagnostic LEDs** -- The VCB02 base module diagnostic LEDs or lights (see Figure 1-1) are lit on power-up. The power-up self-test must properly update the VCB02 lights and inform the user of the status of the VCB02 video subsystem. When the power-up self-test is invoked by the KA630 ROM, the VCB02 lights are updated by the VCB02 ROM to indicate that the power-up self-test is being executed. When the power-up self-test is completed, the VCB02 lights are updated to indicate the status of the subsystem. If the VCB02 lights are out, either the system is functioning correctly, or the lights are not working. Otherwise, the VCB02 lights identify the failing FRU, as listed in Table 5-3.

Table 5-2 Error Codes

Error Number	Description
0	Everything is working correctly.
1	Unexpected machine check or VCB02 interrupt.
2	Register access or data fault.
3	Template RAM fault.
4	Address processor failed to interrupt.
5	Address processor interrupt status fault or too many address processor interrupts.
6	Hardware initialization fault.
7	Update video processor enable chip select fault.
8	Bitmap memory data path fault. (first VCB02 4-plane module)
9	Bitmap memory data path fault. (second VCB02 4-plane module)
10	Bitmap memory data fault. (first VCB02 4-plane module)
11	Bitmap memory data fault. (second VCB02 4-plane module)
12	Bitmap memory address fault. (first VCB02 4-plane module)
13	Bitmap memory address fault. (second VCB02 4-plane module)
14	Bitmap memory fault. (VCB02 base module)
15	Right scroll fault. (first VCB02 4-plane module)
16	Right scroll fault. (second VCB02 4-plane module)
17	Right scroll fault. (VCB02 base module)
18	Down scroll fault. (first VCB02 4-plane module)
19	Down scroll fault. (second VCB02 4-plane module)
20	Down scroll fault. (VCB02 base module)
21	DMA to bitmap failed to interrupt.
22	DMA from bitmap failed to interrupt.
23	Data fault during DMA transfer.
24	Video processor fault during DMA test. (first VCB02 4-plane module)
25	Video processor fault during DMA test. (second VCB02 4-plane module)
26	Address processor fault during DMA test. (VCB02 base module)
27	DMA interrupt status fault or too many DMA interrupts.
28	Cursor B failure during horizontal sync test.
29	Horizontal sync did not occur.
30	Draw boundary lines failure.
31	Cursor B failure during green testing.

Table 5-2 Error Codes (Cont)

Error Number	Description
32	Green color map failure.
33	Cursor B failure during red testing.
34	Red color map failure.
35	Cursor B failure during blue testing.
36	Blue color map failure.
37	DUART failed to interrupt.
38	DUART interrupt status fault, too many DUART interrupts, or data error.
39	Keyboard or loopback not connected properly.
40	Keyboard failure.
41	Mouse loopback not connected properly.
42	Mouse failure.
43	Power-up self-test initialization failure.

Table 5-3 VCB02 Light Definitions

Lights D6	D8	D7	Description
During Power-Up Self-Test			
*	*	*	Hardware state on power-up.
0	0	*	Executing power-up self-test.
X	X	X	One of the light definitions described below.
After Power-up Self-Test			
0	0	0	No faults detected.
0	0	*	VCB02 base module fault.
0	*	0	First VCB02 4-plane module fault.
0	*	*	Second VCB02 4-plane module fault.
*	0	0	Not used.
*	0	*	Keyboard fault.
*	*	0	Mouse or tablet fault.
*	*	*	Catastrophic.

* = On
0 = Off

The symbol D6 represents the bottom LED; the symbol D8 represents the middle LED, and the symbol D7 represents the top LED.

5.5.1.3 **Register Access and Data Test** -- This test reads the secondary alternate console CSR and verifies the VCB02 has a valid configuration. Valid configurations include: either a first VCB02 4-plane module, or first and second VCB02 4-plane modules. If no VCB02 4-plane modules, or only the second VCB02 4-plane module, are detected, an error is reported. The test reads each address on a word boundary within the register mapping space, and verifies that each address on a word boundary responds properly. The test does not read the VCB02 ROM, template RAM, and "not used" address space. The VCB02 ROM is assumed to be working correctly. The template RAM is verified in a succeeding test.

The test writes a set of patterns, reads, and compares the patterns with all readable and writable registers. The patterns include the following.

5555H
AAAAH
3333H
0F0FH
00FFH

The registers include the DMA address counter, DMA byte count, FIFO register, vector base, and memory base register. The vector base is initialized.

5.5.1.4 **Template RAM Test** -- This test writes a set of patterns, reads, and compares the patterns with the first template RAM address. The patterns include the following.

5555H
AAAAH
3333H
0F0FH
00FFH

The remaining test consists of three passes. The first pass writes a 5555H pattern from the first template RAM address through the last template RAM address. The second pass reads and compares the pattern, and writes an AAAAH pattern from the first template RAM address through the last template RAM address. The final pass reads and compares the pattern, and writes a 5555H pattern from the last template RAM address through the first template RAM address.

5.5.1.5 **Address Processor Test** -- This test verifies that the address processor can function and interrupt. The test sets up the hardware to handle address processor interrupts, issues the "cancel" address processor command, and waits for an address processor interrupt generated by the I/D data transmit ready line.

5.5.1.6 **Address Processor and Video Processor Initialization Test** This test initializes the address processor and video processor hardware.

5.5.1.7 **Update Video Processor Enable Chip Select Test** -- This test verifies the video processor chip selection. It begins by initializing point [0,0] for all the planes:

1. Select all planes.

2. Using one processor-to-bitmap Z-mode command, write a 0 to point [0,0] of all the planes.

The video processor selection for the first plane is tested with the following steps.

1. Select the first plane and deselect the other planes.

2. Using one processor-to-bitmap Z-mode command, write a 1 to point [0,0] of all the planes. This operation affects only the first plane because it was the only plane selected.

3. Using one bitmap-to-processor Z-mode command, read point [0,0] from all planes.

4. Compare the results. Point [0,0] of the first plane should be a 1, and the remaining point for the other planes should be a 0.

Point [0,0] is again initialized for all the planes. The sequence performed for the first plane is repeated for each of the remaining planes, but references to the first plane are replaced by those for the plane being tested.

5.5.1.8 **Bitmap Memory Test** -- This test verifies the hardware related to the bitmap memory for all the planes. The test consists of three sections: data path and chip selection, data, and addressing. The data path and chip selection section verifies the data path to the bitmap memory and chip selection. Using a processor-to-bitmap X-mode command, a set of 32-bit patterns are written to point [0,0] through point [31,0] for all planes. Using a series of bitmap-to-processor X-mode commands, the pattern is read from the first plane and compared to the pattern, and read from the second plane and compared to the pattern. This continues until every plane is read and compared. The patterns include the following.

55555555H
AAAAAAAAH
33333333H
0F0F0F0FH
00FF00FFH
0000FFFFH
87654321H

The data section verifies that all bits in all planes of bitmap memory can be set and cleared correctly. Using one scroll fill command, a 55H type pattern is written to each address of bitmap memory for all planes. Using a series of bitmap-to-processor X commands, the pattern is read from each address of the first plane and compared, and read from each address of the second plane and compared. This continues until every address of every plane is read and compared. Using the scroll fill command, an AAH-type pattern is written to each address of bitmap memory for all planes. Using the same series of bitmap-to-processor X commands, the pattern is read from each address of the first plane and compared, and read from each address of the second plane and compared. This continues until every address of every plane is read and compared.

The address section verifies the bitmap memory addressing. Using the processor-to-bitmap Z command, a set of 32-bit patterns are written into a set of bitmap memory addresses for all planes. Using a rasterop function, each plane is complemented. Using the bitmap-to-processor X command, the pattern is read from the set of bitmap memory addresses for the first plane and compared to the 32-bit complemented pattern set. The remaining plane reads the complemented values in exactly the same way. The address pattern is a floating 0 and floating 1. It is listed from the lowest value to the highest value in Table 5-4.

Table 5-4 Bitmap Memory Addresses

Byte Address	Pixel Address	Write Pattern	Read Pattern
00000H	(0000, 0000)	00000000H	FFFFFFFFH
00004H	(0032, 0000)	11111111H	EEEEEEEEH
00008H	(0064, 0000)	22222222H	DDDDDDDDH
00010H	(0128, 0000)	33333333H	CCCCCCCCH
00020H	(0256, 0000)	44444444H	BBBBBBBBH
00040H	(0512, 0000)	55555555H	AAAAAAAAH
00080H	(0000, 0001)	66666666H	99999999H
00100H	(0000, 0002)	77777777H	88888888H
00200H	(0000, 0004)	88888888H	77777777H
00400H	(0000, 0008)	99999999H	66666666H
00800H	(0000, 0016)	AAAAAAAAH	55555555H
01000H	(0000, 0032)	BBBBBBBBH	44444444H
02000H	(0000, 0064)	CCCCCCCCH	33333333H
04000H	(0000, 0128)	DDDDDDDDH	22222222H
08000H	(0000, 0256)	EEEEEEEEH	11111111H
10000H	(0000, 0512)	FFFFFFFFH	00000000H
1FFFCH	(0992, 1023)	00000000H	FFFFFFFFH
20000H	(0000, 1024)	11111111H	EEEEEEEEH
2FFFCH	(0992, 1535)	22222222H	DDDDDDDDH
37FFCH	(0992, 1791)	33333333H	CCCCCCCCH
3BFFCH	(0992, 1919)	44444444H	BBBBBBBBH
3DFFCH	(0992, 1983)	55555555H	AAAAAAAAH
3EFFCH	(0992, 2015)	66666666H	99999999H
3F7FCH	(0992, 2031)	77777777H	88888888H
3FBFCH	(0992, 2039)	88888888H	77777777H
3FDFCH	(0992, 2043)	99999999H	66666666H
3FEFCH	(0992, 2045)	AAAAAAAAH	55555555H
3FF7CH	(0992, 2046)	BBBBBBBBH	44444444H
3FFBCH	(0480, 2047)	CCCCCCCCH	33333333H
3FFDCH	(0736, 2047)	DDDDDDDDH	22222222H
3FFECH	(0864, 2047)	EEEEEEEEH	11111111H
3FFF4H	(0928, 2047)	FFFFFFFFH	00000000H
3FFF8H	(0960, 2047)	00000000H	FFFFFFFFH
3FFFCH	(0992, 2047)	11111111H	EEEEEEEEH

5.5.1.9 **Right Scroll Test** -- This test verifies the scrolling logic in the video processors, and the timing logic associated with the video processors, memory, and the address processor. To test this function a scroll region will be established and filled with a known color with a rasterop function. The scroll boundaries must be on a nibble boundary to use the barrel shifter in the video processor to the fullest. The scroll function causes the video processors to scroll the region, exactly half the distance of the region. In order to test the right scroll function fully, the maximum value of 15 is used. A right scroll of 15 pixels actually develops into a scroll of 16 pixels in the bitmap. Therefore, the scroll region is 32 pixels wide, with the first half filled with AAAAs and the second half filled with 5555s. The horizontal scroll registers is set up to:

```
Scroll X min. = 4
Scroll X max. = scroll X min. + 8
Scroll Y min. = 0
Scroll Y max. = scroll Y min. + 32
(Video processor) scroll constant = 3
```

After the scroll function is performed, the bitmap reads back from each plane, and verifies that both the scroll fill color is in the left half of the block and the old left half has moved to the right half of the block. A different value of the scroll fill color is used in each video processor (the value is the same as the video processor's logical number: 1, 2, 3, etc.). In addition, the area just outside the scrolling region is checked to verify that the scroll edges are correct. Figure 5-2 illustrates the changes of the test.

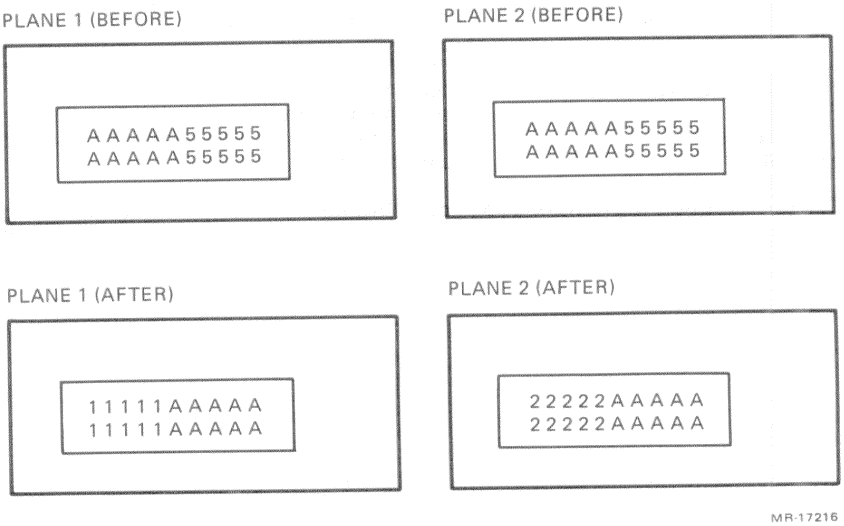


Figure 5-2 Right Scroll Test Changes

5.5.1.10 Down-Scroll Test -- Down-scrolling is implemented by forcing all the video processors to scroll up all data outside the scroll region, and fill the top of the area within the scroll region with the fill color for the same number of scan lines as the distance scrolled down.

To test this function, an area of memory is filled with a known pattern and the down-scroll is performed. Reading the bitmaps discloses if the function is properly performed. In addition, a test is conducted of the system's ability to scroll some bitmaps while not scrolling others. This also tests the scroll enable register, which resides on the I/D bus. The vertical scroll registers are set up to:

```
Scroll X min. = 320
Scroll X max. = scroll X min. + 16
Scroll Y min. = 422
Scroll Y max. = 442
Scroll constant = 10
```

The region to be scrolled is filled with a known pattern of AAAAs (see Figure 5-3), while the rest of the bitmap is filled with a pattern of zeros. The scroll fill register of each video processor is loaded with the logical ascending order of bit patterns from 1111 to 8888. After the scroll is complete, the bitmap region verifies that it:

- Scrolled, if the plane was selected
- Filled, if the plane was not selected
- Region exterior for proper edge masking by the address processor and video processor.

5.5.1.11 DMA Test -- The FIFO and DMA operations are checked by performing a processor-to-bitmap transfer of 65 words immediately followed by a bitmap-to-processor transfer of 65 words. After this the data sent is compared with the data received. To verify the interrupts, the test is "interrupt driven."

5.5.1.12 Video Synchronization Pulse Test -- Presence of the horizontal sync pulse is verified by monitoring the sync detect bit.

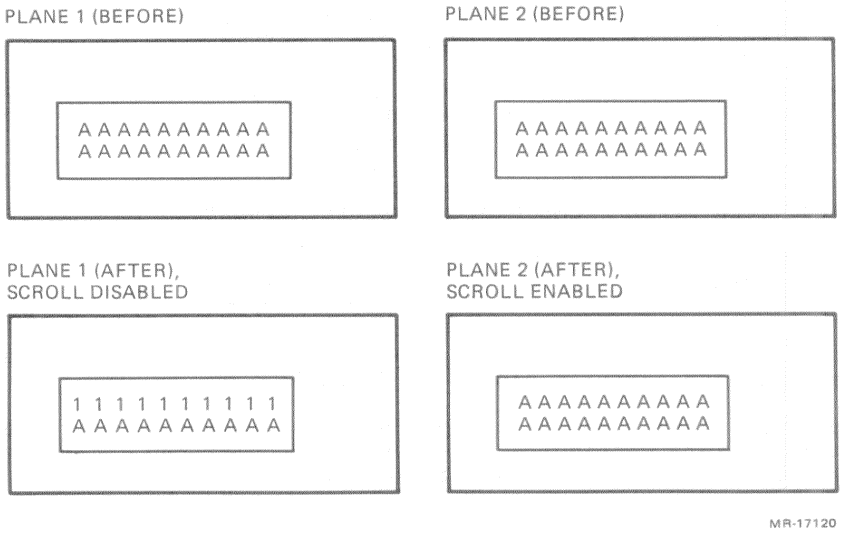


Figure 5-3 Down-Scroll Test Changes

5.5.1.13 Video Signal Level Test -- This test checks the data path from the Q22-Bus to the complementary output of the video DACs. It also checks the accuracy of the video DACs to within 2 LSBs. The resolution of the comparators determines the final delta value that is blasted into the VCB02 ROM for the final product. This means the two LSBs of each color map register address may, or may not, be checked by the self-test.

During this test the color map register's red, green, and blue entries are loaded 3 times; once for each group of 9 lines. This test requires 27 frames of displayed screen data to complete. Nine horizontal lines are drawn on the screen, as listed in Table 5-5.

Table 5-5 Horizontal Scan Line Pixel Values

Scan Line	Pixel Value	
0	0	
1	1	
2	2	
3	4	
4	8	
5	10H	8-plane only
6	20H	8-plane only
7	40H	8-plane only
8	80H	8-plane only

To test the green set of pixel values, the color map locations are loaded with the patterns listed in Table 5-6.

To ensure address fault detection, the preceding test must be completed before continuing. The remaining color map locations are loaded with the following background pattern.

CMR Red Value	CMR Green Value	CMR Blue Value	Expected Results: B-GT-R	G-GT-B	R-GT-G
B-DV	B+DV	DBH	1	1	0

To test the red set of pixel values, the color map locations are loaded with the patterns listed in Table 5-7.

To ensure address fault detection, the preceding test must be completed before continuing. The remaining color map locations are loaded with the following background pattern.

CMR Red Value	CMR Green Value	CMR Blue Value	Expected Results: B-GT-R	G-GT-B	R-GT-G
B-DV	B+DV	DBH	1	1	0

To test the blue set of pixel values, the color map locations are loaded with the patterns listed in Table 5-8.

To ensure address fault detection, the preceding test must be completed before continuing. The remaining color map locations are loaded with the following background pattern.

CMR Red Value	CMR Green Value	CMR Blue Value	Expected Results: B-GT-R	G-GT-B	R-GT-G
G+DV	DBH	G-DV	1	1	0

The DV, or delta value, is 10.

5.5.1.14 DUART Test -- Using local loopback operating mode, this test verifies the DUART interface by transmitting and receiving a set of 8-bit patterns at 4800 baud. The patterns include:

55H
AAH
33H
0FH

To verify the interrupts, the test is "interrupt-driven."

Table 5-6 Color Map Register Green Pixel Value Test

CMR Address	CMR Red Value	CMR Green Value	CMR Blue Value	Expected Results: B-GT-R	G-GT-B	R-GT-G
0	G+DV	55H	G-DV	0	1	1
1	G+DV	AAH	G-DV	0	1	1
2	G+DV	33H	G-DV	0	1	1
4	G+DV	CCH	G-DV	0	1	1
8	G+DV	0FH	G-DV	0	1	1
10H	G+DV	F0H	G-DV	0	1	1
20H	G+DV	FFH-DV	G-DV	0	1	1
40H	G+DV	00H+DV	G-DV	0	1	1
80H	G+DV	EFH	G-DV	0	1	1

Table 5-7 Color Map Register Red Pixel Value Test

CMR Address	CMR Red Value	CMR Green Value	CMR Blue Value	Expected Results: B-GT-R	G-GT-B	R-GT-G
0	55H	R-DV	R+DV	1	0	1
1	AAH	R-DV	R+DV	1	0	1
2	33H	R-DV	R+DV	1	0	1
4	CCH	R-DV	R+DV	1	0	1
8	0FH	R-DV	R+DV	1	0	1
10H	F0H	R-DV	R+DV	1	0	1
20H	FFH-DV	R-DV	R+DV	1	0	1
40H	00H+DV	R-DV	R+DV	1	0	1
80H	EFH	R-DV	R+DV	1	0	1

Table 5-8 Color Map Register Blue Pixel Value Test

CMR Address	CMR Red Value	CMR Green Value	CMR Blue Value	Expected Results: B-GT-R	G-GT-B	R-GT-G
0	B-DV	B+DV	55H	1	1	0
1	B-DV	B+DV	AAH	1	1	0
2	B-DV	B+DV	33H	1	1	0
4	B-DV	B+DV	CCH	1	1	0
8	B-DV	B+DV	0FH	1	1	0
10H	B-DV	B+DV	F0H	1	1	0
20H	B-DV	B+DV	FFH-DV	1	1	0
40H	B-DV	B+DV	00H+DV	1	1	0
80H	B-DV	B+DV	EFH	1	1	0

5.5.1.15 **Manual Input Devices Test** -- This test verifies the manual input devices connected to the DUART's ports. The test sends the "test command" (FDH) to the keyboard port to invoke its self-test, and waits one second for a response from the device. If the device responds with four bytes, and the third byte is zero, the test continues. If the device responds with one byte of value FDH, the test logs a hard error and continues. Otherwise, the test interprets a device fatal error, and returns control to the KA630 ROM.

The test sends the "test command" (T ASCII) to the other port. If the keyboard port received four bytes, and the device responds with four bytes, and the third byte is less than 20H (or the device fails to respond), the test continues. If the keyboard port received one byte of value FDH, and the device responds with one byte of value T ASCII, the test logs a hard error and continues.

NOTE

The device connected to the mouse/tablet port is considered an optional device. This means that in the customer's environment, a mouse or tablet that is improperly connected may not be reported as an error.

5.5.1.16 Calling Sequence --

Power-Up-Initiated

```
; Set up the console page, stack, bitmap (used as work buffer),
; diagnostic scratch pages.
          .
          .
          .
; Map VCB02 ROM

          MOVW     #^X3F, @#QDSS$GW_IOCSR

; Find base address of executable code in "DIAG0    " file
; and return in Rn.
          .
          .
          .
; Pass control to self-test.

          CALLS    #0, (Rn)
```

Software-Initiated

```
; Suggested literal values

SIST$K_IPLMAX    = ^X17          ; Maximum interrupt priority level
SIST$K_IPLMIN    = ^X2           ; Minimum interrupt priority level
SIST$K_DVCVEC    = ^X140         ; Device vector offset
SIST$K_BUSADP    = ^X200         ; Bus adapter offset
SIST$K_FNCCODE   = 1             ; Function code

; Set up the console page, stack, bitmap (used as work buffer),
; diagnostic scratch pages, change to kernel mode.
          .
          .
          .
; Map VCB02 ROM

          MOVW     #^X3F, @#QDSS$GW_IOCSR

; Find base address of executable code in "DIAG0      " file
; and return in Rn.
          .
          .
          .
; Pass parameters.  Note that some physical addresses can easily
; be obtained, but most Q22-Bus addresses and virtual addresses
; cannot.

BLBC    R0, ERROR                ; Branch if error
PUSHL   #SIST$K_IPLMAX           ; Maximum interrupt priority level
PUSHL   #SIST$K_IPLMIN           ; Minimum interrupt priority level
PUSHL   #SIST$K_BITMAPQBUS       ; Bitmap Q22-Bus base address
PUSHL   #SIST$K_DVCVEC           ; Device vector offset
PUSHL   #SIST$K_BUSADP           ; Bus adapter offset
PUSHL   SIST$L_SCBB              ; System control block base address
PUSHL   SIST$L_CPADR             ; Console page address
PUSHL   #SIST$K_FNCCODE          ; Function code (should be 1)

; Pass control to self-test.
CALLS    #8, (Rn)
```

5.5.2 Console I/O

5.5.2.1 Put Character Poll -- This routine is called to determine if a character can be displayed. No parameters are passed to the routine. If a character can be displayed, the routine returns R0 with bit 0 set to 1. If a character cannot be displayed, the routine returns R0 with bit 0 cleared to 0.

Calling Sequence

```
; R11 contains the base address of the console page.

LOOP:
JSB     @CP$A_PUTCHRPOLL(R11)    ; Put a character?
BLBC    R0, LOOP                 ; No, wait
```

5.5.2.2 **Put Character** -- This routine is called to display a character. R1 is passed containing the "zero extended" ASCII value of the character to be displayed. If the character is displayed successfully, the routine returns R0 with bit 0 set to 1. If the character is displayed unsuccessfully, the routine returns R0 with bit 0 cleared to 0.

NOTE

The routine should be used with the put character poll routine. It uses the US ASCII font table and multinational font table in the KA630 ROM.

Calling Sequence

```
; R11 contains the base address of the console page.

LOOP:
JSB     @CP$A_PUTCHRPOLL(R11)    ; Put a character?
BLBC    R0, LOOP                 ; No, wait
MOVZBL  #^A"A", R1               ; Pass ASCII value of "A"
JSB     @CP$A_PUTCHR(R11)        ; Put the character
BLBC    R0, ERROR                ; Branch if error
```

5.5.2.3 **Get Character** -- This routine is called to determine if a character has been received, and if so, return the character. No parameters are passed. If a character is available, the routine returns R1 with the "zero extended" ASCII value of the character, and R0 with bit 0 set to 1. If a character is not available, the routine returns R1 with arbitrary data and R0 is cleared to 0. If the keyboard code is a shift code like "control," "shift," and "lock," or introducer code like "compose character," the routine returns R0 with bit 0 cleared to 0.

NOTE

Get character calls the keycode translate character routine. There are no provisions for timeouts.

Calling Sequence

```
; R11 contains the base address of the console page.

LOOP1:
JSB     @CP$A_GETCHR(R11)        ; Get a character?
BLBC    R0, LOOP1                ; No, wait
PUSHL   R1                       ; Yes, store it

LOOP2:
JSB     @CP$A_PUTCHRPOLL(R11)    ; Put a character?
BLBC    R0, LOOP2                ; No, wait
POPL    R1                       ; Restore R1
JSB     @CP$A_PUTCHR(R11)        ; Echo the character
BLBC    R0, ERROR                ; Branch if error
```

5.5.2.4 **Enable VCB02 ROM and Console Bus Reset** -- This routine is called to initialize the hardware for console output after a bus reset. Refer to Table 5-9. If the VCB02 is initialized successfully, the routine returns R0 with bit 0 set to 1. If the VCB02 is initialized unsuccessfully, the routine returns R0 with bit 0 cleared to 0. The software service routine is called if initialized.

NOTE
This is not provided by the VCB02 ROM. The routine calls N reset.

Calling Sequence

```
; R11 contains the base address of the console page.

JSB     @CP$A_ENBUSRESET(R11)    ; Enable ROM and bus reset
BLBC    R0, ERROR                ; Branch if error
```

Table 5-9 Bus Reset and Reset Defaults

Default	Setting
Console Page	
Software service	Called if not zero
Address Processor	
Scroll X min.	8 * (Character column min.)
Scroll X max.	8 * (Character column max. + 1)
Scroll Y min.	15 * (Character row min.)
Scroll Y max.	15 * (Character row max. + 1)
Clip X min.	8 * (Character column min.)
Clip X max.	8 * (Character column max. + 1)
Clip Y min.	15 * (Character row min.)
Clip Y max.	15 * (Character row max. + 1)
Fast source 1 DX	8
Fast source 1 DY	15
Source 1 X origin	0
Source 1 Y origin	0
Fast destination DX	8
Fast destination DY	0
Slow destination DX	0
Slow destination DY	15

Table 5-9 Bus Reset and Reset Defaults (Cont)

Default	Setting
Video Processor	
Resolution mode	0
Bus width	CH
Scroll constant	60H
Video processor 0 plane address	0
Video processor 1 plane address	1
Video processor 2 plane address	2
Video processor 3 plane address	3
Video processor 4 plane address	4
Video processor 5 plane address	5
Video processor 6 plane address	6
Video processor 7 plane address	7
Mask 1	FFFFH
Source	FFFFH
Fill	0
Left scroll boundary	0
Right scroll boundary	FFFFH
Background color	0
Foreground color	FFFFH
CSR 0	28H
CSR 1	4
CSR 2	0
CSR 4	24H
CSR 5	8H
CSR 6	1

Color Map

Entry	Red	Green	Blue
0	0	0	0
1	255	255	255
2	0	0	0
3	255	255	255
4	0	0	0
5	255	255	255
	.	.	.
	.	.	.
	.	.	.
254	0	0	0
255	255	255	255

Table 5-9 Bus Reset and Reset Defaults (Cont)

Default	Setting
DUART	
Channel A	
Character length	8
Parity	Off
Baud rate	4800
Transmitter	Reset and enabled
Receiver	Reset and enabled
Channel B	
Character length	8
Parity	Odd
Baud rate	4800
Transmitter	Reset and enabled
Receiver	Reset and enabled

5.5.2.5 **Enable VCB02 ROM and Console Reset** -- This routine is called to initialize the hardware for console output after a reset. Refer to Table 5-9. If the VCB02 is initialized successfully, the routine returns R0 with bit 0 set to 1. If the VCB02 is initialized unsuccessfully, the routine returns R0 with bit 0 cleared to 0. The software service routine is called if initialized.

NOTE

This is not provided by the VCB02 ROM. The routine calls N reset.

Calling Sequence

```
; R11 contains the base address of the console page.

JSB     @CP$A_ENRESET(R11)      ; Enable ROM and reset
BLBC    R0, ERROR               ; Branch if error
```

5.5.2.6 **N Reset** -- This routine is called by the reset, bus reset, and power-up to initialize the hardware for console output. R1 is passed containing a function code, where:

0 = Reset
1 = Bus reset
2 = Power-up self-test reset.

Refer to Tables 5-9 and 5-10. If the VCB02 is initialized successfully, the routine returns R0 with bit 0 set to 1. If the VCB02 is initialized unsuccessfully, the routine returns R0 with bit 0 cleared to 0.

Calling Sequence

```
; R11 contains the base address of the console page.

MOVL    #RESET$K_FNCCODE, R1    ; Pass function code
JSB     @CP$A_NRESET(R11)       ; Reset console
BLBC    R0, ERROR               ; Branch if error
```

Table 5-10 Power-Up Defaults

Default	Setting
Console Page Diagnostics	
Current row	Minimum row
Minimum row	0
Maximum row	56
Current column	Minimum column
Minimum column	0
Maximum column	127
Software service	0
Address Processor	
Request enable	0
Interrupt enable	0
I/D data	FIFO cleared
Command	Issued cancel command
Mode	0
Scroll X min.	0
Scroll X max.	1024
Scroll Y min.	0
Scroll Y max.	855
Pause	0
Y offset	0
Y scroll constant	2000H
Pending X index	0
Pending Y index	0
New X index	0
New Y index	0
Old X index	0
Old Y index	0
Clip X min.	0
Clip X max.	1024
Clip Y min.	0
Clip Y max.	855
Fast source 1 DX	8
Fast source 1 DY	15

Table 5-10 Power-Up Defaults (Cont)

Default	Setting
Source 1 X origin	0
Source 1 Y origin	0
Destination X origin	0
Destination Y origin	0
Fast destination DX	8
Fast destination DY	0
Slow destination DX	0
Slow destination DY	15
Fast scale	1FFFH
Slow scale	1FFFH
Source 2 X origin	0
Source 2 Y origin	0
Source 2 height and width	22H
Y scan count 0	135FH
Y scan count 1	3363H
Y scan count 2	2366H
Y scan count 3	388H
X scan configuration	C8H
X limit	1024
Y limit	2048
X scan count 0	2800H
X scan count 1	1020H
X scan count 2	3AH
X scan count 3	38F0H
X scan count 4	6128H
X scan count 5	93AH
X scan count 6	313CH
Sync phase	100H
Video Processor	
Resolution mode	0
Bus width	CH
Scroll constant	60H
Video processor 0 plane address	0
Video processor 1 plane address	1
Video processor 2 plane address	2
Video processor 3 plane address	3
Video processor 4 plane address	4
Video processor 5 plane address	5
Video processor 6 plane address	6
Video processor 7 plane address	7
Mask 1	FFFFH
Source	FFFFH
Fill	0
Left scroll boundary	0
Right scroll boundary	FFFFH
Background color	0
Foreground color	FFFFH
CSR 0	28H

Table 5-10 Power-Up Defaults (Cont)

Default	Setting
CSR 1	4
CSR 2	0
CSR 4	24H
CSR 5	8H
CSR 6	1

Bitmap Memory

All zeros. Only the first plane is used.

Color Map

Entry	Red	Green	Blue
0	0	0	0
1	255	255	255
2	0	0	0
3	255	255	255
4	0	0	0
5	255	255	255
	.	.	.
	.	.	.
	.	.	.
254	0	0	0
255	255	255	255

DUART

Channel A	
Character length	8
Parity	Off
Baud rate	4800
Transmitter	Reset and enabled
Receiver	Reset and enabled
Channel B	
Character length	8
Parity	Odd
Baud rate	4800
Transmitter	Reset and enabled
Receiver	Reset and enabled

Table 5-10 Power-Up Defaults (Cont)

Default	Setting
Keyboard	
Main array keys	Down
Numeric keypad keys	Down
Delete key	Down
Return, tab keys	Down
Lock, compose keys	Down
Shift, control keys	Down/up
Horizontal cursor keys	Down
Vertical cursor keys	Down
Edit keys	Down
Function keys	Down

5.5.2.7 **Keyboard Put Character** -- This routine is called to send a character to the keyboard, which responds with functions such as illuminating the keyboard LEDs and sounding the bell. R0 is passed containing the base address of a string terminated with a zero byte. If the character is sent to the keyboard successfully, the routine returns R0 with bit 0 set to 1. If the character is sent to the keyboard unsuccessfully, the routine returns R0 with bit 0 cleared to 0.

Calling Sequence

```
; R11 contains the base address of the console page.

MOVAB    BELL$T, R0              ; Pass string (sound bell)
JSB      @CP$A_KBDPUTCHR(R11)    ; Send to keyboard
BLBC     R0, ERROR               ; Branch if error
.
.
.
BELL$T: .BYTE    ^XA7, 0         ; String to sound bell
```

5.5.2.8 **Software Service** -- This routine is used to provide extended software functions to higher level software applications, such as operating systems. When the entry is zero, no additional function is performed. When the entry is nonzero, it is expected to contain an address of a subroutine. The subroutine is called with R1 containing a function code. The higher level software application is responsible for providing the address of the routine. Currently, function code 1 has been defined as saving the state of the alternate console device.

Calling Sequence

```
; R11 contains the base address of the console page.

TSTL    @CP$A_SWSERV(R11)       ; Routine?
BEQL    SKIP                    ; Branch if no
MOVL    #1, R1                  ; Pass function code
JSB     @CP$A_SWSERV(R11)       ; Save state

SKIP:
```

5.5.2.9 **VCB02 I/O Space CSR Address** -- This entry contains the address of the VCB02 I/O space CSR address. If the software is executed in physical, the address should be one of the following.

20001F00H (secondary alternate console)
20001F02H
20001F04H
20001F06H
20001F08H
20001F0AH
20001F0CH
20001F0EH

NOTE
This is not provided by the VCB02 ROM.

5.5.2.10 **U.S. Font Table** -- This entry contains the base address of the character font table for displaying U.S. characters. The table is used by the put character routine.

NOTE
This is not provided by the VCB02 ROM.

5.5.2.11 **Multinational Font Table** -- This entry contains the base address of the character font table for displaying multinational characters. The table is used by the put character routine.

NOTE
This is not provided by the VCB02 ROM.

5.5.2.12 **Keycode Translate Character** -- This entry contains the address of a routine responsible for updating the "shift," "shift lock," and "control" key state, and for properly translating the keyboard codes to the correct character representation. It is used by the get character routine. R2 is passed with the keycode value.

NOTE
This is not provided by the VCB02 ROM.

Calling Sequence

```
; R11 contains the base address of the console page.

MOVZBL  #KEYCODE$K, R2          ; Pass keycode
JSB     @CP$A_KBDX(R11)         ; Translate keycode
```

5.5.2.13 **Row and Column Parameters** -- The current character row and column contain the position of the next character to be displayed in character units. The minimum character row and column contain the minimum values in character units. The maximum character row and column contain the maximum values in character units. The minimum and maximum values are used to modify the console window size.

5.5.2.14 **VCB02 Memory Space CSR Base** -- This entry contains the base address of the VCB02 memory space CSRs. If the software is executed in physical mode, this is a physical address, such as 303F0000H.

NOTE

This is not provided by the VCB02 ROM.

5.5.2.15 **Wait 1 Millisecond** -- The wait 1 millisecond routine is used to create a delay of 1 millisecond. No parameters are passed. It is recommended that future implementations of this routine use the stack, and specifically not use any general-purpose registers. For example:

```
CP$WAITMS:

MOVL    #1MS$K, -(SP)           ; Set counter
LOOP:   SOBGTR  (SP), LOOP      ; Delay loop
TSTL    (SP)+                   ; Clean up stack
RSB                             ; Exit
```

NOTE

This is not provided by the VCB02 ROM.

Calling Sequence

```
; R11 contains the base address of the Console Page.

JSB     @CP$A_WAITMS(R11)       ; Delay 1 millisecond
```

APPENDIX A
Q22-BUS SPECIFICATION

A.1 GENERAL DESCRIPTION

The Q22-Bus, also known as the extended LSI-11 Bus, is the low-end member of Digital's bus family. All of Digital's microcomputers, such as the MicroVAX I, MicroVAX II, and MicroPDP-11, use the Q22-Bus.

The Q22-Bus consists of 42 bidirectional and 2 unidirectional signal lines. These form the lines along which the processor, memory, and I/O devices communicate with each other.

Addresses, data, and control information are sent along these signal lines, some of which contain time-multiplexed information. The lines are divided as follows.

- Sixteen multiplexed data/address lines -- BDAL<15:00>
- Two multiplexed address/parity lines -- BDAL<17:16>
- Four extended address lines -- BDAL<21:18>
- Six data transfer control lines -- BBS7, BDIN, BDOUT, BRPLY, BSYNC, BWTBT
- Six system control lines -- BHALT, BREF, BEVNT, BINIT, BDCOK, BPOK
- Ten interrupt control and direct memory access control lines -- BIAKO, BIAKI, BIRQ4, BIRQ5, BIRQ6, BIRQ7, BDMGO, BDMR, BSACK, BDMGI

In addition, a number of power, ground, and space lines are defined for the bus. Refer to Table A-1 for a detailed description of these lines.

The discussion in this Appendix applies to the general 22-bit physical address capability. All modules used with the KA630-A CPU module must use 22-bit addressing.

Most Q22-Bus signals are bidirectional and use terminations for a negated (high) signal level. Devices connect to these lines via high-impedance bus receivers and open collector drivers. The asserted state is produced when a bus driver asserts the line low.

Although bidirectional lines are electrically bidirectional (any point along the line can be driven or received), certain lines are functionally unidirectional. These lines communicate to or from a bus master (or signal source), but not both. Interrupt acknowledge (BIAK) and direct memory access grant (BDMG) signals are physically unidirectional in a daisy-chain fashion. These signals originate at the processor output signal pins. Each is received on device input pins (BIAKI or BDMGI) and is conditionally retransmitted via device output pins (BIAKO or BDMGO). These signals are received from higher-priority devices and are retransmitted to lower-priority devices along the bus, establishing the position-dependent priority scheme.

A.1.1 Master/Slave Relationship

Communication between devices on the bus is asynchronous. A master/slave relationship exists throughout each bus transaction. Only one device has control of the bus at any one time. This controlling device is termed the bus master, or arbiter. The master device controls the bus when communicating with another device on the bus, termed the slave.

The bus master (typically the processor or a DMA device) initiates a bus transaction. The slave device responds by acknowledging the transaction in progress and by receiving data from, or transmitting data to, the bus master. Q22-Bus control signals transmitted or received by the bus master or bus slave device must complete the sequence according to bus protocol.

The processor controls bus arbitration, that is, which device becomes bus master at any given time. A typical example of this relationship is a disk drive, as master, transferring data to memory as slave. Communication on the Q22-Bus is interlocked so that, for certain control signals issued by the master device, there must be a response from the slave in order to complete the transfer. It is the master/slave signal protocol that makes the Q22-Bus asynchronous. The asynchronous operation precludes the need for synchronizing with, and waiting for, clock pulses.

Since bus cycle completion by the bus master requires response from the slave device, each bus master must include a timeout error circuit that aborts the bus cycle if the slave does not respond to the bus transaction within 10 μs. The actual time before a timeout error occurs must be longer than the reply time of the slowest peripheral or memory device on the bus.

A.2 Q22-BUS SIGNAL ASSIGNMENTS

Table A-1 lists the signal assignments for the data/address, control, power/ground, and spare functions of the Q22-Bus.

Table A-1 Signal Assignments

Name	Pin Assignment
DATA AND ADDRESS	
BDAL0	AU2
BDAL1	AV2
BDAL2	BE2
BDAL3	BF2
BDAL4	BH2
BDAL5	BJ2
BDAL6	BK2
BDAL7	BL2
BDAL8	BM2
BDAL9	BN2
BDAL10	BP2
BDAL11	BR2
BDAL12	BS2
BDAL13	BT2
BDAL14	BU2
BDAL15	BV2
BDAL16	AC1
BDAL17	AD1
BDAL18	BC1
BDAL19	BD1
BDAL20	BE1
BDAL21	BF1
CONTROL	
Data Control	
BDOUT	AE2
BRPLY	AF2
BDIN	AH2
BSYNC	AJ2
BWTBT	AK2
BBS7	AP2
Interrupt Control	
BIRQ7	BP1
BIRQ6	AB1
BIRQ5	AA1
BIRQ4	AL2
BIAKO	AN2
BIAKI	AM2
DMA Control	
BDMR	AN1
BSACK	BN1
BDMGO	AS2
BMDGI	AR2

Table A-1 Signal Assignments (Cont)

Name	Pin Assignment
System Control	
BHALT	AP1
BREF	AR1
BEVNT	BR1
BINIT	AT2
BDCOK	BA1
BPOK	BB1
POWER AND GROUND	
+5B (battery) or +12B (battery)	AS1
+12B	BS1
+5B	AV1
+5	AA2
+5	BA2
+5	BV1
+12	AD2
+12	BD2
+12	AB2
-12	AB2
-12	BB2
GND	AC2
GND	AJ1
GND	AM1
GND	AT1
GND	BC2
GND	BJ1
GND	BM1
GND	BT1
SPARES	
SSpare1	AE1
SSpare3	AH1
SSpare8	BH1
SSpare2	AF1
MSpareA	AK1
MSpareB	AL1
MSpareB	BK1
MSpareB	BL1
PSpare1	AU1
ASpare2	BU1

A.3 DATA TRANSFER BUS CYCLES

Data transfer bus cycles are listed and defined in Table A-2.

These bus cycles, executed by bus master devices, transfer 32-bit words or 8-bit bytes to or from slave devices. In block mode, multiple words may be transferred to sequential word addresses, starting from a single bus address. The bus signals listed in Table A-3 are used in the data transfer operations described in Table A-2.

Table A-2 Data Transfer Operations

Bus Cycle Mnemonic	Description	Function (with Respect to the Bus Master)
DATI	Data word input	Read
DATO	Data word output	Write
DATOB	Data byte output	Write-byte
DATIO	Data word input/output	Read-modify-write
DATIOB	Data word input/byte output	Read-modify-write byte
DATBI	Data block input	Read block
DATBO	Data block output	Write block

Table A-3 Bus Signals for Data Transfers

Mnemonic	Description	Function
BDAL<21:00> L	22 Data/address lines	BDAL<15:00> L are used for word and byte transfers. BDAL<17:16> L are used for extended addressing, memory parity error (16), and memory parity error enable (17), functions. BDAL<21:18> L are used for extended addressing beyond 256 Kbytes.
BSYNC L	Bus cycle control	Indicates bus transaction in progress.
BDIN L	Data input indicator	Strobe signals.
BDOUT L	Data output indicator	Strobe signals.
BRPLY L	Slave's acknowledge of bus cycle	Strobe signals.
BWTBT L	Write/byte control	Control signals.
BBS7	I/O device select	Indicates address is in the I/O page.

Data transfer bus cycles can be reduced to five basic types: DATI, DATO(B), DATIO(B), DATBI, and DATBO. These transactions occur between the bus master and one slave device selected during the addressing portion of the bus cycle.

A.3.1 Bus Cycle Protocol

Before initiating a bus cycle, the previous bus transaction must have been completed (BSYNC L negated) and the device must become bus master. The bus cycle can be divided into two parts: an addressing portion, and a data transfer portion. During the addressing portion, the bus master outputs the address for the desired slave device, memory location, or device register. The selected slave device responds by latching the address bits and holding this condition for the duration of the bus cycle until BSYNC L becomes negated. During the data transfer portion, the actual data transfer occurs.

A.3.2 Device Addressing

The device addressing portion of a data transfer bus cycle comprises an address setup and deskew time, and an address hold and deskew time. During the address setup and deskew time, the bus master does the following.

- Asserts BDAL<21:00> L with the desired slave device address bits
- Asserts BBS7 L if a device in the I/O page is being addressed
- Asserts BWTBT L if the cycle is a DATO(B) or DATBO bus cycle

During this time, the address, BBS7 L, and BWTBT L signals are asserted at the slave bus receiver for at least 75 ns before BSYNC goes active. Devices in the I/O page ignore the 9 high-order address bits BDAL<21:13>, and instead, decode BBS7 L along with the 13 low-order address bits. An active BWTBT L signal during address setup time indicates that a DATO(B) or DATBO operation follows, while an inactive BWTBT L indicates a DATI, DATBI, or DATIO(B) operation.

The address hold and deskew time begins after BSYNC L is asserted.

The slave device uses the active BSYNC L bus received output to clock BDAL address bits, BBS7 L, and BWTBT L into its internal logic. BDAL<21:00> L, BBS7 L, and BWTBT L remain active for 25 ns (minimum) after BSYNC L bus receiver goes active. BSYNC L remains active for the duration of the bus cycle.

Memory and peripheral devices are addressed similarly, except for the way the slave device responds to BBS7 L. Addressed peripheral devices must not decode address bits on BDAL<21:13> L. Addressed peripheral devices may respond to a bus cycle when BBS7 L is asserted (low) during the addressing portion of the cycle. When asserted, BBS7 L indicates that the device address resides in the

I/O page (the upper 4 K address space). Memory devices generally do not respond to addresses in the I/O page; however, some system applications may permit memory to reside in the I/O page for use as DMA buffers, read-only memory bootstraps, diagnostics, etc.

DATI -- The DATI bus cycle, shown in Figure A-1, is a read operation. During DATI, data is input to the bus master. Data consists of 16-bit word transfers over the bus. During the data transfer portion of the DATI bus cycle, the bus master asserts BDIN L 100 ns (minimum) after BSYNC L is asserted. The slave device responds to BDIN L active as follows.

- Asserts BRPLY L 0 ns (minimum) (8 ns maximum to avoid bus timeout) after receiving BDIN L, and 125 ns (maximum) before BDAL bus driver data bits are valid.
- Asserts BDAL<21L00> L with the addressed data and error information 0 ns (minimum) after receiving BDIN, and 125 ns (maximum) after assertion of BRPLY.

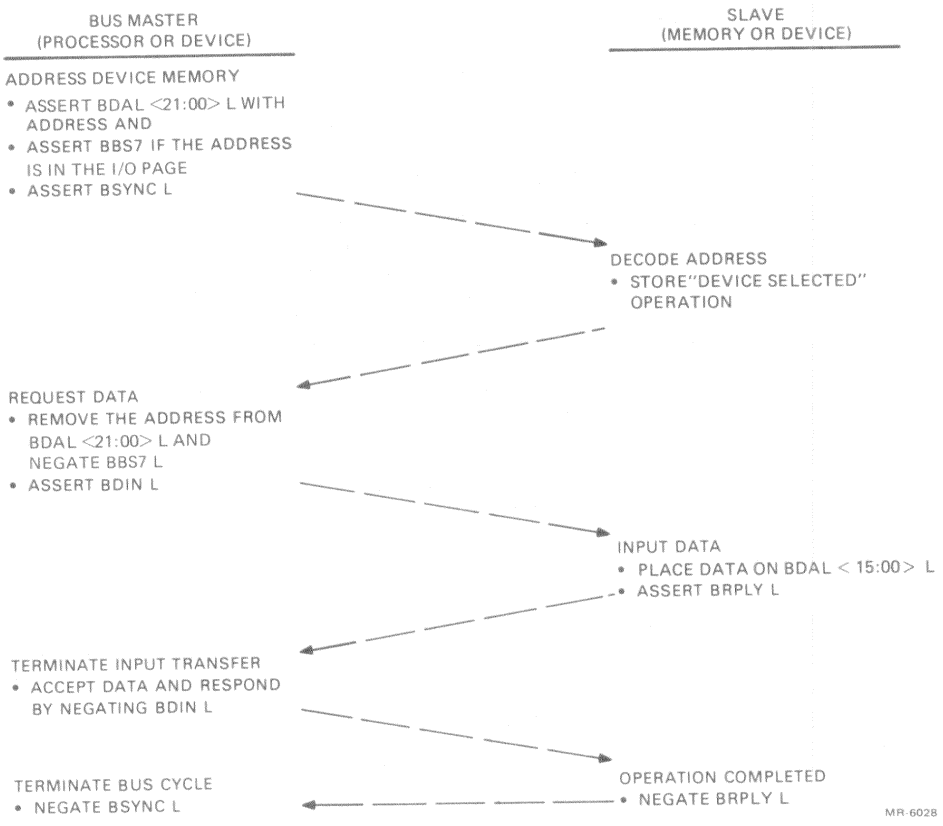


Figure A-1 DATI Bus Cycle

When the bus master receives BRPLY L, it does the following.

- Waits at least 200 ns deskew time and then accepts input data at BDAL<17:00> L bus receivers. BDAL <17:16> L are used for transmitting parity errors to the master.

- Negates BDIN L 200 ns (minimum) to 2 μs (maximum) after BRPLY L goes active.

The slave device responds to BDIN L negation by negating BRPLY L and removing read data from BDAL bus drivers. BRPLY L must be negated 100 ns (maximum) prior to removal of read data. The bus master responds to the negated BRPLY L by negating BSYNC L.

Conditions for the next BSYNC L assertion are as follows.

- BSYNC L must remain negated for 200 ns (minimum).

- BSYNC L must not become asserted within 300 ns of previous BRPLY L negation.

Figure A-2 shows DATI bus cycle timing.

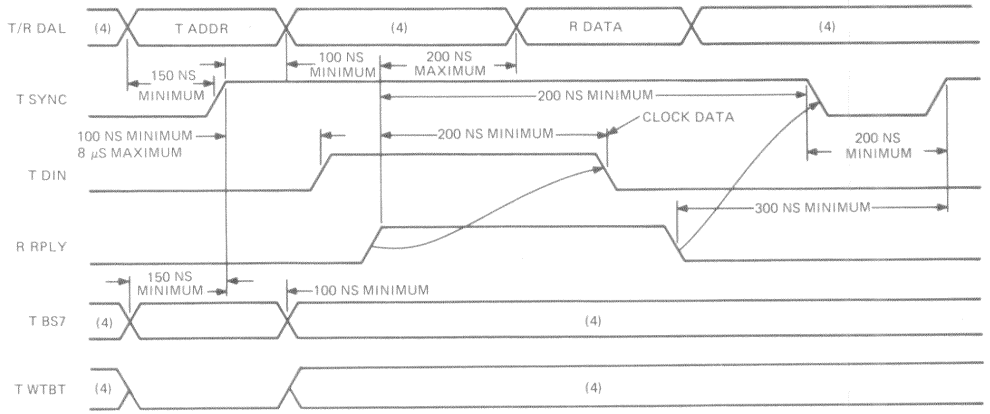
NOTE

Continuous assertion of BSYNC L retains control of the bus by the bus master, and the previously addressed slave device remains selected. This is done for DATIO(B) bus cycles where DATO or DATOB follows a DATI without BSYNC L negation and a second device addressing operation. Also, a slow slave device can hold off data transfers to itself by keeping BRPLY L asserted, which causes the master to keep BSYNC L asserted.

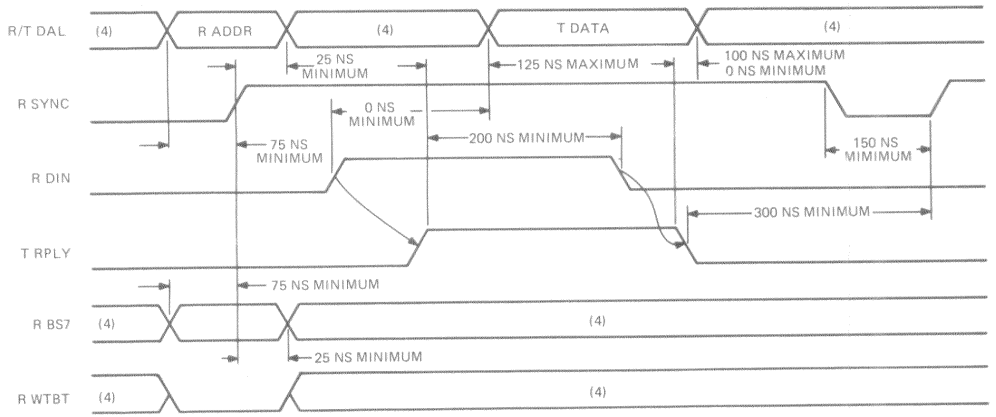
DATO(B) -- DATO(B), shown in Figure A-3, is a write operation. Data is transferred in 32-bit words (DATO) or 8-bit bytes (DATOB) from the bus master to the slave device. The data transfer output can occur after the addressing portion of a bus cycle when BWTBT L has been asserted by the bus master, or immediately following an input transfer part of a DATIO(B) bus cycle.

The data transfer portion of a DATO(B) bus cycle comprises a data setup and deskew time and a data hold and deskew time.

During the data setup and deskew time, the bus master outputs the data on BDAL<15:00> L at least 100 ns after BSYNC L assertion. BWTBT L remains negated for the length of the bus cycle. If the transfer is a byte transfer, BWTBT L remains asserted. If it is the output of a DATIOB, BWTBT L becomes asserted and lasts the duration of the bus cycle.



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES:
1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

MR-6037

Figure A-2 DATI Bus Cycle Timing

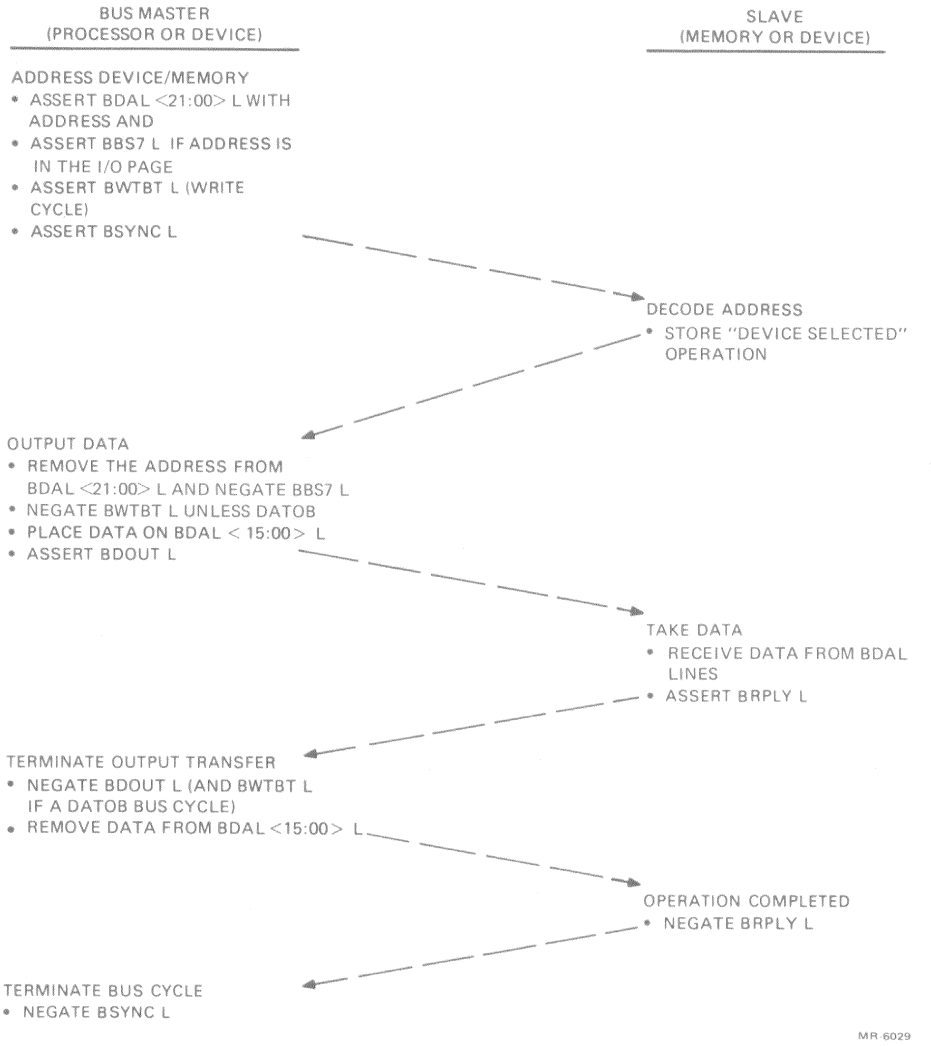


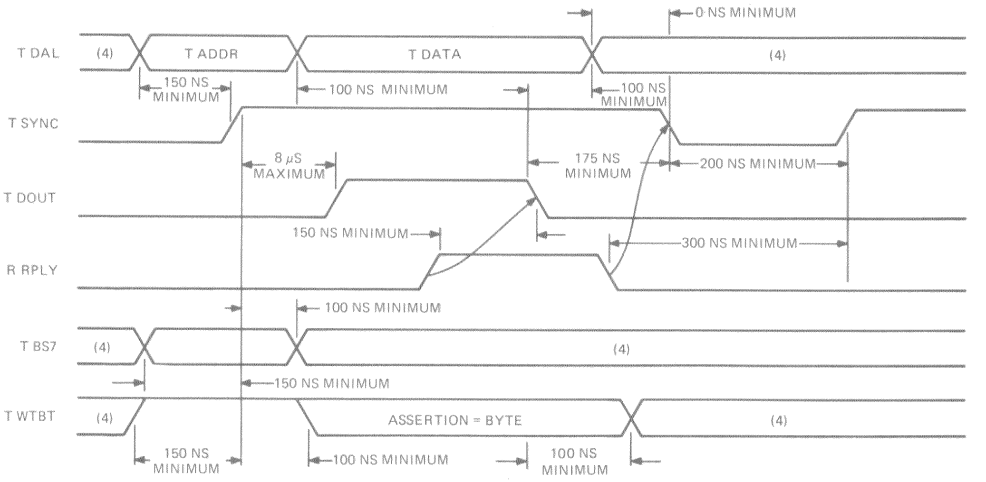
Figure A-3 DATO or DATOB Bus Cycle

During a byte transfer, BDAL<00> L selects the high or low byte. This occurs while in the addressing portion of the cycle. If asserted, the high byte (BDAL<15:08> L) is selected; otherwise, the low byte (BDAL<07:00> L) is selected. An asserted BDAL 16 L at this time forces a parity error to be written into memory if the memory is a parity-type memory. BDAL 17 L is not used for write operations. The bus master asserts BDOUT L at least 100 ns after BDAL and BDWTBT L bus drivers are stable. The slave device responds by asserting BRPLY L within 10 μs to avoid bus timeout. This completes the data setup and deskew time.

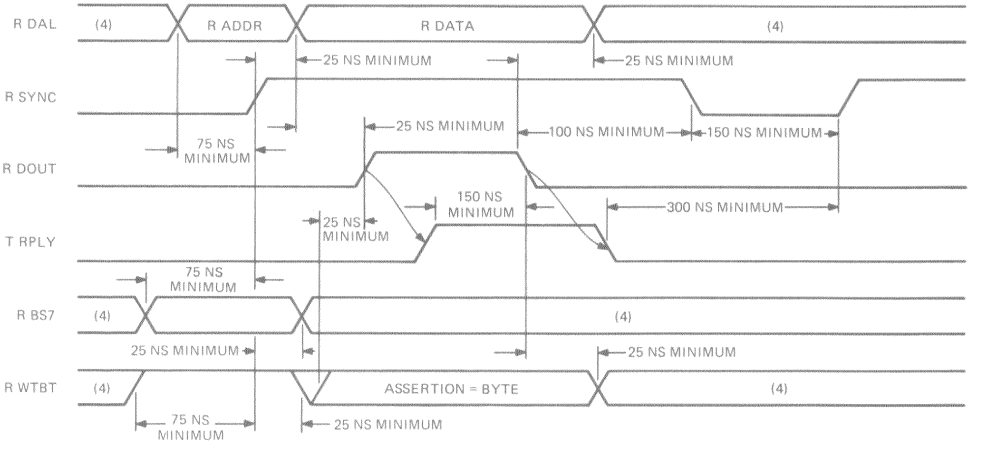
During the data hold and deskew time, the bus master receives BRPLY L and negates BDOUT L, which must remain asserted for at least 150 ns from the receipt of BRPLY L before being negated by the bus master. BDAL<17:00> L bus drivers remain asserted for at least 100 ns after BDOUT L negation. The bus master then negates BDAL inputs.

During this time, the slave device senses BDOUT L negation. The data is accepted and the slave device negates BRPLY L. The bus master responds by negating BSYNC L. However, the processor does not negate BSYNC L for at least 175 ns after negating BDOUT L. This completes the DATO(B) bus cycle. Before the next cycle, BSYNC L must remain unasserted for at least 200 ns. Figure A-4 shows DATO(B) bus cycle timing.

DAITO(B) -- The protocol for a DATIO(B) bus cycle is identical to the addressing and data transfer portions of the DATI and DATO(B) bus cycles, and is shown in Figure A-5. After addressing the device, a DATI cycle is performed as explained earlier; however, BSYNC L is not negated. BSYNC L remains active for an output word or byte transfer [DATO(B)]. The bus master maintains at least 200 ns between BRPLY L negation during the DATI cycle and BDOUT L assertion. The cycle is terminated when the bus master negates BSYNC L, as described for DATO(B). Figure A-6 illustrates DATIO(B) bus cycle timing.



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES:

1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

MR-1179

Figure A-4 DATO or DATOB Bus Cycle Timing

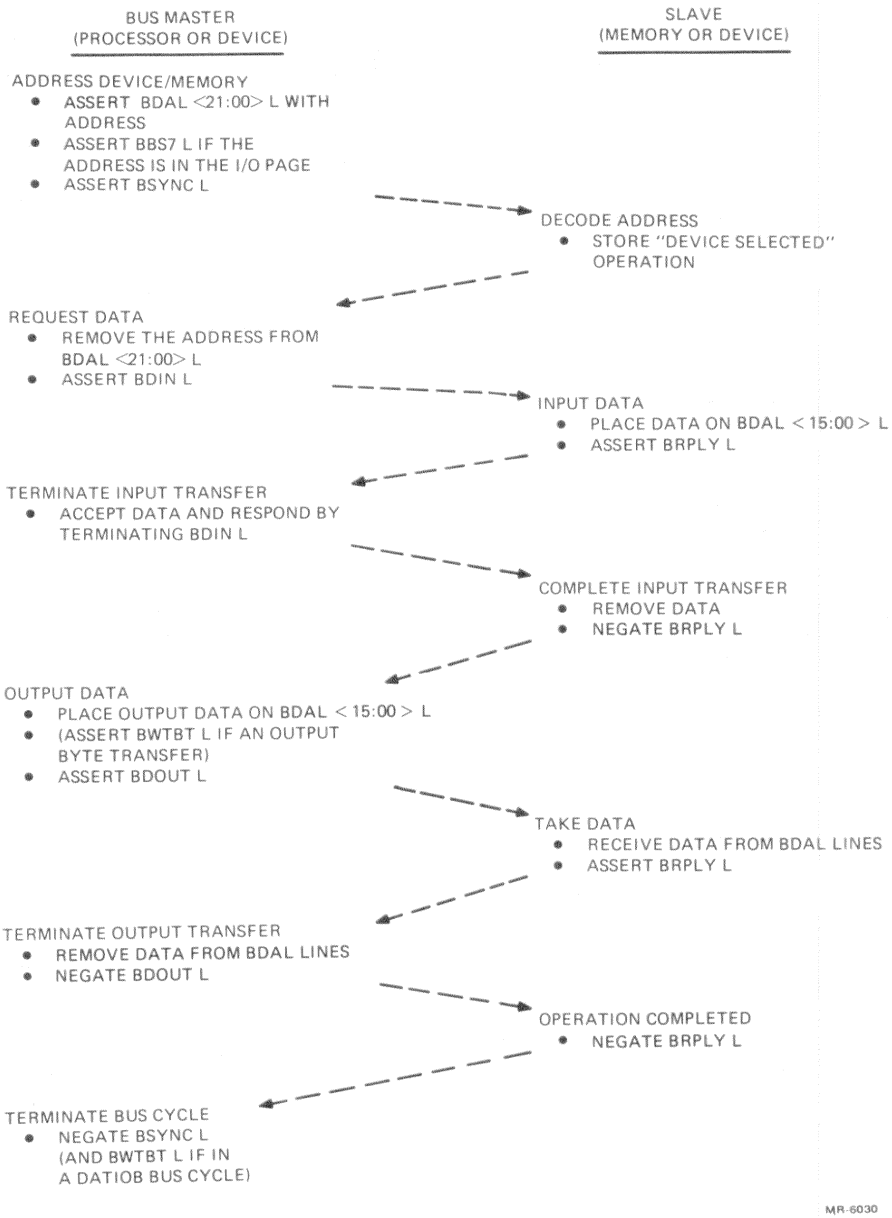
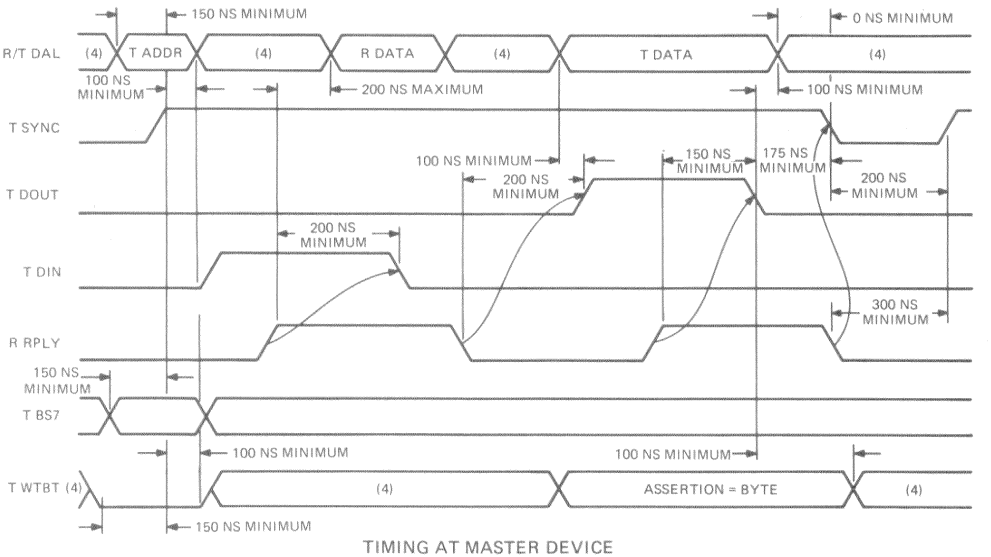
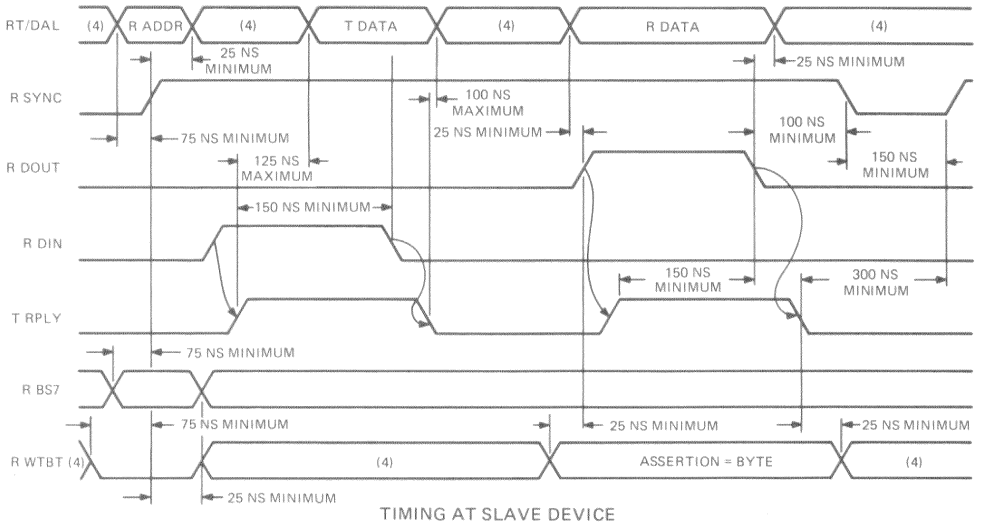


Figure A-5 DATIO or DATIOB Bus Cycle



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES:

1. TIMING SHOWN AT REQUESTING DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

MR-6036

Figure A-6 DATIO or DATIOB Bus Cycle Timing

A.4 DIRECT MEMORY ACCESS

The direct memory access (DMA) capability allows direct data transfer between I/O devices and memory. This is useful when using mass storage devices (for example, disks) that move large blocks of data to and from memory. A DMA device needs to know only the starting address in memory, the starting address in mass storage, the length of the transfer, and whether the operation is read or write. When this information is available, the DMA device can transfer data directly to or from memory. Since most DMA devices must perform data transfers in rapid succession or lose data, DMA devices are provided the highest priority.

DMA is accomplished after the processor (normally bus master) has passed bus mastership to the highest-priority DMA device that is requesting the bus. The processor arbitrates all requests and grants the bus to the DMA device electrically closest to it. A DMA device remains bus master until it relinquishes its mastership. The following control signals are used during bus arbitration.

BDMGI L	DMA grant input
BDMGO L	DMA grant output
BDMR L	DMA request line
BSACK L	Bus grant acknowledge

A.4.1 DMA Protocol

A DMA transaction can be divided into three phases:

1. Bus mastership acquisition phase
2. Data transfer phase
3. Bus mastership relinquishment phase.

During the bus mastership acquisition phase, a DMA device requests the bus by asserting BDMR L. The processor arbitrates the request and initiates the transfer of bus mastership by asserting BDMGO L.

The maximum time between BDMR L assertion and BDMGO L assertion is DMA latency. This time is processor-dependent. BDMGO L/BDMGI L is one signal that is daisy-chained through each module in the backplane. It is driven out of the processor on the BDMGO L pin, enters each module on the BDMGI L pin, and exits on the BDMGO L pin. This signal passes through the modules in descending order of priority until it is stopped by the requesting device. The requesting device blocks the output of BMDGO L and asserts BSACK L. If BDMR L is continuously asserted, the bus hangs.

During the data transfer phase, the DMA device continues asserting BSACK L. The actual data transfer is performed as described earlier.

The DMA device can assert BSYNC L for a data transfer 250 ns (minimum) after it received BDMGI L and its BSYNC L bus receiver becomes negated.

During the bus mastership relinquishment phase, the DMA device gives up the bus by negating BSACK L. This occurs after completing (or aborting) the last data transfer cycle (BRPLY L negated). BSACK L may be negated up to a maximum of 300 ns before negating BSYNC L. Figure A-7 shows the DMA protocol, and Figure A-8 shows DMA request/grant timing.

NOTE

If multiple data transfers are performed during this phase, consideration must be given to the use of the bus for other system functions, such as memory refresh (if required).

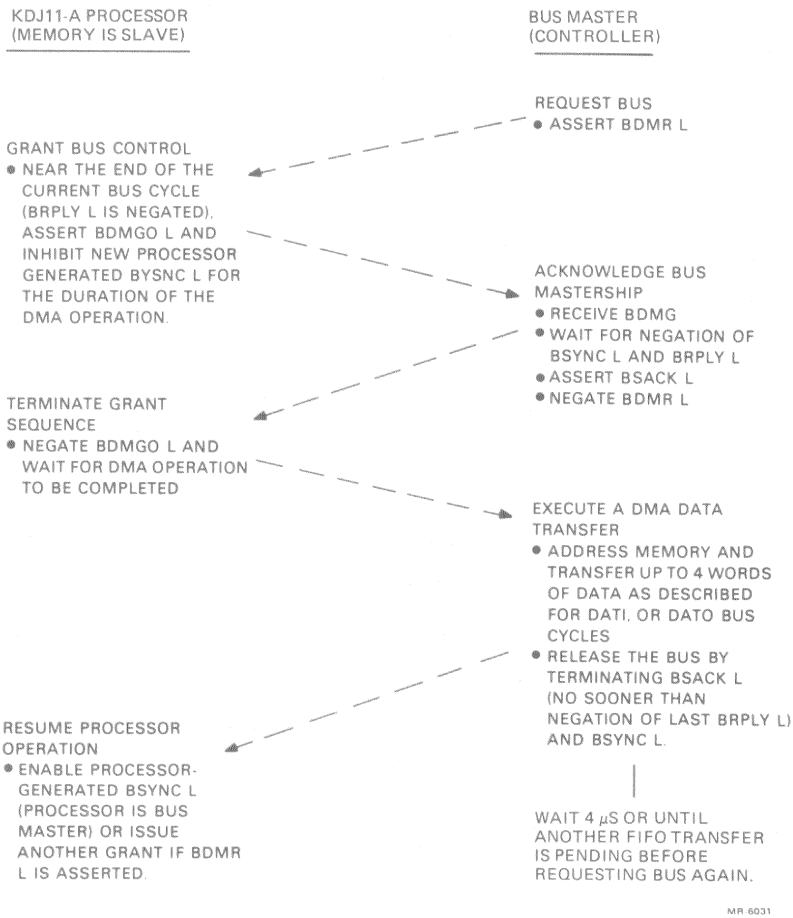


Figure A-7 DMA Protocol

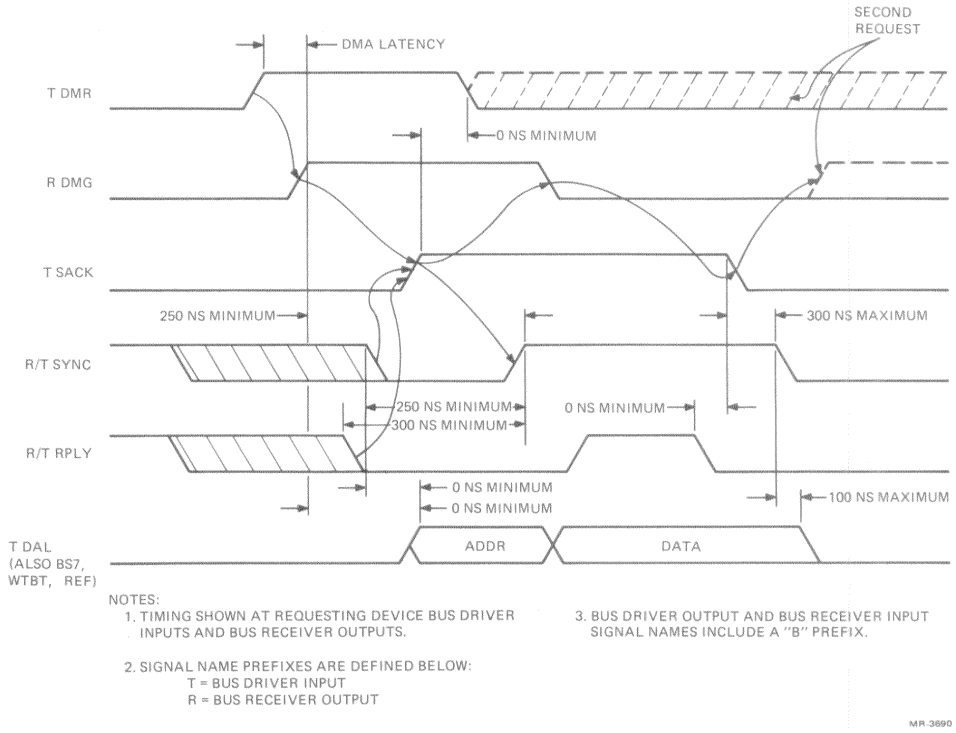


Figure A-8 DMA Request/Grant Timing

A.4.2 Block Mode DMA

For increased throughput, block mode DMA may be implemented on a device for use with memories that support this type of transfer. In a block mode transaction, the starting memory address is asserted, followed by data for that address, and data for consecutive addresses.

By eliminating the assertion of the address for each data word, the transfer rate is almost doubled. The DATBI and DATBO bus cycles are described below.

A.4.2.1 **DATBI** -- The device addressing portion of the cycle is the same as described earlier for other bus cycles. (See Figure A-9.) The bus master gates BDAL<21:00>, BBS7, and the negation of BWTBT onto the bus.

The master asserts the first BDIN 100 ns after BSYNC, and asserts BBS7 a maximum of 50 ns after asserting BDIN for the first time. BBS7 is a request to the slave for a block mode transfer. BBS7 remains asserted until a maximum of 50 ns after the assertion of BDIN for the last time. BBS7 may be gated as soon as the conditions for asserting BDIN are met.

The slave asserts BRPLY a minimum of 0 ns (8 ns maximum to avoid bus timeout) after receiving BDIN. It asserts BREF concurrently with BRPLY if it is a block mode device capable of supporting another BDIN after the current one. The slave gates BDAL<15:00> onto the bus 0 ns (minimum) after the assertion of BDIN, and 125 ns (maximum) after the assertion of BRPLY.

The master receives the stable data from 200 ns (maximum) after the assertion of BRPLY until 20 ns (minimum) after the negation of BDIN. It negates BDIN 200 ns (minimum) after the assertion of BRPLY.

The slave negates BRPLY 0 ns (minimum) after the negation of BDIN. If BBS7 and BREF are both asserted when BRPLY is negated, the slave prepares for another BDIN cycle. BBS7 is stable from 125 ns after BDIN is asserted until 150 ns after BRPLY is negated. The master asserts BDIN 150 ns (minimum) after BRPLY is negated, and the cycle is continued as before. (BBS7 remains asserted and the slave responds to BDIN with BRPLY and BREF.) BREF is stable from 75 ns after BRPLY is asserted until 20 ns (minimum) after BDIN is negated.

If BBS7 and BREF are not both asserted when BRPLY is negated, the slave removes the data from the bus 0 ns (minimum) and 100 ns (maximum) after negating BRPLY. The master negates BSYNC 250 ns (minimum) after the assertion of the last BRPLY, and 0 ns (minimum) after the negation of that BRPLY.

A.4.2.2 **DATBO** -- The device addressing portion of the cycle is the same as shown in Figure A-10. The bus master gates BDAL<21:00>, BBS7, and the assertion of BWTBT onto the bus.

A minimum of 100 ns after BSYNC is asserted, data on BDAL<15:00> and the negated BWTBT are put onto the bus. The master then asserts BDOUT a minimum of 100 ns after gating the data.

The slave receives stable data and BWTBT from 25 ns (minimum) before the assertion of BDOUT to 25 ns (minimum) after the negation of BDOUT. The slave asserts BRPLY 0 ns (minimum) after receiving BDOUT. It also asserts BREF concurrently with BRPLY if it is a block mode device capable of supporting another BDOUT after the current one.

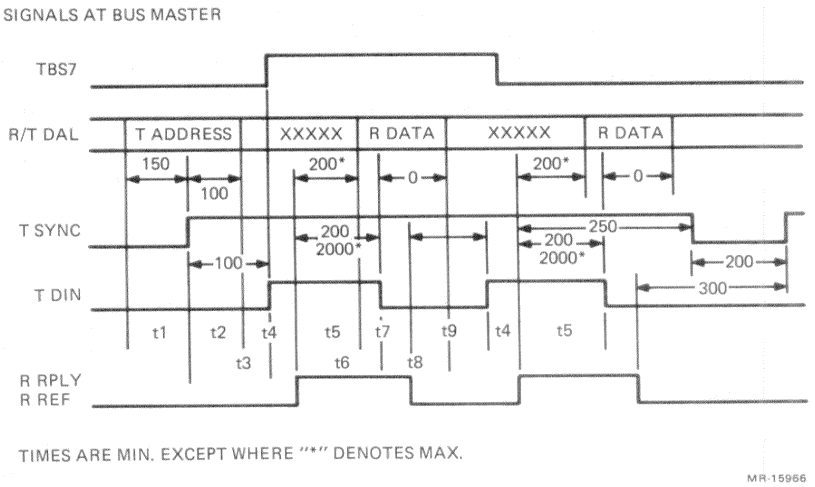


Figure A-9 DATBI Bus Cycle Timing

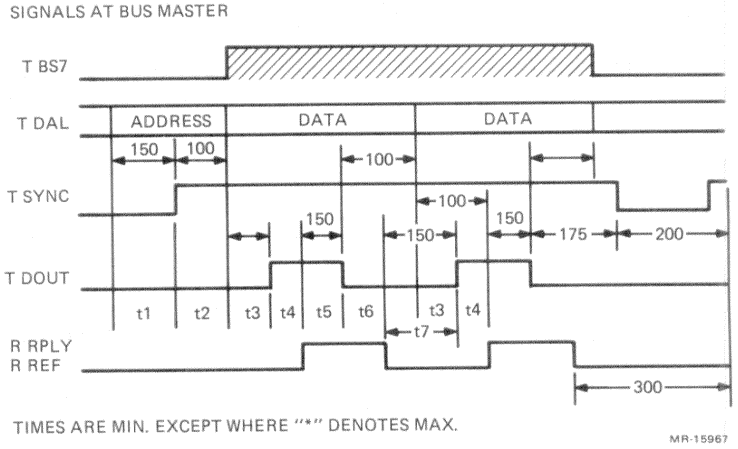


Figure A-10 DATBO Bus Cycle Timing

The master negates BDOUT 150 ns (minimum) after the assertion of BRPLY. If BREF was asserted when BDOUT was negated, and the master wants to transmit more data in this block mode cycle, the new data is gated onto the bus 100 ns (minimum) after BDOUT is negated. BREF is stable from 75 ns (maximum) after BRPLY is asserted until 20 ns (minimum) after BDOUT is negated. The master asserts BDOUT 100 ns (minimum) after gating new data onto the bus and 150 ns minimum after BRPLY negates. The cycle continues as before.

If BREF was not asserted when BDOUT was negated, or if the bus master does not want to transmit more data in this cycle, the master removes data from the bus 100 ns (minimum) after negating BDOUT. The slave negates BRPLY 0 ns (minimum) after negating BDOUT. The bus master negates BSYNC 175 ns (minimum) after negating BDOUT, and 0 ns (minimum) after the negation of BRPLY.

A.4.3 DMA Guidelines

1. Systems with memory refresh over the bus must not include devices that perform more than one transfer per acquisition.

2. Bus masters that do not use block mode are limited to four DATI, four DATO, or two DATIO transfers per acquisition.

3. Block mode bus masters that do not monitor BDMR are limited to eight transfers per acquisition.

4. If BDMR is not asserted after the seventh transfer, block mode bus masters that do monitor BDMR may continue making transfers until the bus slave fails to assert BREF, or until they reach the total maximum of 16 transfers. Otherwise, they stop after eight transfers.

A.5 INTERRUPTS

The interrupt capability of the Q22-Bus allows an I/O device to temporarily suspend (interrupt) current program execution and divert processor operation to service the requesting device. The processor inputs a vector from the device to start the service routine (handler). Like the device register address, hardware fixes the device vector at locations within a designated range below location 001000. The vector indicates the first of a pair of addresses. The processor reads the contents of the first address, the starting address of the interrupt handler. The contents of the second address is a new processor status word (PS).

The new PS can raise the interrupt priority level, thereby preventing lower-level interrupts from breaking into the current interrupt service routine. Control is returned to the interrupted program when the interrupt handler is ended. The original interrupted program's address (PC) and its associated PS are stored on a stack. The original PC and PS are restored by a return from interrupt (RTI or RTT) instruction at the end of the handler. The use of the stack and the Q22-Bus interrupt scheme can allow interrupts to occur within interrupts (nested interrupts), depending on the PS.

Interrupts can be caused by Q22-Bus options or the MicroVAX CPU. Those interrupts that originate from within the processor are called "traps". Traps are caused by programming errors, hardware errors, special instructions, and maintenance features.

The following are Q22-Bus signals used in interrupt transactions.

BIRQ4 L	Interrupt request priority level 4
BIRQ5 L	Interrupt request priority level 5
BIRQ6 L	Interrupt request priority level 6
BIRQ7 L	Interrupt request priority level 7
BIAKI L	Interrupt acknowledge input
BIAKO L	Interrupt acknowledge output
BDAL<21:00>	Data/address lines
BDIN L	Data input strobe
BRPLY L	Reply

A.5.1 Device Priority

The Q22-Bus supports the following two methods of device priority.

1. Distributed Arbitration -- Priority levels are implemented on the hardware. When devices of equal priority level request an interrupt, priority is given to the device electrically closest to the processor.

2. Position-Defined Arbitration -- Priority is determined solely by electrical position on the bus. The closer a device is to the processor, the higher its priority is.

A.5.2 Interrupt Protocol

Interrupt protocol on the Q22-Bus has three phases: the interrupt request phase, interrupt acknowledge and priority arbitration phase, and interrupt vector transfer phase. Figure A-11 shows the interrupt request/acknowledge sequence.

The interrupt request phase begins when a device meets its specific conditions for interrupt requests. For example, the device is ready, done, or an error has occurred. The interrupt enable bit in a device status register must be set. The device then initiates the interrupt by asserting the interrupt request line(s). BIRQ4 L is the lowest hardware priority level and is asserted for all interrupt requests for compatibility with previous Q22 processors. The level at which a device is configured must also be asserted. A special case exists for level 7 devices that must also assert level 6. For an explanation, refer to the discussion below on arbitration involving the 4-level scheme.

Interrupt Level	Lines Asserted by Device
4	BIRQ4 L
5	BIRQ4 L, BIRQ5 L
6	BIRQ4 L, BIRQ6 L
7	BIRQ4 L, BIRQ6 L, BIRQ7 L

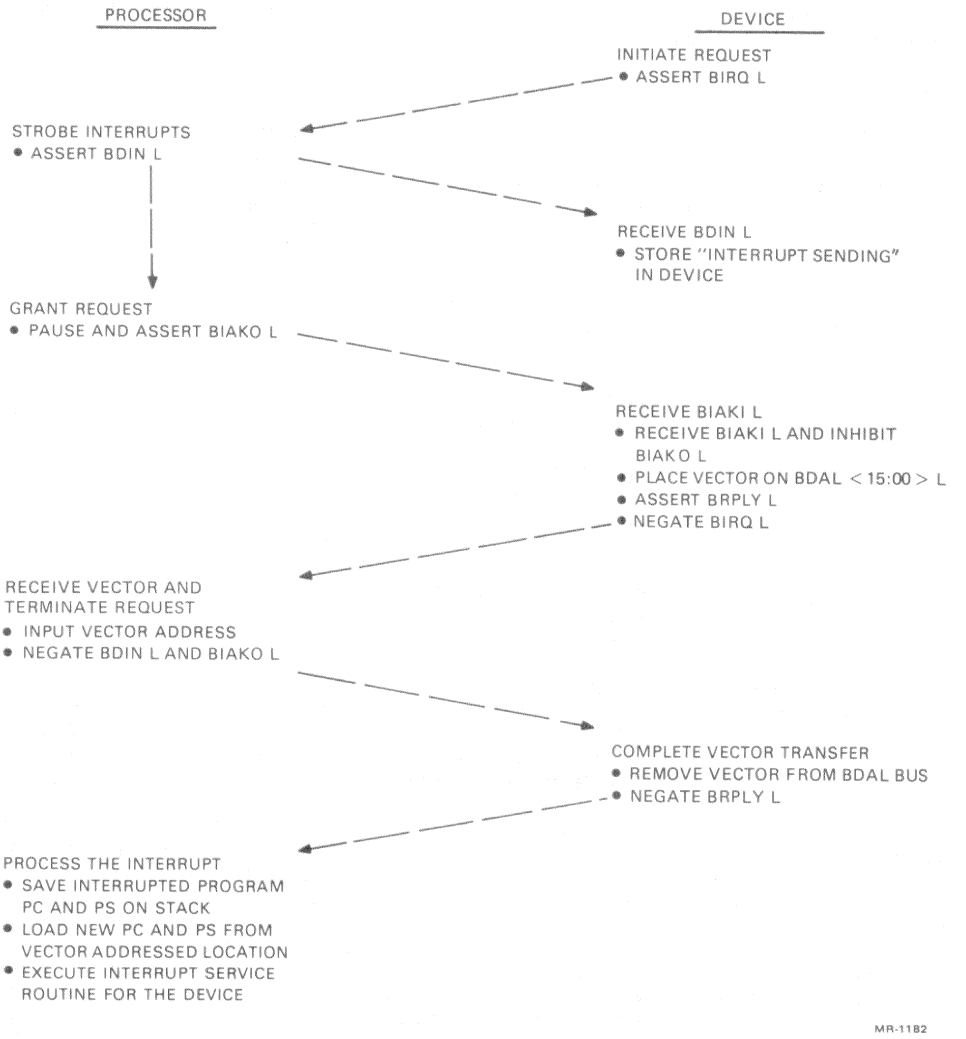


Figure A-11 Interrupt Request/Acknowledge Sequence

The interrupt request line remains asserted until the request is acknowledged.

During the interrupt acknowledge and priority arbitration phase, the LSI-11/23 processor acknowledges interrupts under the following conditions.

1. The device interrupt priority is higher than the current PS<7:5>.

2. The processor has completed instruction execution and no additional bus cycles are pending.

The processor acknowledges the interrupt request by asserting BDIN L, and 150 ns (minimum) later asserting BIAKO L. The device electrically closest to the processor receives the acknowledge on its BIAKI L bus receiver.

At this point, the two types of arbitration must be discussed separately. If the device that receives the acknowledge uses the 4-level interrupt scheme, it reacts as follows.

1. If not requesting an interrupt, the device asserts BIAKO L and the acknowledge propagates to the next device on the bus.

2. If the device is requesting an interrupt, it must check that no higher-level device is currently requesting an interrupt. This is done by monitoring higher-level request lines.The table below lists the lines that need to be monitored by devices at each priority level.

   In addition to asserting levels 7 and 4, level 7 devices must drive level 6. This is done to simplify the monitoring and arbitration by level 4 and 5 devices. In this protocol, level 4 and 5 devices need not monitor level 7 because level 7 devices assert level 6. Level 4 and 5 devices become aware of a level 7 request because they monitor the level 6 request. This protocol has been optimized for level 4, 5, and 6 devices, since level 7 devices are very seldom necessary.

Device Priority Level	Line(s) Monitored
4	BIRQ5, BIRQ6
5	BIRQ6
6	BIRQ7
7	--

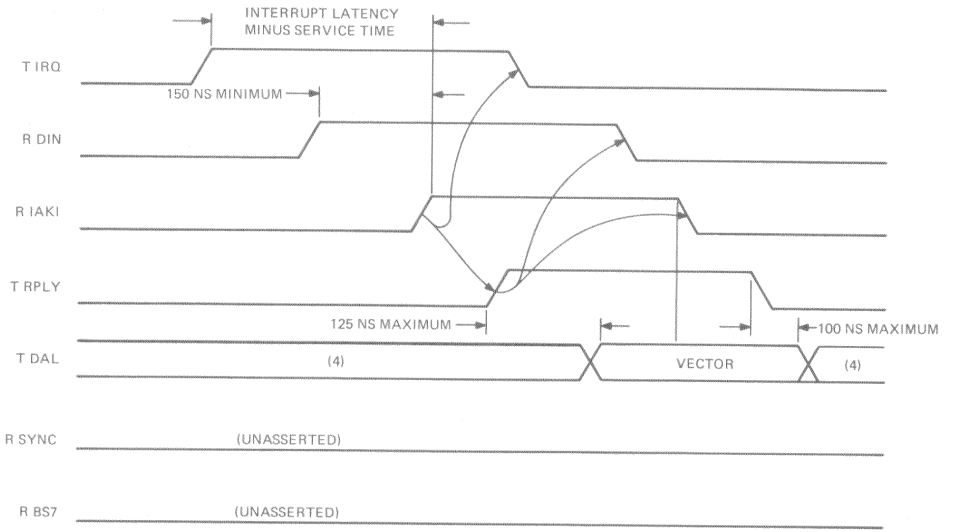
3. If no higher-level device is requesting an interrupt, the acknowledge is blocked by the device. (BIAKO L is not asserted.) Arbitration logic within the device uses the leading edge of BDIN L to clock a flip-flop that blocks BIAKO L. Arbitration is won, and the interrupt vector transfer phase begins.

4. If a higher-level request line is active, the device disqualifies itself and asserts BIAKO L to propagate the acknowledge to the next device along the bus.

Signal timing must be considered carefully when implementing 4-level interrupts. See Figure A-12.

If a single-level interrupt device receives the acknowledge, it reacts as follows.

1. If not requesting an interrupt, the device asserts BIAKO L and the acknowledge propagates to the next device on the bus.



NOTES:
1. TIMING SHOWN AT REQUESTING DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

MR-1183

Figure A-12 Interrupt Protocol Timing

2. If the device was requesting an interrupt, the acknowledge is blocked using the leading edge of BDIN L, and arbitration is won. The interrupt vector transfer phase begins.

The interrupt vector transfer phase is enabled by BDIN L and BIAKI L. The device responds by asserting BRPLY L and its BDAL<15:00> L bus driver inputs with the vector address bits. The BDAL bus driver inputs must be stable within 125 ns (maximum) after BRPLY L is asserted. The processor then inputs the vector address and negates BDIN L and BIAKO L. The device then negates BRPLY L and 100 ns (maximum) later removes the vector address bits. The processor then enters the device's service routine.

NOTE

Propagation delay from BIAKI L to BIAKO L must not be greater than 500 ns per Q22-Bus slot.

The device must assert BRPLY L within 10 μs (maximum) after the processor asserts BIAKI L.

A.5.3 Q22-Bus 4-Level Interrupt Configurations

If you have high-speed peripherals and desire better software performance, you can use the 4-level interrupt scheme. Both position-independent and position-dependent configurations can be used with the 4-level interrupt scheme.

Figure A-13 shows the position-independent configuration. This allows peripheral devices that use the 4-level interrupt scheme to be placed in the backplane in any order. These devices must send out interrupt requests and monitor higher-level request lines as described. The level 4 request is always asserted from a requesting device regardless of priority. If two or more devices of equally high priority request an interrupt, the device physically closest to the processor wins arbitration. Devices that use the single-level interrupt scheme must be modified, or placed at the end of the bus, for arbitration to function properly.

Figure A-14 shows the position-dependent configuration. This configuration is simpler to implement. A constraint is that peripheral devices must be inserted with the highest-priority device located closest to the processor, and the remaining devices placed in the backplane in decreasing order of priority (with the lowest-priority devices farthest from the processor). With this configuration, each device has to assert only its own level and level 4. Monitoring higher-level request lines is unnecessary. Arbitration is achieved through the physical positioning of each device on the bus. Single-level interrupt devices on level 4 should be positioned last on the bus.

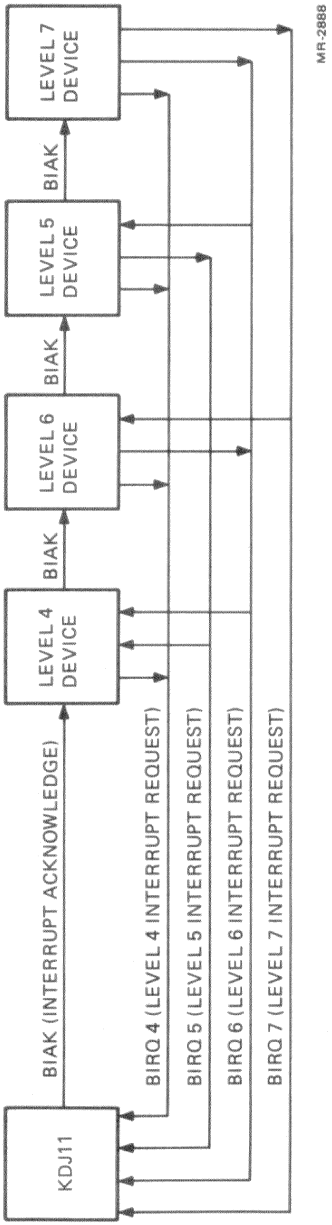


Figure A-13 Position-Independent Configuration

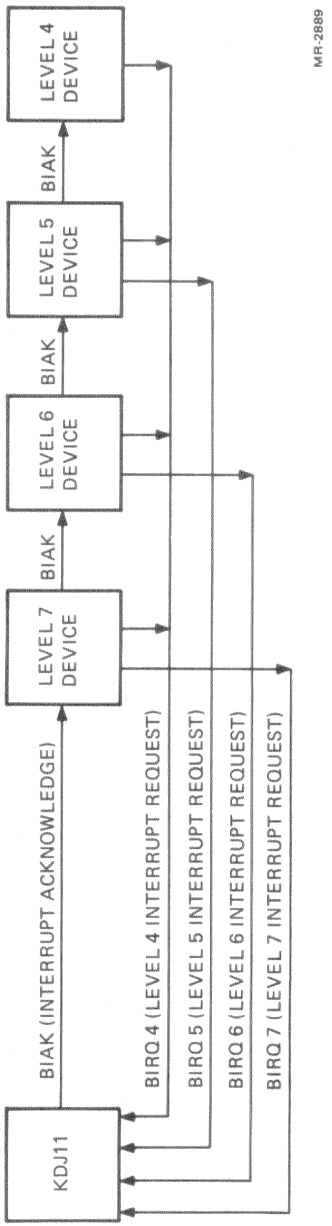


Figure A-14 Position-Dependent Configuration

A.6 CONTROL FUNCTIONS

The following Q22-Bus signals provide control functions.

BREF L	Memory refresh (also block mode DMA)
BHALT L	Processor halt
BINIT L	Initialize
BPOK H	Power OK
BDCOK H	DC power OK

A.6.1 Memory Refresh

If BREF is asserted during the address portion of a bus data transfer cycle, it causes all dynamic MOS memories to be addressed simultaneously. The sequence of addresses required for refreshing the memories is determined by the specific requirements for each memory. The complete memory refresh cycle consists of a series of refresh bus transactions. A new address is used for each transaction. A complete memory refresh cycle must be completed within 1 or 2 ms. Multiple data transfers by DMA devices must be avoided since they could delay memory refresh cycles. This type of refresh is done only for memories that do not perform on-board refresh.

A.6.2 Halt

Assertion of BHALT L for at least 25 ns interrupts the processor, which stops program execution and forces the processor unconditionally into console I/O mode.

A.6.3 Initialization

Devices along the bus are initialized when BINIT L is asserted. The processor can assert BINIT L as a result of executing a reset instruction as part of a power-up or power-down sequence. BINIT L is asserted for approximately 10 μs when reset is executed.

A.6.4 Power Status

Power status protocol is controlled by two signals, BPOK H and BDCOK H. These signals are driven by an external device (usually the power supply).

A.6.5 BDCOK H

When asserted, this indicates that dc power has been stable for at least 3 ms. Once asserted, this line remains asserted until the power fails. It indicates that only 5 μs of dc power reserve remains.

A.6.6 BPOK H

When asserted, this indicates there is at least an 8 ms reserve of dc power, and that BDCOK H has been asserted for at least 70 ms. Once BPOK has been asserted, it must remain asserted for at least 3 ms. The negation of this line, the first event in the power-fail sequence, indicates that power is failing and that only 4 ms of dc power reserve remains.

A.6.7 Power-Up/Down Protocol

Power-up protocol begins when the power supply applies power with BDCOK H negated. This forces the processor to assert BINIT L. When the dc voltages are stable, the power supply or other external device asserts BDCOK H. The processor responds by clearing the PS, floating point status register (FPS), and floating point exception register (FEC). BINIT L is asserted for 12.6 μs, and then negated for 110 μs. The processor continues to test for BPOK H until it is asserted. The power supply asserts BPIK H 70 ms (minimum) after BDCOK H is asserted. The processor then performs its power-up sequence. Normal power must be maintained at least 3.0 ms before a power-down sequence can begin.

A power-down sequence begins when the power supply negates BPOK H. When the current instruction is completed, the processor traps to a power-down routine at location 24. The end of the routine is terminated with a halt instruction to avoid any possible memory corruption as the dc voltages decay.

When the processor executes the halt instruction, it tests the BPOK H signal. If BPOK H is negated, the processor enters the power-up sequence. It clears internal registers, generates BINIT L, and continues to check for the assertion of BPOK H. If it is asserted and dc voltages are still stable, the processor performs the rest of the power-up sequence. Figure A-15 shows power-up/power-down timing.

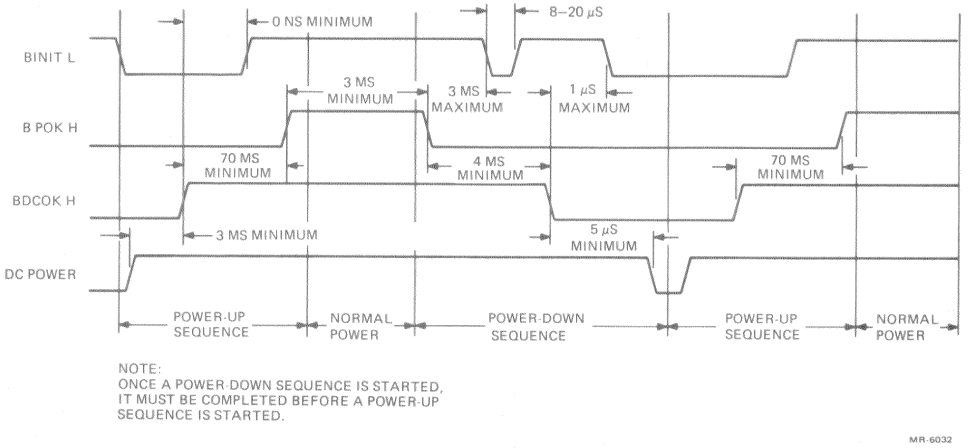


Figure A-15 Power-Up/Power-Down Timing

A.7 Q22-BUS ELECTRICAL CHARACTERISTICS

SIGNAL LEVEL SPECIFICATION

Input Logic Levels:

TTL Logical Low	0.8 Vdc (maximum)
TTL Logical High	2.0 Vdc (minimum)

Output Logic Levels:

TTL Logical Low	0.4 Vdc (maximum)
TTL Logical High	2.4 Vdc (minimum)

A.7.1 Load Definition

AC loads make up the maximum capacitance allowed per signal line to ground. A unit load is defined as 9.35 pF of capacitance. DC loads are defined as maximum current allowed with a signal line driver asserted or unasserted. A unit load is defined as 210 μA in the unasserted state.

A.7.2 120-Ohm Q22-Bus

The electrical conductors interconnecting the bus device slots are treated as transmission lines. A uniform transmission line, terminated in its characteristic impedance, propagates an electrical signal without reflections. Since bus drivers, receivers, and wiring connected to the bus have finite resistance and nonzero reactance, the transmission line impedance is not uniform, and introduces distortions into pulses propagated along it. Passive components of the Q22-Bus (such as wiring, cabling, and etched signal conductors) are designed to have a nominal characteristic impedance of 120 ohms.

The maximum length of interconnecting cable, excluding wiring within the backplane, is limited to 4.88 m (16 ft).

A.7.3 Bus Drivers

Devices driving the 120-ohm Q22-Bus must have open collector outputs and meet the following specifications.

DC SPECIFICATIONS

Output low voltage when sinking 70 mA of current: 0.7 V (maximum).

Output high leakage current when connected to 3.8 Vdc: 25 μA (even if no power is applied, except for BDCOK H and BPOK H).

These conditions must be met at worst-case supply temperature, and input signal levels.

AC SPECIFICATIONS

Bus driver output pin capacitance load: Not to exceed 10 pF.

Propagation delay: Not to exceed 35 ns.

Skew (difference in propagation time between slowest and fastest gate): Not to exceed 25 ns.

Rise/fall times: Transition time (from 10% to 90% for positive transition, 90% to 10% for negative transition) must be no faster than 10 ns.

A.7.4 Bus Receivers

Devices that receive signals from the 120-ohm Q22-Bus must meet the following requirements.

DC SPECIFICATIONS

Input low voltage (maximum): 1.3 V.

Input high voltage (minimum): 1.7 V.

Maximum input current when connected to 3.8 Vdc: 80 μA (even if no power is applied).

These specifications must be met at worst-case supply voltage, temperature, and output signal conditions.

AC SPECIFICATIONS

Bus receiver input pin capacitance load: Not to exceed 10 pF.

Propagation delay: Not to exceed 35 ns.

Skew (difference in propagation time between slowest and fastest gate): Not to exceed 25 ns.

A.7.5 Bus Termination

The 120-ohm Q22-Bus must be terminated at each end by an appropriate terminator, as shown in Figure A-16. This is to be done as a voltage divider with its Thevenin equivalent equal to 120 ohms and 3.4 V (nominal). This type of termination is provided by an REV11-A refresh/boot/terminator, BDV11-AA, KPV11-B, TEV11, or by certain backplanes and expansion cards.

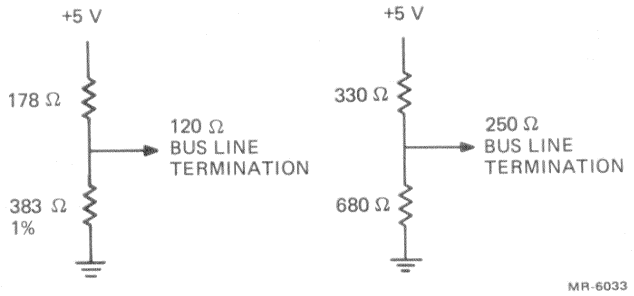


Figure A-16 Bus Line Terminations

Each of the several Q22-Bus lines (all signals whose mnemonics start with the letter B) must see an equivalent network with the following characteristics at each end of the bus.

Input impedance (with respect to ground)	120 ohm +5%, -15%
Open circuit voltage	3.4 Vdc +5%
Capacitance load	Not to exceed 30 pF

NOTE

The resistive termination may be provided by the combination of two modules. (The processor module supplies 220 ohms to ground. This, in parallel with another 220-ohm card, provides 120 ohms.) Both terminators must reside physically within the same backplane.

A.7.6 Bus Interconnecting Wiring

A.7.6.1 Backplane Wiring -- The wiring that connects all device interface slots on the Q22-bus must meet the following specifications.

1. The conductors must be arranged so that each line exhibits a characteristic impedance of 120 ohms (measured with respect to the bus common return).

2. Crosstalk between any two lines must be no greater than 5 percent. Note that worst-case crosstalk is manifested by simultaneously driving all but one signal line and measuring the effect on the undriven line.

3. DC resistance of the signal path, as measured between the near-end terminator and the far-end terminator module (including all intervening connectors, cables, backplane wiring, connector-module etch, etc.) must not exceed 20 ohms.

4. DC resistance of the common return path, as measured between the near-end terminator and the far-end terminator module (including all intervening connectors, cables, backplane wiring, connector-module etch, etc.) must not exceed an equivalent of 2 ohms per signal path. Thus, the composite signal return path dc resistance must not exceed 2 ohms divided by 40 bus lines, or 50 milliohms. Note that although this common return path is nominally at ground potential, the conductance must be part of the bus wiring. The specified low impedance return path must be provided by the bus wiring as distinguished from the common system or power ground path.

A.7.6.2 **Intra-Backplane Bus Wiring** -- The wiring that connects the bus connector slots within one contiguous backplane is part of the overall bus transmission line. Owing to implementation constraints, the nominal characteristic impedance of 120 ohms may not be achievable. Distributed wiring capacitance in excess of the amount required to achieve the nominal 120-ohm impedance may not exceed 60 pF per signal line per backplane.

A.7.6.3 **Power and Ground** -- Each bus interface slot has connector pins assigned for the following dc voltages. The maximum allowable current per pin is 1.5 A. +5 Vdc must be regulated to 5 percent with a maximum ripple of 100 mV pp. +12 Vdc must be regulated to 3 percent with a maximum ripple of 200 mV pp.

- +5 Vdc -- Three pins (4.5 A maximum per bus device slot)
- +12 Vdc -- Two pins (3.0 A maximum per bus device slot)
- Ground -- Eight pins (shared by power return and signal return)

NOTE

Power is not bussed between backplanes on any interconnecting bus cables.

A.8 SYSTEM CONFIGURATIONS

Q22-Bus systems can be divided into two types:

1. Systems containing one backplane
2. Systems containing multiple backplanes

Before configuring any system, three characteristics for each module in the system must be known:

- Power consumption -- +5 Vdc and +12 Vdc current requirements.
- AC bus loading -- The amount of capacitance a module presents to a bus signal line. AC loading is expressed in terms of ac loads, where one ac load equals 9.35 pF of capacitance.
- DC bus loading -- The amount of dc leakage current a module presents to a bus signal when the line is high (undriven). DC loading is expressed in terms of dc loads, where one dc load equals 210 μA (nominal).

Power consumption, ac loading, and dc loading specifications for each module are included in the Microcomputer Interface Handbook.

NOTE

The ac and dc loads and the power consumption of the processor module, terminator module, and backplane must be included in determining the total loading of a backplane.

Rules for configuring single-backplane systems:

1. When using a processor with 220-ohm termination, the bus can accommodate modules that have up to 20 ac loads (total) before additional termination is required. (See Figure A-17.) If more than 20 ac loads are included, the other end of the bus must be terminated with 120 ohms, and then up to 35 ac loads may be present.

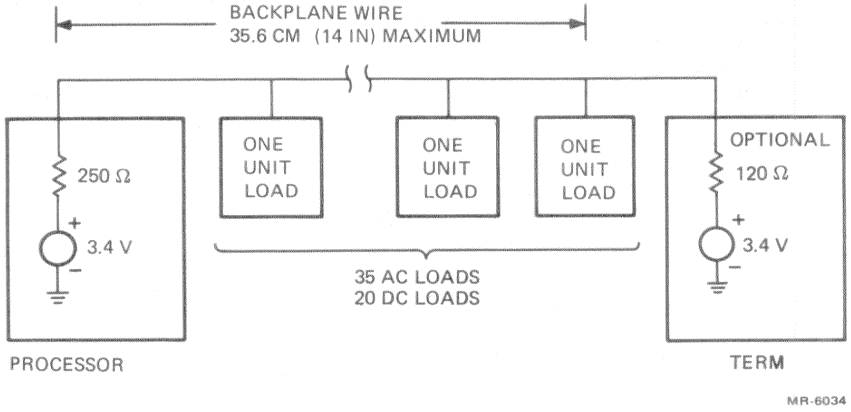


Figure A-17 Single-Backplane Configuration

2. With 120-ohm processor termination, up to 35 ac loads can be used without additional termination. If 120-ohm bus termination is added, up to 45 ac loads can be configured in the backplane.

3. The bus can accommodate modules up to 20 dc loads (total).

4. The bus signal lines on the backplane can be up to 35.6 cm (14 in) long.

Rules for configuring multiple-backplane systems:

1. Figure A-18 shows that up to three backplanes may make up the system.

2. The signal lines on each backplane can be up to 25.4 cm (10 in) long.

3. Each backplane can accommodate modules that have up to 22 ac loads (total). Unused ac loads from one backplane may not be added to another backplane if the second backplane loading will exceed 22 ac loads. It is desirable to load backplanes equally, or with the highest ac loads in the first and second backplanes.

4. DC loading of all modules in all backplanes cannot exceed 20 loads (total).

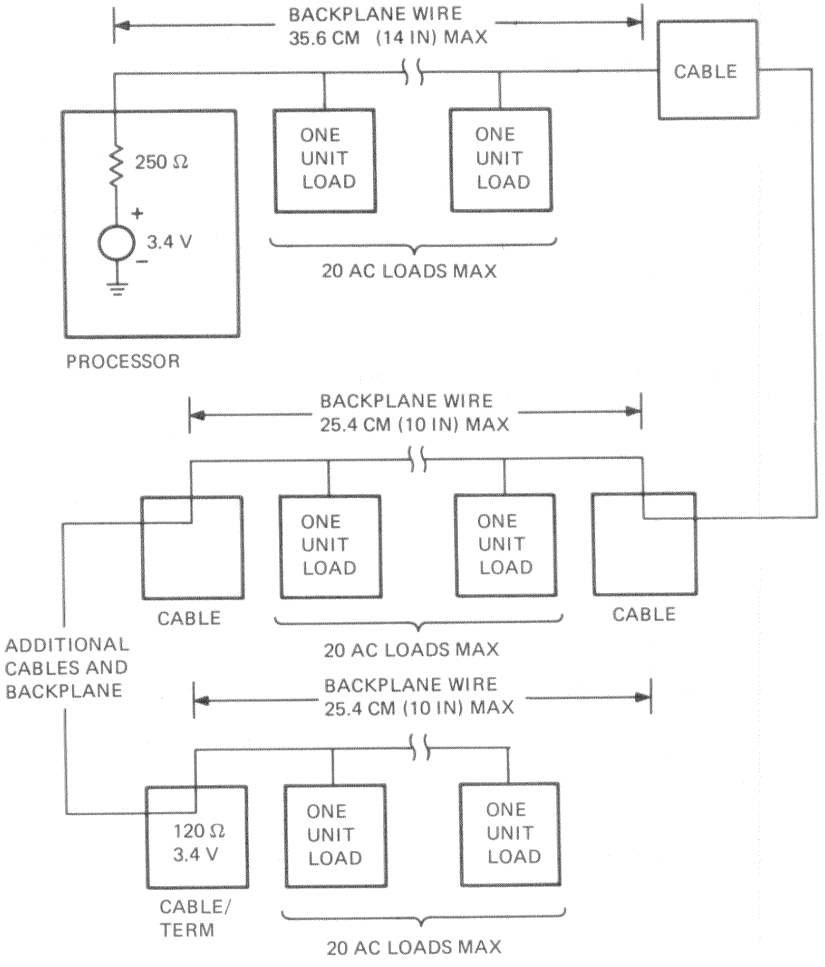
5. Both ends of the bus must be terminated with 120 ohms. This means the first and last backplanes must have an impedance of 120 ohms. To achieve this, each backplane may be lumped together as a single point. The resistive termination may be provided by a combination of two modules in the backplane - the processor providing 220 ohms to ground in parallel with an expansion paddle card providing 250 ohms to give the needed 120-ohm termination. Alternately, a processor with 120-ohm termination would need no additional termination on the paddle card to attain 120 ohms in the first box. The 120-ohm termination in the last box can be provided in two ways: the termination resistors may reside either on the expansion paddle card, or on a bus termination card (such as the BDV11).

6. The cable(s) connecting the first two backplanes is (are) 61 cm (2 ft) or more in length.

7. The cable(s) connecting the second backplane to the third backplane is (are) 122 cm (4 ft) longer or shorter than the cable(s) connecting the first and second backplanes.

8. The combined length of both cables cannot exceed 4.88 m (16 ft).

9. The cables used must have a characteristic impedance of 120 ohms.



NOTES:

1. TWO CABLES (MAX) 4.88 M (16 FT) (MAX) TOTAL LENGTH.
2. 20 DC LOADS TOTAL (MAX).

MR-6035

Figure A-18 Multiple-Backplane Configuration

A.8.1 Power Supply Loading

Total power requirements for each backplane can be determined by obtaining the total power requirements for each module in the backplane. Obtain separate totals for +5 V and +12 V power. Power requirements for each module are specified in the Microcomputer Interfaces Handbook.

When distributing power in multiple-backplane systems, do not attempt to distribute power via the Q22-Bus cables. Provide separate, appropriate power wiring from each power supply to each backplane. Each power supply should be capable of asserting BPOK H and BDCOK H signals according to bus protocol; this is required if automatic power-fail/restart programs are implemented, or if specific peripherals require an orderly power-down halt sequence. The proper use of BPOK H and BDCOK H signals is strongly recommended.

A.9 MODULE CONTACT FINGER IDENTIFICATION

Digital's plug-in modules all use the same contact finger (pin) identification system. A typical pin is shown in Figure A-19.

The Q22-Bus is based on the use of quad-height modules that plug into a 2-slot bus connector. Each slot contains 36 lines (18 lines on both the component side and the solder side of the circuit board).

Slots, row A, and row B include a numeric identifier for the side of the module. The component side is designated side 1, the solder side is designated side 2, as shown in Figure A-20. Letters ranging from A through V (excluding G, I, O, and Q) identify a particular pin on a side of a slot. Table A-4 lists and identifies the bus pins of the quad-height module. A bus pin identifier ending with a 1 is found on the component side of the board, while a bus pin identifier ending with a 2 is found on the solder side of the board.

The positioning notch between the two rows of pins mates with a protrusion on the connector block for correct module positioning.

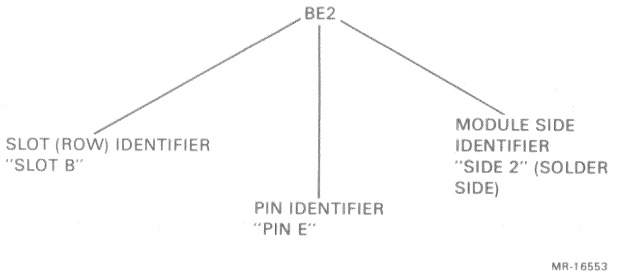


Figure A-19 Typical Pin Identification System

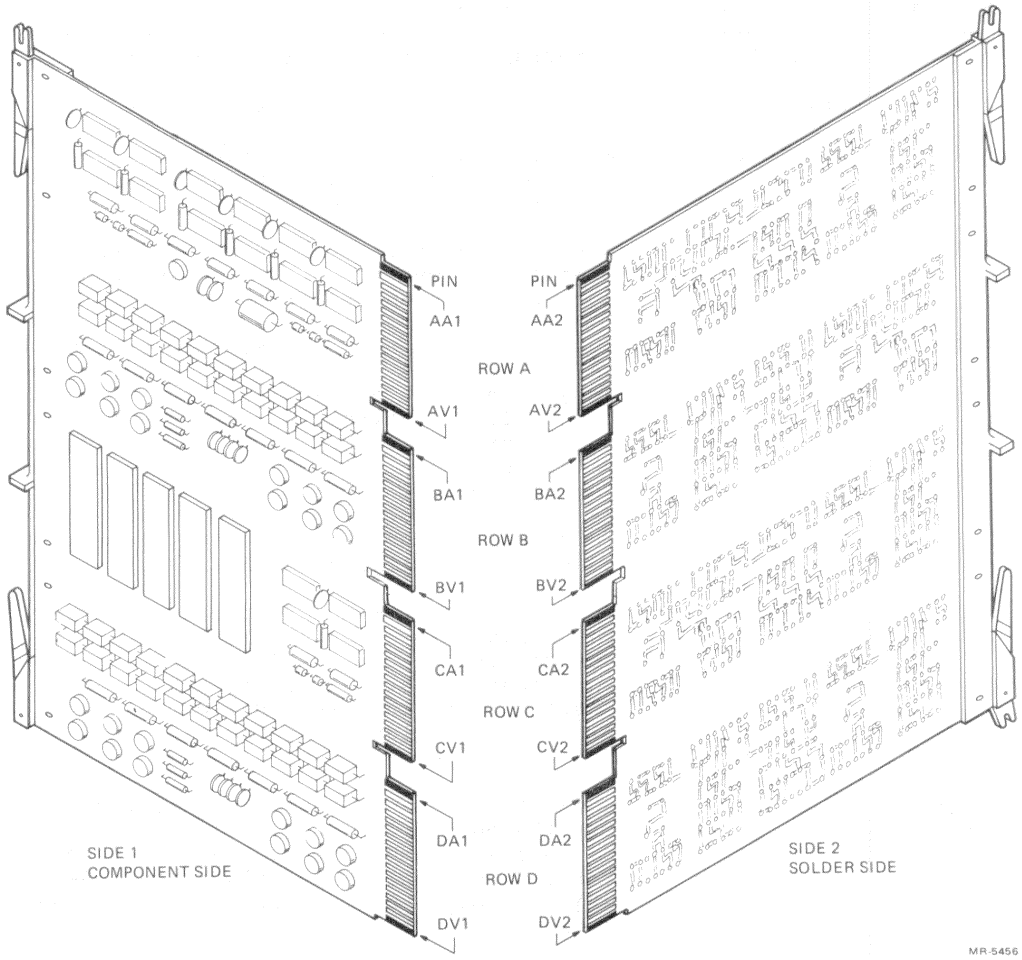


Figure A-20 Quad-Height Module Contact Finger Identification

Table A-4 Bus Pin Identifiers

Bus Pin	Mnemonic(s)	Description
AA1	BIRQ5 L	Interrupt request priority level 5.
AB1	BIRQ6 L	Interrupt request priority level 6.
AC1	BDAL16 L	Extended address bit during addressing protocol; memory error data line during data transfer protocol.
AD1	BDAL17 L	Extended address bit during addressing protocol; memory error logic enable during data transfer protocol.
AE1	SSPARE1 (alternate +5B)	Special Spare -- Not assigned or bussed in Digital's cable or backplane assemblies; available for user connection. Optionally, this pin may be used for +5 V battery (+5 B) backup power to keep critical circuits alive during power failures. A jumper is required on Q22-Bus options to open (disconnect) the +5 B circuit in systems that use this line as SSPARE1.
AF1	SSPARE2	Special Spare -- Not assigned or bussed in Digital's cable or backplane assemblies; available for user interconnection. In the highest-priority device slot, the processor may use this pin for a signal to indicate its RUN state.
AH1	SSPARE3 SRUN	Special Spare -- Not assigned or bussed simultaneously in Digital's cable or backplane assemblies; available for user interconnection. An alternate SRUN signal may be connected in the highest-priority set.
AJ1	GND	Ground -- System signal ground and dc return.
AK1	MSPAREA	Maintenance Spare -- Normally connected together on the backplane at each option location (not a bussed connection).
AL1	MSPAREB	Maintenance Spare -- Normally connected together on the backplane at each option location (not a bussed connection).

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AM1	GND	Ground -- System signal ground and dc return.
AN1	BDMR L	Direct Memory Access (DMA) Request -- A device asserts this signal to request bus mastership. The processor arbitrates bus mastership between itself and all DMA devices on the bus. If the processor is not bus master (it has completed a bus cycle and BSYNC L is not being asserted by the processor), it grants bus mastership to the requesting device by asserting BDMGO L. The device responds by negating BDMR L and asserting BSACK L.
AP1	BHALT L	Processor Halt -- When BHALT L is asserted for at least 25 μs, the processor services the halt interrupt and responds by halting normal program execution. External interrupts are ignored but memory refresh interrupts in Q22 are enabled if W4 on the M7264 and M7264-YA processor modules is removed and DMA request/grant sequences are enabled. The processor executes the ODT microcode, and the console device operation is invoked.
AR1	BREF L	Memory Refresh -- Asserted by a DMA device. This signal forces all dynamic MOS memory units requiring bus refresh signals to be activated for each BSYNC L/BDIN L bus transaction. It is also used as a control signal for block mode DMA. **CAUTION:** The user must avoid multiple DMA data transfers (burst or "hot" mode) that could delay refresh operation if using DMA refresh. Complete refresh cycles must occur once every 1.6 ms if required.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AS1	+12 B or +5 B	+12 Vdc or +5 V battery backup power to keep critical circuits alive during power failures. This signal is not bussed to BS1 in all of Digital's backplanes. A jumper is required on all Q22-Bus options to open (disconnect) the backup circuit from the bus in systems that use this line at the alternate voltage.
AT1	GND	Ground -- System signal ground and dc return.
AU1	PSPARE 1	Spare -- Not assigned; customer usage not recommended. Prevents damage when modules are inserted upside down.
AV1	+5 B	+5 V Battery Power -- Secondary +5 V power connection. Battery power can be used with certain devices.
BA1	BDCOK H	DC Power OK -- A power supply-generated signal that is asserted when the available dc voltage is sufficient to sustain reliable system operation.
BB1	BPOK H	Power OK -- Asserted by the power supply 70 ms after BDCOK is negated when ac power drops below the value required to sustain power (approximately 75% of nominal). When negated during processor operation, a power-fail trap sequence is initiated.
BC1	SSPARE4 BDAL18 L (22-bit only)	Special Spare in the Q22-Bus -- Not assigned. Bussed in 22-bit cable and backplane assemblies; available for user interconnection.
BD1	SSPARE5 BDAL19 L (22-bit only)	**CAUTION:** These pins may be used by manufacturing as test points in some options.
BE1	SSPARE6 BDAL20 L	In the Q22-Bus, these bussed address lines are address lines <21:18>; currently not used during data time.
BF1	SSPARE7 BDAL21 L	In the Q22-Bus, these bussed address lines are address lines <21:18>; currently not used during data time.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
BH1	SSPARE8	Special Spare -- Not assigned or bussed in Digital's cable and backplane assemblies; available for user interconnection.
BJ1	GND	Ground -- System signal ground and dc return.
BK1 BL1	MSPAREB MSPAREB	Maintenance Spare -- Normally connected together on the backplane at each option location (not a bussed connection).
BM1	GND	Ground -- System signal ground and dc return.
BN1	BSACK L	This signal is asserted by a DMA device in response to the processor's BDMGO L signal, indicating that the DMA device is bus master.
BP1	BIRQ7 L	Interrupt request priority level 7.
BR1	BEVNT L	External Event Interrupt Request -- When asserted, the processor responds by entering a service routine via vector address 100$_8$. A typical use of this signal is as a line-time clock interrupt.
BS1	+12 B	+12 Vdc battery backup power (not bussed to AS1 in all of Digital's backplanes).
BT1	GND	Ground -- System signal ground and dc return.
BU1	PSPARE2	Power Spare 2 -- Not assigned a function; not recommended for use. If a module is using -12 V (on pin AB2), and if the module is accidentally inserted upside down in the backplane, -12 Vdc appears on pin BU1.
BV1	+5	+5 V Power -- Normal +5 Vdc system power.
AA2	+5	+5 V Power -- Normal +5 Vdc system power.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AB2	-12	-12 V Power -- -12 Vdc power for (optional) devices requiring this voltage. NOTE: Each Q22-Bus module that requires negative voltages contains an inverter circuit that generates the required voltage(s). Therefore, -12 V power is not required with Digital's options.
AC2	GND	Ground -- System signal ground and dc return.
AD2	+12	+12 V Power -- +12 Vdc system power.
AE2	BDOUT L	Data Output -- When asserted, BDOUT implies that valid data is available on BDAL<0:15> L and that an output transfer, with respect to the bus master device, is taking place. BDOUT L is deskewed with respect to data on the bus. The slave device responding to the BDOUT L signal must assert BRPLY L to complete the transfer.
AF2	BRPLY L	Reply -- BRPLY L is asserted in response to BDIN L or BDOUT L and during IAK transactions. It is generated by a slave device to indicate that it has placed its data on the BDAL bus or that it has accepted output data from the bus.
AH2	BDIN L	Data Input -- BDIN L is used for two types of bus operations: When asserted during BSYNC L time, BDIN L implies an input transfer with respect to the current bus master, and requires a response (BRPLY L). BDIN L is asserted when the master device is ready to accept data from a slave device. When asserted without BSYNC L, it indicates that an interrupt operation is occurring. The master device must deskew input data from BRPLY L.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AJ2	BSYNC L	Synchronize -- BSYNC L is asserted by the bus master device to indicate that it has placed an address on BDAL<0:17> L. The transfer is in process until BSYNC L is negated.
AK2	BWTBT L	Write/Byte -- BWTBT L is used in two ways to control a bus cycle: It is asserted at the leading edge of BSYNC L to indicate that an output sequence (DATO or DATOB), rather than an input sequence, is to follow. It is asserted during BDOUT L, in a DATOB bus cycle, for byte addressing.
AL2	BIRQ4 L	Interrupt Request Priority Level 4 -- A level 4 device asserts this signal when its interrupt enable and interrupt request flips-flops are set. If the PS word bit 7 is 0, the processor responds by acknowledging the request by asserting BDIN L and BIAKO L.
AM2 AN2	BIAKI L BIAKO L	Interrupt Acknowledge -- In accordance with interrupt protocol, the processor asserts BIAKO L to acknowledge receipt of an interrupt. The bus transmits this to BIAKI L of the device electrically closest to the processor. This device accepts the interrupt acknowledge under two conditions: 1.) the device requested the bus by asserting BIRQXL, and 2.) the device has the highest-priority interrupt request on the bus at that time. If these conditions are not met, the device asserts BIAKO L to the next device on the bus. This process continues in a daisy-chain fashion until the device with the highest-interrupt priority receives the interrupt acknowledge signal.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
AP2	BBS7 L	Bank 7 Select -- The bus master asserts this signal to reference the I/O page (including that portion of the I/O page reserved for nonexistent memory). The address in BDAL<0:12> L when BBS7 L is asserted is the address within the I/O page.
AR2 AS2	BDMGI L BDMGO L	Direct Memory Access Grant -- The bus arbitrator asserts this signal to grant bus mastership to a requesting device, according to bus mastership protocol. The signal is passed in a daisy-chain from the arbitrator (as BDMGO L) through the bus to BDMGI L of the next priority device (the device electrically closest on the bus). This device accepts the grant only if it requested to be bus master (by a BDMR L). If not, the device passes the grant (asserts BDMGO L) to the next device on the bus. This process continues until the requesting device acknowledges the grant. **CAUTION:** DMA device transfers must not interfere with the memory refresh cycle.
AT2	BINIT L	Initialize -- This signal is used for system reset. All devices on the bus are to return to a known, initial state; that is, registers are reset to zero, and logic is reset to state 0. Exceptions should be completely documented in programming and engineering specifications for the device.
AU2 AV2	BDAL0 L BDAL1 L	Data/Address lines -- These two lines are part of the 16-line data/address bus over which address and data information are communicated. Address information is first placed on the bus by the bus master device. The same device then either receives input data from, or outputs data to, the addressed slave device or memory over the same bus lines.

Table A-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic(s)	Description
BA2	+5	+5 V Power -- Normal +5 Vdc system power.
BB2	-12	-12 V Power (voltage normally not supplied) -- -12 Vdc power for (optional) devices requiring this voltage.
BC2	GND	Ground -- System signal ground and dc return.
BD2	+12	+12 V Power -- +12 V system power.
BE2	BDAL2 L	Data/Address Lines -- These 14 lines are part of the 16-line data/address bus.
BF2	BDAL3 L	
BH2	BDAL4 L	
BJ2	BDAL5 L	
BK2	BDAL6 L	
BL2	BDAL7 L	
BM2	BDAL8 L	
BN2	BDAL9 L	
BP2	BDAL10 L	
BR2	BDAL11 L	
BS2	BDAL12 L	
BT2	BDAL13 L	
BU2	BDAL14 L	
BV2	BDAL15 L	

B.1 GENERAL DESCRIPTION

The LK201 keyboard has 105 concave-surfaced keys that are divided into four groups: typing keys, numeric keypad for data entry, screen/cursor control keys for editing, and special programmable command/function keys supported by an enclosed printed circuit board. Four LEDs indicate when a specific function is in operation. A 1.9 m (6 ft) coiled cable connects the keyboard to a dedicated 4-pin MICRO-DIN connector on the monitor's rear panel. Figure B-1 shows the LK201 keyboard.

The keyboard is the user interface to the system. It detects keystrokes, encodes them, and transmits the information to the central processor. The keyboard also receives information from the central processor.

Communication between the keyboard and the central processor in the BA23 or BA123 enclosure is full-duplex, serial asynchronous, at a speed of 4800 baud. The communication lines conform to EIA Standard RS-423, which applies to unbalanced voltage interfaces.

B.2 PHYSICAL DESCRIPTION

The keyboard used with the MicroVAX workstation has 105 keys arranged in the following four groups. (See Figure B-1.)

- Main keypad (57 keys)
- Numeric keypad (18 keys)
- Special function keypad (20 keys)
- Editing keypad (10 keys)

The keycaps can be installed manually, but require a special tool for removal.

The keyboard circuitry is contained in a low-profile cabinet with a nominal height of 30 mm (1.2 in) from table top to home row. The keyboard case is made of two plastic shells that can be separated with a screwdriver. Nonslip plastic strips along the bottom prevent the keyboard from sliding on a table top. Two feet can be manually inserted into holes to raise the back edge of the keyboard.

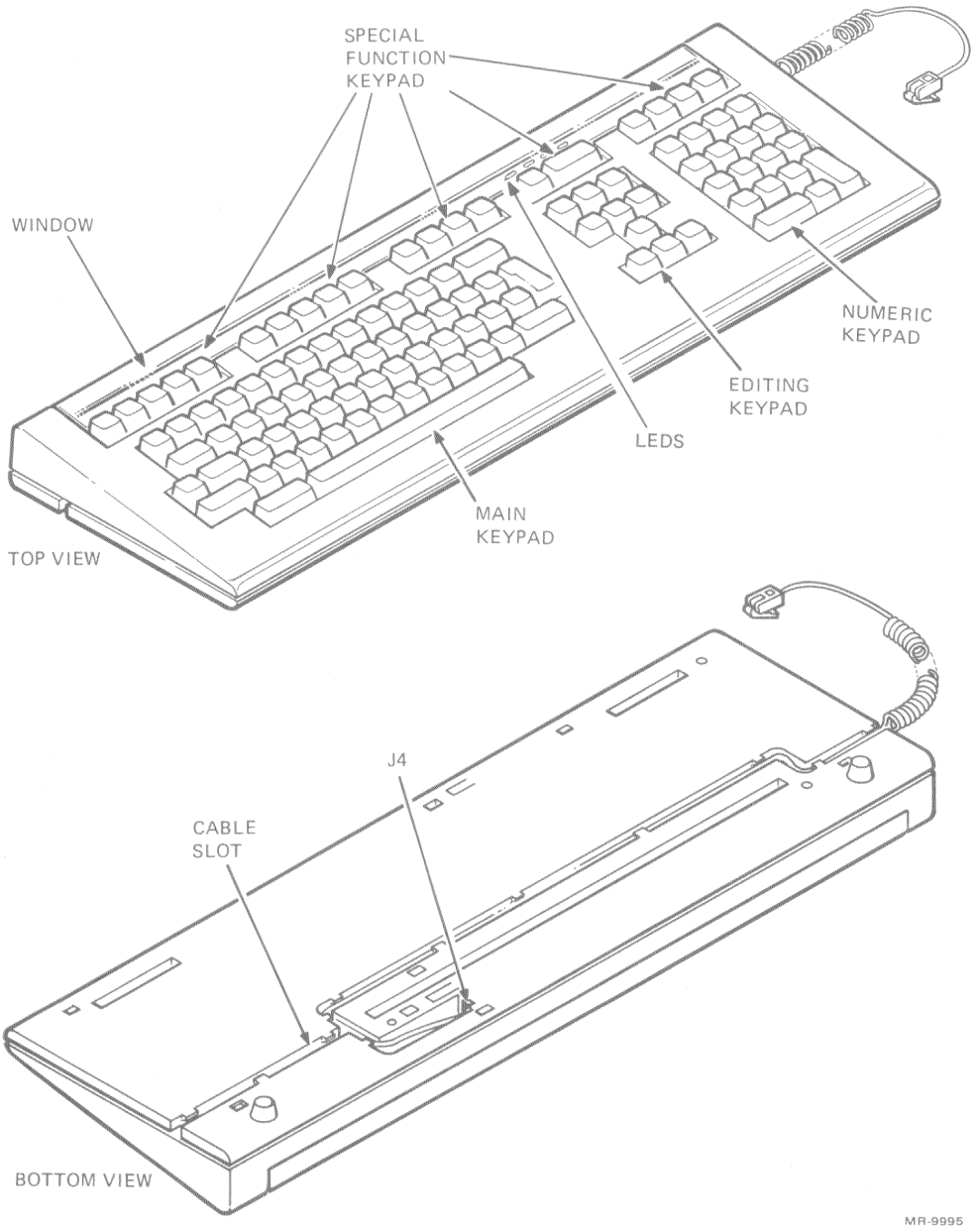


Figure B-1 LK201 Keyboard

A plastic window along the top edge above the special function keys can be lifted to insert a keyboard label strip. The label, a thin paper strip, fits into the indented space and varies according to the application program.

Four LEDs are located beneath the plastic window: hold screen, lock, compose, and wait.

A coiled cable (PN BCC01), with a 4-pin modular connector on each end, connects the keyboard to the video monitor. The keyboard transmits four types of signals to the monitor that pass unchanged via the video cable to the system enclosure as shown in Figure B-2. The four signals are

- +12 V power to keyboard
- Ground to keyboard
- Serial Out (transmit line from keyboard)
- Serial In (receive line to keyboard).

The cable can be placed into a channel in the bottom case, and the modular-type telephone connector fits into the jack, J4. The cable can be inserted into the channel at either side of the keyboard.

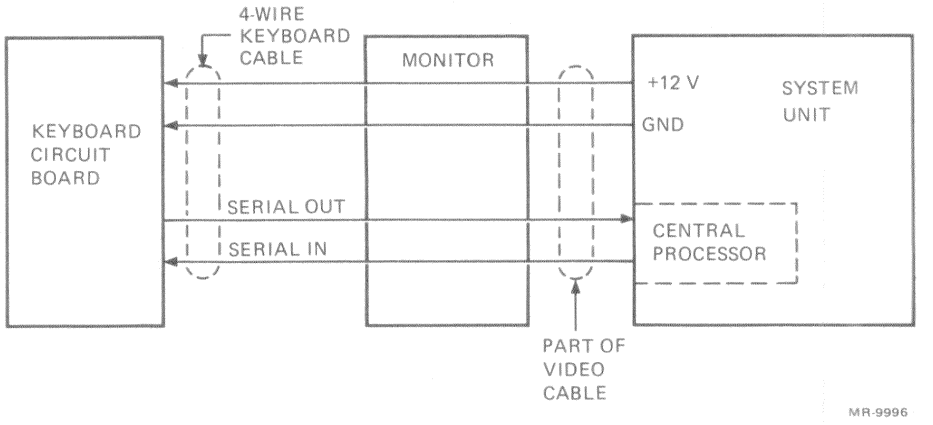


Figure B-2 Keyboard Cable Connections

B.3 BLOCK DIAGRAM DESCRIPTION

Figure B-3 is a simplified block diagram of the keyboard circuitry. All diagram blocks except the block marked "keyboard matrix" are on the printed circuit board. This block represents the connections between the keyboard switches and the signals from the 8051 microprocessor.

The firmware in the 8051 8-bit microprocessor controls the following three major keyboard operations at the same time.

1. Scans the keyboard to detect changes in the keyboard matrix.

2. Transmits the results of the keyboard scan to the system's central processor.

3. Receives information from the system's central processor.

B.3.1 Keyboard Scanning

The keyboard switches are connected at the intersections of an 18 X 8 line matrix. This provides a fixed-position identifier for each key.

The firmware scans the 18-line axis and detects a depressed or newly released key by reading the 8-line axis. The firmware then verifies the detected keystroke and changes this positional information into an 8-bit code that is unique to that key.

B.3.2 Control of Audio Transducer and Indicators

Two circuits control the audio transducer and the indicators. One circuit receives its inputs from the 8051 microprocessor and controls the transducer (beeper). A long beep represents the bell, a short beep represents the keyclick.

A separate circuit, controlled by a signal from the 8051 microprocessor, controls each of the four indicators. The firmware, responding to commands received from the system's central processor, turns the indicators on or off.

B.3.3 Keyboard Firmware Functions

This paragraph describes the keyboard firmware functions. The functions are divided into two categories: those that cannot be changed by instructions from the system's central processor, and those that can.

B.3.3.1 Functions Not Changed by the System's Central Processor Instructions -- The following functions cannot be changed by instructions from the system's central processor.

- Power-up test
- Keycodes
- Special codes

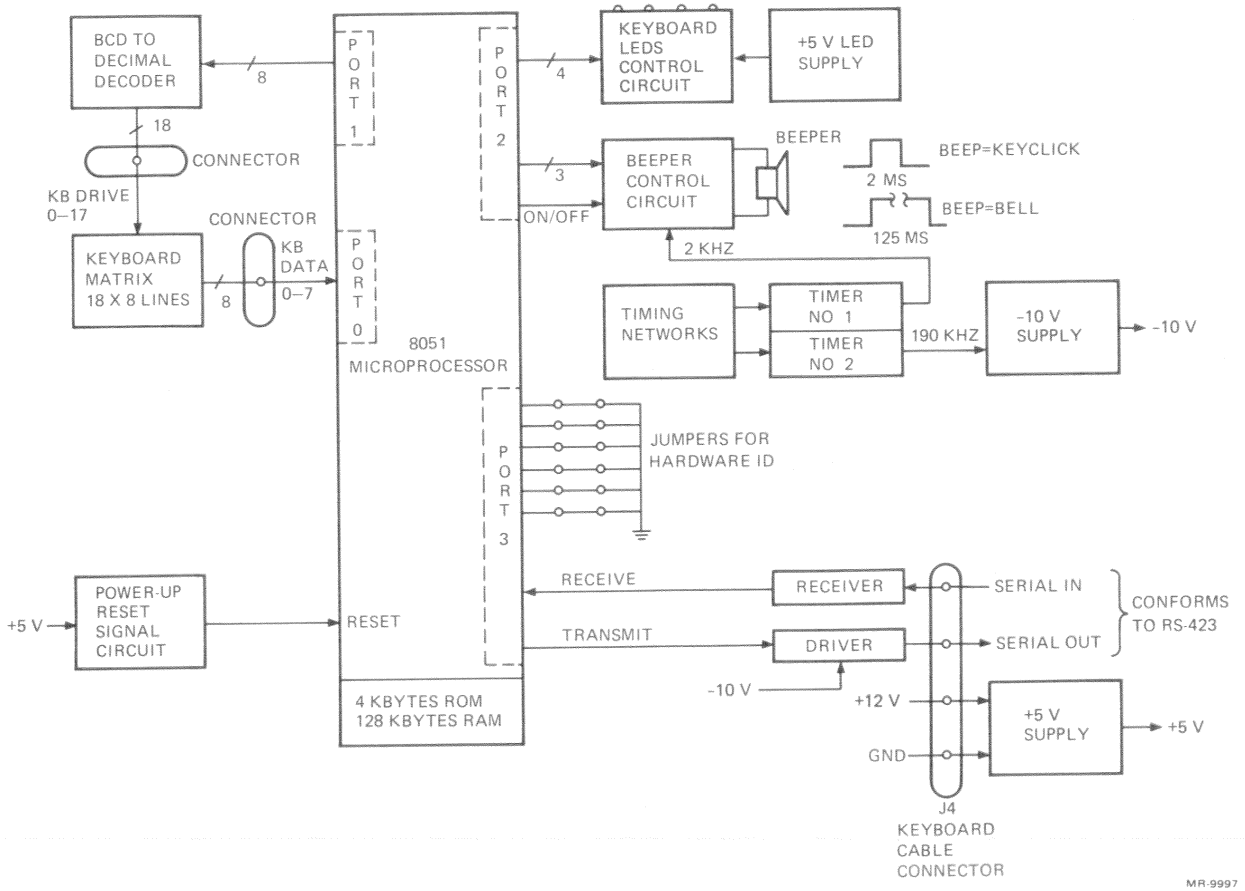


Figure B-3 LK201 Keyboard Circuitry, Simplified Block Diagram

Power-Up Test
Upon power-up, the firmware performs a self-test in less than 70 ms. The test results are transmitted to the system's central processor in four bytes.

The keyboard indicators are lit during the self-test. The indicators blink once during the self-test routine. The indicators go off if the test is passed, but remain lit if the test fails. The system module can also request self-test at any time.

Keycodes
The keycodes represent fixed positions in the key switch matrix. The key associated with a particular matrix position is always represented by the same keycode.

Special Codes
Thirteen special codes are transmitted by the keyboard. Four codes transmit the results of the power-up self-test. The other nine codes are status indicators or command acknowledgements.

B.3.3.2 **Functions Changed by the System's Central Processor Instructions** -- The system's central processor can issue instructions to change some keyboard transmission characteristics and to control the keyboard indicators and beeper.

Upon completion of a successful power-up self-test, the firmware sets certain functions to predetermined conditions. These are referred to as "default" conditions. The conditions can be changed, but they always come up to the default condition after a successful power-up self-test.

B.3.3.3 **Firmware Functions That Can Be Changed** -- Certain firmware functions can be changed by commands (instructions) from the system's central processor. These commands are categorized as "transmission" commands and "peripheral" commands. Transmission commands include a mode set command and an auto-repeat rate set command. Peripheral commands include a variety of commands.

B.4 DETAILED KEYBOARD CIRCUIT DESCRIPTION

The following section describes the keyboard circuitry shown in Figure B-3.

B.4.1 Keyboard Matrix Scanning

The key locations are arranged in an 18 X 8 line matrix. Each key switch is connected across a matrix intersection. This gives a fixed position for each key connected in the matrix. This matrix accommodates all 105 keys in the LK201 keyboard.

Figure B-4 is a simplified block diagram of the matrix scanning circuit. Eight lines from port 1 of the 8051 microprocessor go to the binary-coded decimal (BCD) inputs of two 74LS145 BCD-to-decimal decoders. Ten outputs from one decoder and eight outputs from the other decoder provide the drive lines for the matrix. These 18 lines are called KB DRIVE 0-17.

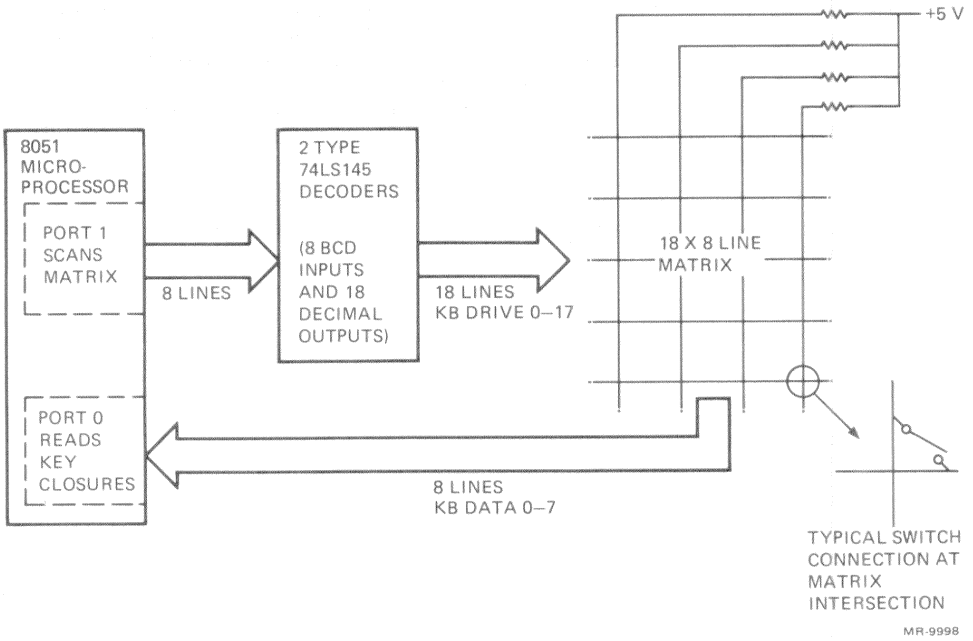


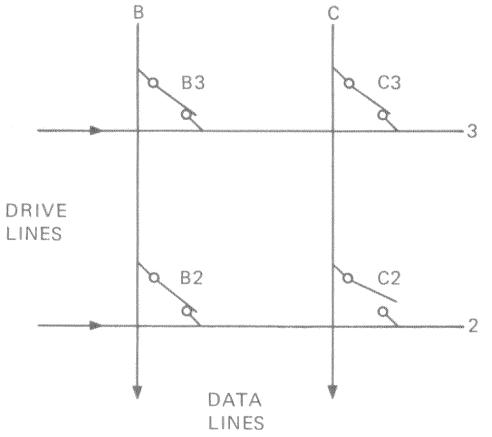
Figure B-4 Matrix Scanning Circuit, Simplified Block Diagram

The other axis of the matrix consists of eight lines tied to +5 V through pull-up resistors. These lines go to port 0 of the 8051 microprocessor and are called KB DATA 0-7.

The 8051 microprocessor scans the 18 drive lines. Key closures are detected by reading the eight data lines. The complete matrix is scanned every 8.33 ms. When a key closure is detected, it is scanned again to verify that it is a key closure and not electrical noise. Once the key closure is verified, the 8051 microprocessor firmware translates the position information into a keycode, and transmits it to the system's central processor. Transmission is handled by the universal asynchronous receiver/transmitter (UART) in the 8051 microprocessor.

A "sneak path" or "ghost key" indication can occur when three of the four corners of a matrix rectangle are closed, as shown in Figure B-5. The key positions in the matrix are arranged to avoid sneak paths. However, if a sneak path does occur, the firmware prevents the keycode for the key (that caused the sneak path) to be transmitted until one of the involved keys is released. This prevents transmission of ghost keys entirely.

Table B-1 shows the keyboard matrix on the LK201-AA (U.S.A.) keyboard. Keycap designations are shown for reference only and can be compared to Figure B-6.



NOTES:

1. CONDITIONS ARE: SWITCHES B2, B3, AND C3 CLOSED, SWITCH C2 OPEN; LINE 2 IS BEING DRIVEN, AND LINE C IS BEING READ.

2. INTERSECTION C2 IS BEING LOOKED AT. IT SHOULD NOT SHOW A KEY CLOSURE BECAUSE SWITCH C2 IS OPEN.

3. HOWEVER, A SNEAK PATH IS PRESENT FROM LINE 2 THROUGH SWITCHES B2, B3, AND C3 TO LINE C. A GHOST KEY IS READ AT INTERSECTION C2.

MR-9999

Figure B-5 Example of Ghost Key Generation

Table B-1 Keyboard Matrix (LK201-AA)

KB Drive	KB Data 7	6	5	4	3	2	1	0
17	Reserved	F19	Reserved	F20	PF4	N---	N, (Note 1)	Enter
		G22		G23	E23 |	D23	C23	A23
16	F18	PF3	Reserved	N9	V	N6	N3	N
	G21	E22		D22	B17	C22	B22	A22
15	F17	PF2	Reserved	N8	N5	->	N2	N0 (Note 2)
	G20	E21		D21	C21	B18	B21	
14	PF1	Next Screen	Remove	^ |	N7	N4	N1	N0
	E20	D18	E18	C17	D20	C20	B20	A20
13	Insert Here	--- -	DO	Prev Screen	{ [	" '	Reserved	Reserved
	E17	E11	G16	D17	D11	C11		
12	Find	+ =	Help	Select	}]	Return	<-	
	E16	E12	G15	D16	D12	C13	B16	C12
11	Addtnl Options	<X| (delete)	Reserved	) 0	P	(Note 3)	: ;	? /
	G14	E13		E10	D10		C10	B10
10	Reserved	F12 (BS)	Reserved	F13 (LF)	(9	O	L	.
		G12		G13	E09	D09	C09	B09
9	Reserved	F11 (ESC)	Reserved	Reserved	* 8	I	K	, ,
		G11			E08	D08	C08	B08
8	Reserved	Main Screen	Reserved	Exit	& 7	U	J	M
		G08		G09	E07	D07	C07	B07

Table B-1 Keyboard Matrix (LK201-AA) (Cont)

KB Drive	KB Data 7	6	5	4	3	2	1	0
7	Reserved	Cancel	Reserved	Resume	^ 6	Y	H	N
		G07		G06	E06	D06	C06	B06
6	Reserved	Reserved	Reserved	Inter-rupt	% 5	T	G	B
				G05	E05	D05	C05	B05
5	F4	Break	Reserved	$ 4	R	F	V	SPACE
	G02	G03		E04	D04	C04	B04	A01-A09
4	Reserved	Print Screen	Reserved	Set-Up	# 3	E	D	C
		G00		G01	E03	D03	C03	B03
3	Hold Screen	@ 2	Reserved	Tab	W	S	X	> <
	G99	E02		D00	D02	C02	B02	B00
2	Reserved	Reserved	Reserved	~	! 1	Q	A	Z
				E00	E01	D01	C01	B01
1	Ctrl	Lock	Compose	Reserved				
	C99	C00	A99					
0	Shift							
	B99,B11							

NOTES:

1. Note that N0--N9, N---, N, and N. refer to the numeric keypad.

2. N0 of the numeric keypad can be divided into two keys. Normally only the N0 keyswitch is implemented as a double-size key.

3. The Return key occupies two positions that are decoded as the Return (C13) key.

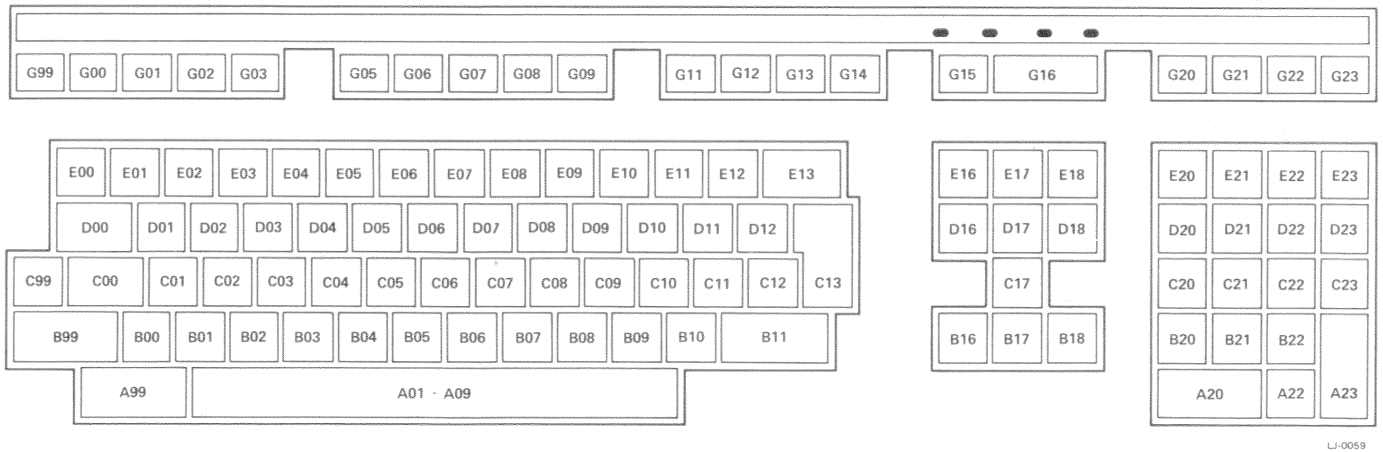


Figure B-6 LK201 Keyboard Layout

B.4.2 Audio Transducer Control Circuit

Figure B-7 shows the audio transducer, or beeper, control circuit. The beeper is driven by a transistor whose base is connected to a 2 kHz square wave from a 556 timer IC. This signal is biased by a network of four type 74LS05 open collector inverters. The 8051 microprocessor controls all four inverters via the firmware. The on/off inverter connects directly to the transistor base. When the 8051 microprocessor puts a high on the on/off inverter input, its output goes low and removes the 2 kHz square wave from the transistor base. This cuts off the transistor and disables the beeper.

To turn on the beeper, the 8051 microprocessor puts a low on the on/off inverter input. Its output goes high and allows the 2 kHz signal to reach the transistor base. This turns on the beeper. The firmware generates a keyclick (on for 2 ms) or a bell tone (on for 125 ms). The 8051 microprocessor sets up the three level control inverters by putting one of eight binary combinations on the inverter inputs. All highs give the softest sound, all lows give the loudest sound.

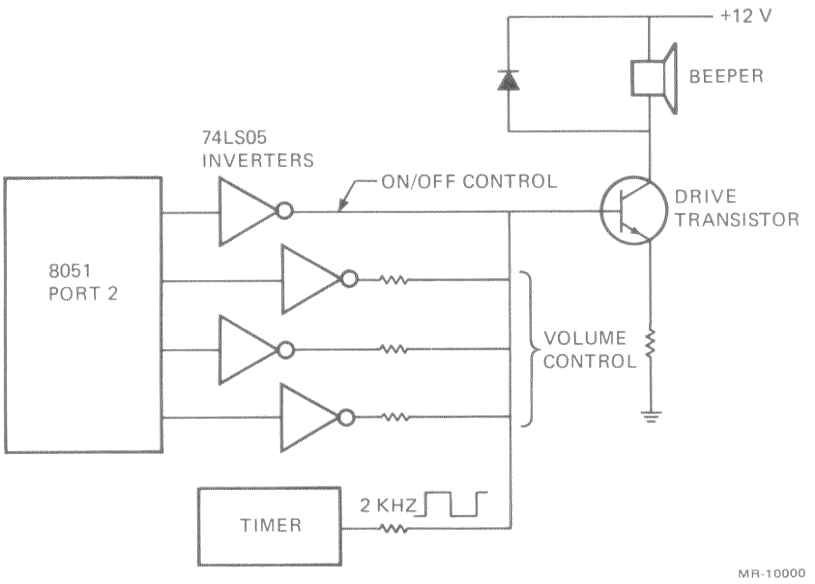


Figure B-7 Audio Transducer (Beeper) Control Circuit

The firmware controls the keyclick and the bell tone independently. The bell tone is sounded only upon request from the system's control processor. The keyclick is sounded (unless disabled) under the following conditions.

1. When a key is pressed.

2. When a metronome code is sent.

3. When a command to sound the keyclick is received from the system's control processor.

B.4.3 Indicator (LED) Control Circuit

Figure B-8 shows the LED indicator control circuit. The control signal for each LED goes from port 2 of the 8051 microprocessor to the input of a type 74LS05 open collector inverter. The inverter output goes to the LED cathode; its anode is connected to +5 V. A separate +5 V source relieves the LED's load on the main +5 V supply.

A low signal from the 8051 microprocessor drives the inverter output high, which cuts off the LED. A high signal from the 8051 drives the inverter output low. This provides a path to ground from the +5 V through the LED. The LED then turns on.

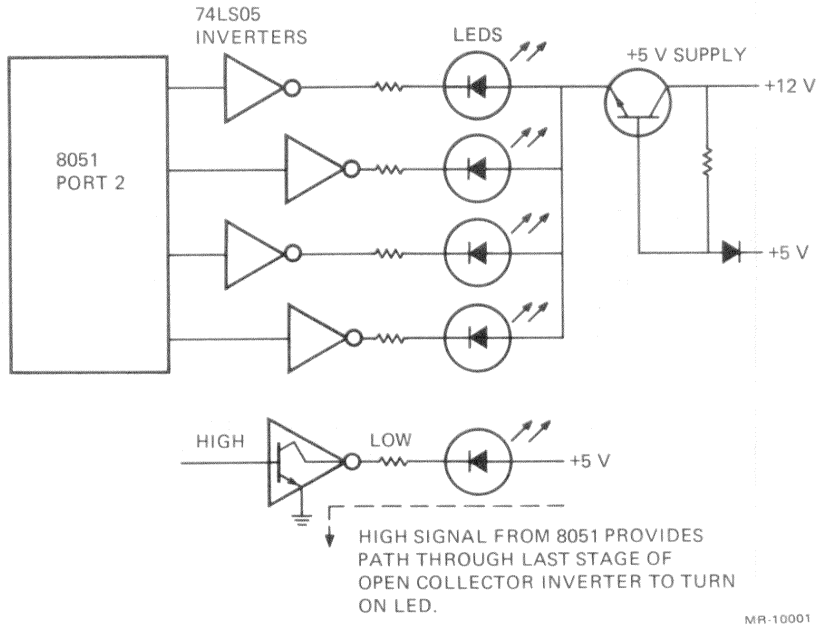


Figure B-8 Indicator (LED) Control Circuit

B.4.4 Keyboard Communication

B.4.4.1 Keyboard Transmit Mode -- The keyboard codes and a few other special codes are transmitted via a serial line output in port 3 of the 8051 microprocessor. The transmitted signal goes from the 8051 microprocessor to a driver, through the keyboard cable, monitor, and video cable to the system's central processor. A UART within the 8051 controls the transmission.

Transmitted characters conform to a specific format. Each character is 10 bits long. The first bit is the START bit. It is always a logical 0 (space). The next eight bits represent the encoded data. The last bit is the STOP bit. It is always a logical 1 (mark). Figure B-9 shows the character format.

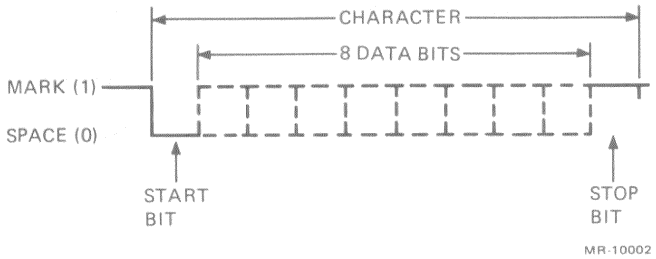


Figure B-9 Keyboard Transmit and Receive Character Format

B.4.4.2 Keyboard Receive Mode -- The firmware contains features that can be enabled by commands from the system's central processor. There are two categories of features: one sets keyboard transmission characteristics, the other controls the keyboard peripherals. A peripheral command covers indicator control, bell and keyclick volume, keyboard ID code, and reinstate keyboard. The commands come from the system's central processor, through the video cable, monitor, and keyboard cable to the receiver, and into the 8051 microprocessor via port 3. They go to the UART in the 8051.

Received characters conform to the same 10-bit format used for transmitted characters. The eight data bits are arranged in a specified protocol depending on the command type.

B.4.5 Reset Signal for the 8051 Microprocessor

Whenever the system is turned on, the 8051 microprocessor in the keyboard must be reset. This allows the 8051 to start operating.

The reset signal generator is active only during power-up. The input is +5 V. The output is connected to the reset input of the 8051. When power is turned on, the +5 voltage starts to rise from zero. The reset signal circuit output follows it, and drops off when a steady state of +5 V is reached. This circuit holds the 8051 reset input high (+3.5 V to +5 V) long enough to enable the reset action in the 8051. This action occurs only during power-up.

B.4.6 Hardware Keyboard Identification (ID)

At power-up, the keyboard performs a self-test and sends the results to the system's central processor. One piece of information to be sent is the keyboard hardware ID, which is read from hardwired jumpers.

There are six jumpers. Each jumper line goes from an input in port 3 of the 8051 microprocessor to ground. All jumpers are installed so the keyboard hardware ID is zero.

B.4.7 Voltage Supplies

The only voltage sent to the keyboard is +12 V. However, +5 V and -10 V are also needed. These voltages are derived from the +12 V.

A +5 V supply handles most of the requirements for this voltage. The four keyboard LEDs have their own +5 V supply. A -10 V supply provides voltage for the driver in the serial out line.

B.5 KEYBOARD PROGRAMMING

This section describes the functions the keyboard performs under system central processor control. It also describes keyboard programming machine language. High-level user programming is not described here.

B.5.1 Keyboard Layout and Key Identification

Each keyboard key has a unique location. Each location is scanned, and when closure or release is detected, the location is verified. This is then decoded to an 8-bit keycode. Figure B-6 shows the keyswitch locations. Table B-2 lists the 14 functional divisions of the keyboard. Table B-3 lists the divisions, keycaps, and keycodes.

Table B-2 Keyboard Functional Divisions

Division	Description	Representation
1	48 graphic keys, spacebar	0001
2	Numeric keypad	0010
3	Delete character (E12)	0011
4	Return (C13) Tab (D00)	0100
5	Lock (C00) Compose (A99)	0101
6	Shift (B99 and B11), Ctrl (C99)	0110
7	Horizontal cursors (B16 and B18)	0111
8	Vertical cursors (B17 and C17)	1000
9	Six keys directly above the cursor keys (D16-D18 and E16-E18)	1001
10	Function keys (G99-G03)	1010
11	Function keys (G05-G09)	1011
12	Function keys (G11-G14)	1100
13	Function keys (G15-G16)	1101
14	Function keys (G20-G23)	1110

Table B-3 Keycode Translation Table

Division	Position	Keycap*	Keycode** (decimal)	Keycode (hexadecimal)
FUNCTION KEYS				
10	G99	Hold Screen	086	56
	G00	Print Screen	087	57
	G01	Set-Up	088	58
	G02	F4	089	59
	G03	Break	090	5A
		Reserved	091-098	5B-62
11		Reserved	099	63
	G05	Interrupt	100	64
	G06	Resume	101	65
	G07	Cancel	102	66
	G08	Main Screen	103	67
	G09	Exit	104	68
		Reserved	105-110	69-6E
12			111	6F
		Reserved	112	70
	G11	F11 (ESC)	113	71
	G12	F12 (BS)	114	72
	G13	F13 (LF)	115	73
	G14	Addtnl Options	116	74
		Reserved	117-122	75-7A
13		Reserved	123	7B
	G15	Help	124	7C
	G16	Do	125	7D
14		Reserved	126-127	7E-7F
	G20	F17	128	80
	G21	F18	129	81
	G22	F19	130	82
	G23	F20	131	83
		Reserved	132-135	84-87
BASIC EDITING KEYS				
9		Reserved	136-137	88-89
	E16	Find	138	8A
	E17	Insert Here	139	8B
	E18	Remove	140	8C
	D16	Select	141	8D
	D17	Prev Screen	142	8E
	D18	Next Screen	143	8F
		Reserved	144	90

Table B-3 Keycode Translation Table (Cont)

Division	Position	Keycap*	Keycode** (decimal)	Keycode (hexadecimal)
NUMERIC KEYPAD				
2		Reserved	145	91
	A20	0	146	92
		Reserved	147	93
	A22	.	148	94
	A23	Enter	149	95
	B20	1	150	96
	B21	2	151	97
	B22	3	152	98
	C20	4	153	99
	C21	5	154	9A
	C22	6	155	9B
	C23	,	156	9C
	D20	7	157	9D
	D21	8	158	9E
	D22	9	159	9F
	D23	-	160	A0
	E20	PF1	161	A1
	E21	PF2	162	A2
	E22	PF3	163	A3
	E23	PF4	164	A4
		Reserved	165	A5
CURSOR KEYS				
7		Reserved	166	A6
	B16	Left	167	A7
	B18	Right	168	A8
8	B17	Down	169	A9
	C17	Up	170	AA
		Reserved	171-172	AB-AC
SHIFT, LOCK, CTRL, A99, and A10				
6		Reserved	173	AD
	B99,B11	Shift	174	AE
	C99	Ctrl	175	AF
5	C00	Lock	176	B0
	A99	Compose	177	B1
		Reserved	178	B2

Table B-3 Keycode Translation Table (Cont)

Division	Position	Keycap*	Keycode** (decimal)	Keycode (hexadecimal)
SPECIAL CODES				
		All Ups	179	B3
		Metronome	180	B4
		Output Error	181	B5
		Input Error	182	B6
		KBD LOCKED Acknowledge	183	B7
		TEST MODE Acknowledge	184	B8
		PREFIX to keys Down	185	B9
		MODE CHANGE Acknowledge	186	BA
		Reserved	187	BB
DELETE				
3	E13	Delete <X|	188	BC
RETURN and TAB				
4	C13	Return	189	BD
	D00	Tab	190	BE
48 GRAPHICS KEYS and SPACEBAR				
2	E00	~	191	BF
	E01	!1	192	D0
	D01	Q	193	C1
	C01	A	194	C2
	B01	Z	195	C3
		Reserved	196	C4
	E02	@2	197	C5
	D02	W	198	C6
	C02	S	199	C7
	B02	X	200	C8
	B00	><	201	C9
		Reserved	202	CA
	E03	#3	203	CB
	D03	E	204	CC
	C03	D	205	CD
	B03	C	206	CE
		Reserved	207	CF

Table B-3 Keycode Translation Table (Cont)

Division	Position	Keycap*	Keycode** (decimal)	Keycode (hexadecimal)
	E04	$4	208	D0
	D04	R	209	D1
	C04	F	210	D2
	C04	V	211	D3
	A01-A09	Space	212	D4
		Reserved	213	D5
	E05	%5	214	D6
	D05	T	215	D7
	C05	G	216	D8
	B05	B	217	D9
		Reserved	218	DA
	E06	^6	219	DB
	D06	Y	220	DC
	C06	H	221	DD
	B06	N	222	DE
1		Reserved	223	DF
	E07	&7	224	E0
	D07	U	225	E1
	C07	J	226	E2
	B07	M	227	E3
		Reserved	228	E4
	C08	*8	229	E5
	D08	I	230	E6
	C08	K	231	E7
	B08	"	232	E8
		Reserved	233	E9
	E09	(9	234	EA
	D09	0	235	EB
	C09	L	236	EC
	B09	. .	237	ED
		Reserved	238	EE
	E10	)0	239	EF
	D10	P	240	F0
		Reserved	241	F1
	C10	: ;	242	F2
	B10	? /	243	F3
		Reserved	244	F4
	E12	+ =	245	F5
	D12	}]	246	F6
	C12	| \	247	F7
		Reserved	248	F8

Table B-3 Keycode Translation Table (Cont)

Division	Position	Keycap*	Keycode** (decimal)	Keycode (hexadecimal)
	E11	_ -	249	F9
	D11	{ [	250	FA
	C11	< ,	251	FB
		Reserved	252-255	FC-FF

NOTES:

* The legends under Keycap are taken from the keycap legends of the LK201-AA (U.S.A.).

** Keycodes 000 through 064 are reserved. Keycodes 065 through 085 are unused.

B.5.2 Modes

This section describes the functions of the keycode transmission modes. The mode set command allows any 1 of the 14 keyboard divisions to be set to any 1 of the following 3 modes. (Division defaults are described in subsequent paragraphs.)

1. Down-Only Mode — The keyboard transmits a keycode when the key is pressed.

2. Auto-Repeat Down — The keyboard transmits a keycode when the key is first pressed. If the key is held down past the specified timeout period (usually 300 to 500 ms), a fixed metronome code is sent at the specified rate until the key is released.

3. Down/Up — The keyboard transmits a keycode when the key is pressed and an up code when the key is released. If any other down/up keys are pressed, the up code is a repeat of the down code. If no other down/up keys are pressed, the keyboard sends an ALL UPS code.

B.5.2.1 **Special Considerations Regarding Auto-Repeat** -- The auto-repeat rate set command allows the following changes in auto-repeat mode:

1. The auto-repeat rate buffer association can be changed for the selected keyboard division.

2. The timeout and interval values can be changed in any one of the four auto-repeat rate buffers.

3. If multiple auto-repeating keys are held down, metronome codes are still generated. The metronome codes apply to the keycode transmitted most recently. If the last key pressed down is released, and another key is still down, the keycode of the key still down is retransmitted.

   Example: The A key is held down.

   This produces the following transmission.

   A metronome metronome

   Now the B key is pressed. This produces the following transmission.

   A metronome metronome B metronome metronome

   Now the B key is released. This produces the following transmission.

   A metronome metronome B metronome metronome A metronome met.....

   While metronome codes are being generated for an auto-repeating key, a nonauto-repeating keycode or special code may be transmitted. The keyboard transmits this special code instead of the next metronome code, and then returns to the auto-repeated code. The keycode to be auto-repeated is always the last byte transmitted.

   Example: The A key is held down.

   This produces the following transmission.

   A metronome metronome

   Now the shift key is pressed. This produces the following transmission.

   A metronome metronome shift A metronome

   Now the shift key is released. This produces the following transmission.

   A metronome metronome shift A metronome ALL UPS A metronome met.....

4. If an auto-repeating key is not to auto-repeat (for example, Ctrl C), the system module must issue a temporary inhibit auto-repeat command. This halts the transmission of any metronome codes or keyclicks for that key only. Metronome codes continue when another key is pressed. The command must be issued after the keycode for the auto-repeating key is received.

5. Auto-repeat can be enabled and disabled independently of the division settings by using the enable/disable auto-repeat commands. These commands apply to all keys on the keyboard. When auto-repeat is disabled, the keyboard continues to auto-repeat characters internally. However, it does not transmit metronome codes or keyclicks. When auto-repeat is enabled, the keyboard transmits the metronome codes from the point where they were before auto-repeat was disabled. This may be within either the timeout or interval period, depending on the time elapsed since the key was pressed.

6. If the keyboard receives a request to change a division mode to auto-repeat while a key is being pressed, the keyboard makes the change immediately. After the specified timeout period, the keyboard transmits metronome codes for the pressed key. In place of the first metronome code, the keyboard transmits the keycode of the auto-repeating key.

All auto-repeating division modes can be changed to down-only with one command. This and other auto-repeat commands are grouped with the peripheral commands.

B.5.2.2 **Special Considerations Regarding Down/Up Mode** -- If two down/up keys are released simultaneously (within the same scan), and there are no other down/up keys down on the keyboard, only one ALL UPS code is generated.

B.5.2.3 **Auto-Repeat Rates** -- Four buffers in the keyboard, numbered 0 through 3, store auto-repeat rates. Each buffer stores two values, which can be changed by the system module:

1. Timeout value
2. Interval value.

Timeout is the amount of time the keyboard waits before starting to auto-repeat a character. The timeout value is the amount of time between the detection of a down key and the transmission of the first metronome code. (Defaults range from 300 to 500 ms.) The rate of auto-repeating a character is called the "interval." The interval value is the number of metronome codes per second (defaults to 30).

Each division is associated with one of the four buffers. Rates are taken from the associated buffer each time the auto-repeat timers are loaded. This buffer-to-division association can be changed by the system module or left to default.

B.5.3 Keyboard Peripherals

This section describes the peripherals available on the keyboard. The keyclick, bell, and LEDs are all considered keyboard peripherals.

B.5.3.1 **Audio** -- The keyclick is a 2 ms beep; the bell is a 125 ms beep. The bell is sounded only upon request from the system module. The keyclick (if not disabled by the system module) is sounded when:

1. A key is pressed
2. A metronome code is sent
3. The system module receives a sound keyclick command.

If either the B11 or B99 key (the left or right shift key on the LK201) or the C99 key (the Ctrl key on the LK201) is pressed, the keyclick is not generated. However, if a command is sent from the system module to enable the keyclick on the C99 key, the keyclick is generated. Figure B-6 shows the positions of these keys.

The keyclick or bell (or both) may be disabled not to sound. If the system module requests sound (see Paragraph B.5.5.3), the keyclick or the bell does not sound.

Both the keyclick and bell may be set independently to one of the following eight volume levels:

000 - highest
001
010 - default
011
100
101
110
111 - lowest

B.5.3.2 **Indicators (LEDs)** -- The system module normally transmits indicator control commands. However, the following are exceptions.

1. Upon power-up, the keyboard turns all LEDs off.

2. After receiving the inhibit transmission command, the keyboard turns on the lock LED. The LED is turned off after the keyboard receives a resume transmission command.

B.5.4 **Keyboard-to-System Module Protocol**
The following paragraphs describe the keyboard-to-system module protocol.

B.5.4.1 **Keycode Transmission** -- The keyboard transmits single-byte keycodes that reflect the keyboard matrix status. The 8-bit codes above 64 (decimal) are used for keycodes. Every key is identified by a unique keycode. There are no special codes for shifted or control keys.

Refer to Figure B-6 and Tables B-1 and B-2 for the complete keycode matrix translation table.

B.5.4.2 **Special Code Transmission** -- There are 13 special codes: 9 codes with values above 64 (decimal) and 4 codes below. Table B-4 lists the nine special codes above the 64 (decimal) keycode value range. Table B-5 lists the four special codes below the 64 (decimal) value range.

Table B-4 Special Codes Above 64 (Decimal)

Special Codes	Keycode (Decimal)	(Hexadecimal)
All Ups	179	B3
Metronome Code	180	B4
Output Error	181	B5
Input Error	182	B6
KBD Locked ACK	183	B7
Test Mode ACK	184	B8
Prefix to Keys Down	185	B9
Mode Change ACK	186	BA
Reserved	127	7F

All Ups -- Indicates to the system module that a down/up mode key was just released and no other down/up keys are being pressed.

Metronome Code -- Indicates to the system module that an interval has passed, a keyclick has been generated, and the last key received by the system module is still being pressed.

Output Error -- Indicates an output buffer overflow to the system module. The overflow occurred after receiving a keyboard inhibit command from the system module and some keystrokes may be lost.

Input Error Code -- Indicates to the system module that the keyboard received a meaningless command, too many, or too few parameters.

Keyboard Locked Acknowledge -- Indicates to the system module that the keyboard received an inhibit transmission command.

Test Mode Acknowledge -- Indicates that the keyboard has entered test mode. This is a special mode used during the production test. If the system module receives this acknowledge, it sends 80 (hexadecimal). This terminates the test mode and jumps to power-up.

Prefix to Keys Down -- Indicates that the next byte is a keycode for a key already down in a division that has been changed to down/up.

Mode Change Acknowledge -- Indicates that the keyboard has received and processed a mode change command.

Reserved -- Keycode 7F is reserved for internal use.

Table B-5 Special Codes Below 64 (Decimal)

Special Codes	Keycode (Decimal)	(Hexadecimal)
Keyboard ID -- Firmware	01	01
Keyboard ID -- Hardware	00	00
Key Down on Power-Up Error Code	61	3D
Power-Up Self-test Error Code	62	3E

Keyboard ID -- This is a 2-byte identification code, transmitted after the power-up self-test (power-up transmission). It is also sent on request from the system module.

Key Down on Power-Up Error Code -- Indicates that a key was pressed on power-up.

Power-Up Self-Test Error Code -- Indicates to the system module that the ROM or RAM self-test of the system module failed.

B.5.4.3 **Power-Up Transmission** -- Upon power-up, the keyboard performs a self-test in less than 70 ms. It transmits the self-test results to the system module in 4 bytes.

Byte 1: KBID (firmware) -- This is the keyboard identification (ID) that is stored in the firmware.

Byte 2: KBID (hardware) -- This is the keyboard ID that is read from hardware jumpers.

Byte 3: ERROR -- Two error codes indicate either failure of the ROM or RAM self-test within the processor (3E hexadecimal), or key-down on power-up (3D hexadecimal). No error is indicated by 00.

Byte 4: KEYCODE -- This byte contains the first keycode detected if there was a key down on power-up. No error is indicated by 00.

If the ROM self-test (CHECKSUM) fails and the error is fatal, the keyboard is unable to transmit. Nonfatal errors permit the keyboard to continue operation.

If the keyboard finds a key down on the first scan, it continues to look for an ALL UP condition. The keyboard sends the corrected 4-byte power-up sequence when the pressed key is released. This avoids a fatal error condition if a key is pressed by mistake while powering up.

The keyboard LEDs are lit during the power-up self-test. If the self-test is passed, the keyboard turns the LEDs off. If a bell is selected on power-up, the system module can transmit a sound bell command to the keyboard. However, this should not be done until the system module receives the last byte of the 4-byte sequence. The request for self-test tests the serial line and system module connection. The power-up self-test takes 70 ms or less.

The system module can request a jump to power-up at any time. This causes the LEDs on the keyboard to blink on and off (for the power-up self-test).

B.5.5 System Module-to-Keyboard Protocol

The system module controls both the peripherals associated with the keyboard and the keyboard transmit characteristics. Figure B-10 shows the protocol for the transmission of commands and parameters from the system module to the keyboard.

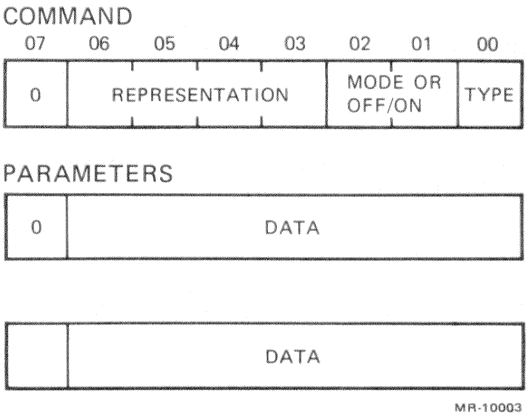


Figure B-10 System Module-to-Keyboard Protocol

Table B-6 Command Types

Transmission Commands	Peripheral Commands
Mode Set	Flow Control
Auto-Repeat Rate Set	Indicator
	Audio
	Keyboard ID
	Reinitiate Keyboard
	Some Auto-Repeat Control
	Jump to Test Mode
	Reinstate Defaults

B.5.5.1 **Commands** -- Table B-6 lists the two kinds of commands: those that control keyboard transmission characteristics, and those that control keyboard peripherals. The low bit of the command is the TYPE flag. It is clear if the command is a transmission command; it is set if the command is a peripheral command.

The high-order bit of every command is the PARAMS flag. If there are any parameters to follow, this flag is clear. If there are no parameters, this flag is set.

B.5.5.2 **Parameters** -- The high-order bit of every parameter is the PARAMS flag. It is clear if there are parameters to follow. It is set on the last parameter. The remaining seven bits of the parameter are for data.

B.5.5.3 **Peripheral Commands** -- Two commands can turn the data flow from the keyboard off and on:

1. Inhibit Keyboard Transmission -- This command shuts off, or locks, the keyboard and turns on the lock LED. After receiving the inhibit command, the keyboard sends a special command to the system's central processor. If the central processor receives this code without requesting it, this indicates that noise on the line was interpreted as the inhibit command. The central processor then responds immediately with the resume keyboard transmission command.

2. Resume Keyboard Transmission -- This command turns on, or unlocks, the keyboard and turns off the lock LED. If any keystrokes are lost, the keyboard responds with an error code.

Each keyboard LED can be turned on and off. The following are the eight commands that control the keyclick and bell sounds.

1. Disable Keyclick
2. Enable Keyclick and Set Volume
3. Disable Ctrl Keyclick
4. Enable Ctrl Keyclick
5. Sound Keyclick
6. Disable Bell
7. Enable Bell and Set Volume
8. Sound Bell

The following four commands are related to the control of auto-repeat mode.

1. Temporary Auto-Repeat Inhibit -- Auto-repeat is stopped for a specific key only. It resumes automatically when another key is pressed.

2. Enable Auto-Repeat Across the Board -- Starts transmission of metronome codes without affecting auto-repeat timing or keyboard division.

3. Disable Auto-Repeat Across the Board -- Stops transmission of metronome codes without affecting auto-repeat timing or keyboard division.

4. Change All Auto-Repeat to Down-Only -- Changes all keyboard auto-repeating divisions to down-only mode.

The following are three other miscellaneous commands.

1. Request Keyboard ID -- The keyboard sends the 2-byte ID (firmware and hardware). The keyboard does not jump to the power-up sequence.

2. Reinitiate Keyboard -- The keyboard jumps to the power-up sequence. Transmission to the keyboard should be held until the host processor receives the last byte of the power-up self-test.

3. Reinstate Defaults -- Sets the following functions back to the default settings after a successful completion of the power-up self-test.

   Division mode settings
   Auto-repeat interval and timeout rates
   Auto-repeat buffer selections
   Audio volume
   Ctrl key keyclick

To send a peripheral command, set the TYPE flag (low-order bit). Bits 6-3 contain a command representation from the table below. Bits 2 and 1 specify on (01), off (00), or sound (11). Bit 7 should be set if there are no parameters to follow.

Table B-7 lists the peripheral commands (in hexadecimal) and Table B-8 indicates the representation for each command function.

The jump to power-up command is FD (hexadecimal).

The following are some of the peripheral commands.

1. Flow Control -- The system module can lock the keyboard with the inhibit keyboard transmission command. When the keyboard is unlocked, it responds with an error code if any keystrokes were missed.

Table B-7 Peripheral Commands in Hexadecimal

Function	Hexadecimal	Parameters
FLOW CONTROL		
Resume Keyboard Transmission	8B	None
Inhibit Keyboard Transmission	89	None
INDICATORS		
Light LEDs	13	Bit pattern
Turn Off LEDs	11	Bit pattern
AUDIO		
Disable Keyclick	99	None
Enable Keyclick, Set Volume	1B	Volume
Disable Ctrl Keyclick	B9	None
Enable Ctrl Keyclick	BB	None
Sound Keyclick	9F	None
Disable Bell	A1	None
Enable Bell, Set Volume	23	Volume
Sound Bell	A7	None
AUTO-REPEAT		
Temporary Auto-Repeat Inhibit	C1	None
Enable Auto-Repeat Across Keyboard	E3	None
Disable Auto-Repeat Across Keyboard	E1	None
Change All Auto-Repeat to Down-Only	D9	None
OTHER		
Request Keyboard ID	AB	None
Jump to Power-Up	FD	None
Jump to Test Mode	CB	None
Reinstate Defaults	D3	None

Table B-8 Command Function Representation

Command	Representation
Flow Control	0001
Indicator (LEDs)	0010
Keyclick	0011
Bell	0100
Keyboard ID	0101
Keyclick for Ctrl Key	0111
Temporarily Inhibit Auto-Repeat	1000
Jump to Test Mode	1001
Change All Auto-Repeat Characters to Down-Only	1010
Enable/Disable Auto-Repeat	1100

2. Indicators (LEDs) -- Figure B-11 shows the LED parameter. Figure B-12 shows the LED layout on the LK201 keyboard without the label strip installed.

3. Audio -- Figure B-13 shows the audio volume parameter.

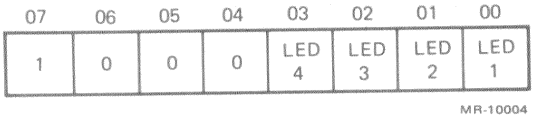


Figure B-11 Indicator (LED) Parameter

Figure B-12 Indicator (LED) Layout

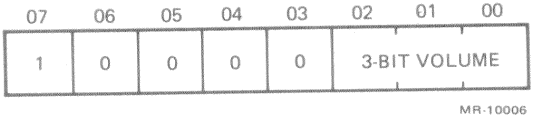


Figure B-13 Audio Volume Parameter

The volume levels for the audio are as follows.

000 - highest
001
010
011
100
101
110
111 - lowest

The keyclick or the bell (or both) can be disabled. When the keyclick or bell is disabled, it does not sound, even if the system module requests it.

The following are additional peripheral commands.

1. Temporary Auto-Repeat Inhibit -- Stops auto-repeat for this key only. Auto-repeat automatically continues when another key is pressed.

2. Disable/Enable Auto-Repeat Across Keyboard -- Stops/starts transmission of metronome codes without affecting auto-repeat timing or division settings.

3. Change All Auto-Repeat to Down-Only -- Changes division settings for all auto-repeating divisions to down-only.

4. Request Keyboard ID -- Keyboard sends a 2-byte keyboard ID. Keyboard does not jump to power-up.

5. Reinitiate Keyboard -- Keyboard jumps to its power-up routine. The system module should not try to transmit anything to the keyboard until the last byte of the power-up sequence is received.

6. Jump to Test Mode -- Special test mode for manufacturing testing.

7. Reinstate Defaults -- Set the following functions back to the default settings after a successful completion of the power-up self-test.

   Division mode settings
   Auto-repeat interval and timeout rates
   Auto-repeat buffer selections
   Audio volume
   Ctrl key keyclick

B.5.5.4 Mode Set Commands -- The following describe the mode set commands.

1. Division mode settings -- Refer to Paragraph B.5.2.

2. Each division on the keyboard has a unique 4-bit representation. Table B-2 describes these representations.

3. Each mode has a unique 2-bit code as indicated in Table B-9.

To set the key transmission mode on a particular keyboard division, the system module must send the PARAMS flag, then the keyboard division representation with the mode code, and then the TYPE flag (cleared).

Example: Set main array to down/up. Refer to Figure B-14.

The PARAMS flag is set to 1 if there are no parameters. The PARAMS flag is clear if there are parameters.

Table B-9 Mode Representation

Modes	Representation
Down-only	00
Auto-repeat down	01
Down/up	11

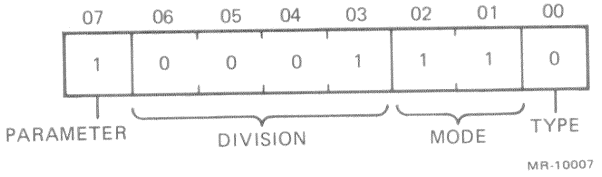


Figure B-14 Setting Key Transmission Mode

Auto-repeat rate buffer association -- If the auto-repeat mode is selected, the system module can transmit a parameter to change the buffer association of the selected division.

Example: Set main array to auto-repeat, change buffer association to buffer 3. Refer to Figure B-15.

Auto-repeat rate buffer values -- At keyboard power-up time, the four auto-repeat rate buffers contain default values. The system module may change these values.

In the command byte, bit 7 (PARAMS flag) should be clear, bits 6-3 are llll (to indicate that this is a rate set command), bits 2 and 1 should be the buffer number (0 to 3), bit 0 (TYPE flag) is clear. There should be two parameters carrying the rate set data.

Example: Change rates in buffer 3. Refer to Figure B-16.

The first parameter specifies the timeout to the store in the selected buffer. The second parameter specifies the interval.

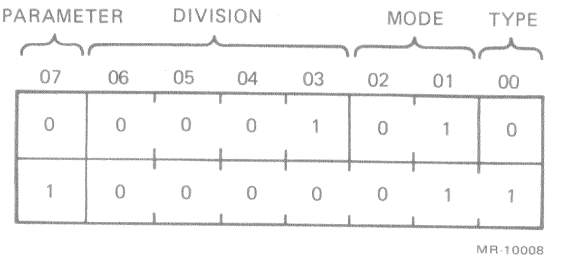


Figure B-15 Setting Auto-Repeat Rate Buffer Association

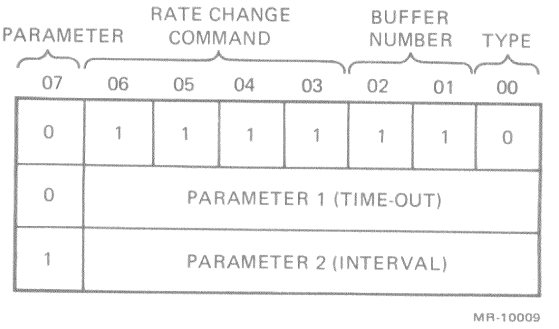


Figure B-16 Setting Auto-Repeat Rate Buffer Values

For example, to set the auto-repeat rate in buffer 1, the system module firmware transmits 00000011, followed by 2 bytes of numeric parameters.

The auto-repeat timeout is the transmitted number times 5 ms. To specify a rate of 5 ms delay, the first parameter received is 00000001. The maximum allowable time is 630 ms (01111110). The system module must not send 635 (01111111).

NOTE

The code (01111111) is reserved for internal keyboard use. 00 is an illegal value.

Auto-repeat timeout is implemented as a multiple of 8.33 ms, the keyboard's internal scan rate. Timeout rates can vary + 4.15 ms.

The auto-repeat interval is the number of metronome codes per second. In order to specify a speed of 16 Hz, the second parameter received is 10010000. Note that the high-order bit is set because it is the last parameter. The highest value that may be sent is 124 (11111100).

The lowest rate that can be implemented by the keyboard is 12 Hz. Values as low as 1 can be transmitted, but are translated to 12 Hz.

NOTE

The system module must not send 125, or 11111101. This code is the power-up command.

B.5.6 Special Considerations

The following paragraphs describe the special codes and their considerations.

B.5.6.1 **Error Handling** -- There are four error codes. The first two are sent at power-up if the self-test fails. The other two are the input error code and the output error code.

The output error (B5 hexadecimal) is sent after the keyboard receives a resume transmission command, if the output buffer overflowed while the keyboard was locked.

The input error (B6 hexadecimal) is sent when the keyboard detects noise (an unidentified command or parameter) on the line. B6 is also sent if the keyboard detects a delay of more than 100 ms while expecting a parameter.

B.5.6.2 **Keyboard Locked Condition** -- When the keyboard receives an inhibit transmission command, it lights the lock LED and transmits one more byte - a special code indicating the keyboard is locked (keyboard locked acknowledge). If the system module receives this code without a request, noise on the line was interpreted as an inhibit transmission command. The system module should immediately send the resume transmission command to unlock the keyboard.

The output first-in/first-out (FIFO) buffer in RAM is four bytes. When the keyboard is locked, the buffer attempts to store characters received from the keyboard. The keyboard stops scanning its matrix. When the keyboard is unlocked by the system module, it transmits all four bytes in the output buffer. If any keystrokes were missed due to buffer overflow, the keyboard transmits an error code as the fifth byte (output error). Keys that were not transmitted and are being held down when the keyboard is unlocked are processed as new keys. Upon unlocking the keyboard, an error code indicates a possible loss of keystrokes to the system module.

The keyboard stops scanning its matrix when its buffer is full. However, it processes all incoming commands.

B.5.6.3 **Reserved Code** -- The number 7F (hexadecimal) is reserved for the internal keyboard input and output buffers handling routines.

B.5.6.4 **Test Mode** -- The keyboard jumps to a test mode by command during production test. It transmits a special code to the system module to confirm the test mode. If the system module receives this code, it should send the byte 80 (hexadecimal) to continue. This causes a jump to power-up.

B.5.6.5 **Future Expansion** -- Some keycodes are reserved for future use as special codes or keycodes.

B.5.7 **Default Conditions**

Certain keyboard divisions have specific default modes. Some divisions default to the auto-repeat mode; therefore, they have an associated buffer that contains the default values for timeout and interval. Table B-10 lists the default modes and Table B-11 lists the default rates in the four keyboard division auto-repeat rate buffers.

The volume level for the keyclick and bell has an 8-step range. The default volume level for the keyclick and bell is the third loudest. Both keyclick and bell volumes are 2 (decimal) (010 binary) by default. The key in position C99 of the keyboard (the Ctrl key on the LK201) does not generate a click unless enabled by the system module. The keys in position B99 and B11 (shift keys on the LK201) never generate a keyclick.

For the LK201 keyboard, the Ctrl key defaults to the no-keyclick state.

Table B-10 Keyboard Division Default Modes

Keyboard Division	Mode	AR Buffer
Main array	Auto-repeat	0
Keypad	Auto-repeat	0
Delete	Auto-repeat	1
Cursor keys	Auto-repeat	1
Return and Tab	Down-only	
Lock and Compose	Down-only	
Shift and Control	Down/up	
Six basic editing keys	Down/up	

Table B-11 Default Rates in Auto-Repeat Buffers

Buffer Number	Timeout (ms)	Internal (Hz)
0	500	30
1	300	30
2	500	40
3	300	40

B.6 SPECIFICATIONS

FUNCTIONAL

Electronics	8-bit microprocessor, 4 Kbytes of ROM, 256 bytes of RAM, 4 LEDs, transducer
Cord	1.9 m (6 ft), coiled, 4-pin telephone-type modular connectors, plugs into display monitor (PN BCC01)
Keypad	Sculptured key array
Home row key height	3 cm (1.2 in) above desk top
Keys	105 matte, texture-finished keys
Main keypad	57 keys
Numeric keypad	18 keys
Special function keypad	20 keys; firmware- and software-driven
Editing keypad	10 keys

Spacing	1.9 cm (0.75 in) center-to-center (single-width keys)
Wobble	Less than 0.5 cm (0.020 in)
Diagnostics	Power-up self-test, generates identification upon passing test
PHYSICAL	
Height	5 cm (2.0 in) at highest point
Length	53.3 cm (21 in)
Width	17.1 cm (6.75 in)
Weight	2 kg (4.5 lb)

B.7 CHARACTER SETS

The MicroVAX workstation recognizes all the 8-bit character codes of Digital's multinational character set, shown in Figure B-17. It also recognizes the special graphics character set, shown in Figure B-18, when preceded by a select character set (SCS) sequence (described in subsequent paragraphs).

Using Digital's multinational character set allows the MicroVAX workstation to process character codes from the keyboards listed in Table B-12.

Each of the 15 keyboards supported is used with corresponding language ROMs that are inserted into the system module. The language ROMs translate keyboard position codes into the required character codes necessary for further processing.

Table B-12 Multinational Character Set Keyboards

Keyboard	Part Number
American (English)	LK201-AA
Austrian/German	LK201-AG
Belgian/Flemish	LK201-AB
Belgian/French	LK201-AP
British	LK201-AE
Canadian (French)	LK201-AC
Danish	LK201-AD
Dutch	LK201-AH
Finnish	LK201-AF
Italian	LK201-AI
Norwegian	LK201-AN
Spanish	LK201-AS
Swedish	LK201-AM
Swiss (French)	LK201-AK
Swiss (German)	LK201-AL

COLUMN		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
b8 b7 b6 b5 / b4 b3 b2 b1	ROW	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p		DCS		°	À		à	
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q		PU1	¡	±	Á	Ñ	á	ñ
0 0 1 0	2	STX	DC2	"	2	B	R	b	r		PU2	¢	²	Â	Ò	â	ò
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s		STS	£	³	Ã	Ó	ã	ó
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t	IND	CCH			Ä	Ô	ä	ô
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u	NEL	MW	¥	µ	Å	Õ	å	õ
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v	SSA	SPA		¶	Æ	Ö	æ	ö
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w	ESA	EPA	§	·	Ç	Œ	ç	œ
1 0 0 0	8	BS	CAN	(	8	H	X	h	x	HTS		¤		È	Ø	è	ø
1 0 0 1	9	HT	EM	)	9	I	Y	i	y	HTJ		©	¹	É	Ù	é	ù
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z	VTS		ª	º	Ê	Ú	ê	ú
1 0 1 1	11	VT	ESC	+	;	K	[	k	{	PLD	CSI	«	»	Ë	Û	ë	û
1 1 0 0	12	FF	FS	,	<	L	\	l	|	PLU	ST		¼	Ì	Ü	ì	ü
1 1 0 1	13	CR	GS	-	=	M	]	m	}	RI	OSC		½	Í	Ý	í	ý
1 1 1 0	14	SO	RS	.	>	N	^	n	~	SS2	PM			Î		î	¿
1 1 1 1	15	SI	US	/	?	O	_	o	DEL	SS3	APC		¿	Ï	ß	ï	

ASCII CONTROL SET (C0) | ASCII GRAPHIC CHARACTER SET (GL) | ADD'L CONTROL SET (C1) | DEC SUPPLEMENTAL GRAPHIC SET (GR)

DEC MULTINATIONAL CHARACTER SET

MR-10132

Figure B-17 Digital's Multinational Character Set

The MicroVAX workstation processes 7-bit character codes as though they were 8-bit character codes with the eighth bit not set. Each character set consists of displayable or graphic characters, and nondisplayable or control characters.

In 7-bit-coded character sets, control characters are contained in columns 0 and 1, and in position 7/15, while graphic characters are contained in the remaining positions of columns 2 through 7.

In 8-bit-coded character sets, control characters are contained in columns 0, 1, 8, and 9, and in positions 7/15 and 15/15, while graphic characters are contained in the remaining positions of columns 2 through 7, and 10 through 15.

ROW	COLUMN 0	1	2	3	4	5	6	7
0	NUL		SP	0	@	P	`	— SCAN 3
1		DC1 (XON)	!	1	A	Q	a	— SCAN 5
2			"	2	B	R	b	— SCAN 7
3	ETX	DC3 (XOFF)	#	3	C	S	c	— SCAN 9
4	EOT		$	4	D	T	d	┼
5	ENQ		%	5	E	U	e	┤
6			&	6	F	V	f	┴
7	BEL		'	7	G	W	g	┬
8	BS	CAN	(	8	H	X	h	│
9	HT		)	9	I	Y	i	≤
10	LF	SUB	*	:	J	Z	j	≥
11	VT	ESC	+	;	K	[	k	π
12	FF		,	<	L	\	l	≠
13	CR		-	=	M	]	m	£
14	SO		.	>	N	^	n	·
15	SI		/	?	O		o	

	0	1	2	3	4	5	6	7
ROW	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
0	0	20 16 10	40 32 20	60 48 30	100 64 40	120 80 50	140 96 60	160 112 70
1	1	21 17 11	41 33 21	61 49 31	101 65 41	121 81 51	141 97 61	161 113 71
2	2	22 18 12	42 34 22	62 50 32	102 66 42	122 82 52	142 98 62	162 114 72
3	3	23 19 13	43 35 23	63 51 33	103 67 43	123 83 53	143 99 63	163 115 73
4	4	24 20 14	44 36 24	64 52 34	104 68 44	124 84 54	144 100 64	164 116 74
5	5	25 21 15	45 37 25	65 53 35	105 69 45	125 85 55	145 101 65	165 117 75
6	6	26 22 16	46 38 26	66 54 36	106 70 46	126 86 56	146 102 66	166 118 76
7	7	27 23 17	47 39 27	67 55 37	107 71 47	127 87 57	147 103 67	167 119 77
8	10 8 8	30 24 18	50 40 28	70 56 38	110 72 48	130 88 58	150 104 68	170 120 78
9	11 9 9	31 25 19	51 41 29	71 57 39	111 73 49	131 89 59	151 105 69	171 121 79
10	12 10 A	32 26 1A	52 42 2A	72 58 3A	112 74 4A	132 90 5A	152 106 6A	172 122 7A
11	13 11 B	33 27 1B	53 43 2B	73 59 3B	113 75 4B	133 91 5B	153 107 6B	173 123 7B
12	14 12 C	34 28 1C	54 44 2C	74 60 3C	114 76 4C	134 92 5C	154 108 6C	174 124 7C
13	15 13 D	35 29 1D	55 45 2D	75 61 3D	115 77 4D	135 93 5D	155 109 6D	175 125 7D
14	16 14 E	36 30 1E	56 46 2E	76 62 3E	116 78 4E	136 94 5E	156 110 6E	176 126 7E
15	17 15 F	37 31 1F	57 47 2F	77 63 3F	117 79 4F	137 95 5F	157 111 6F	177 127 7F

KEY

CHARACTER: ESC | 33 OCTAL | 27 DECIMAL | 1B HEX

MR-10133

Figure B-18 Special Graphics Character Set

In all character sets, the control characters in columns 0 and 1, and position 7/15, are designated as belonging to the control character group C0. The graphic characters in the remaining positions of columns 2 through 7 are designated as belonging to the graphics left character group GL.

The control characters in columns 8 and 9, and position 15/15, are designated as belonging to the control character group C1. The graphic characters in the remaining positions of columns 10 through 15 are designated as belonging to the graphics right character group GR.

Figure B-19 shows how the character sets are designated and used in the MicroVAX workstation.

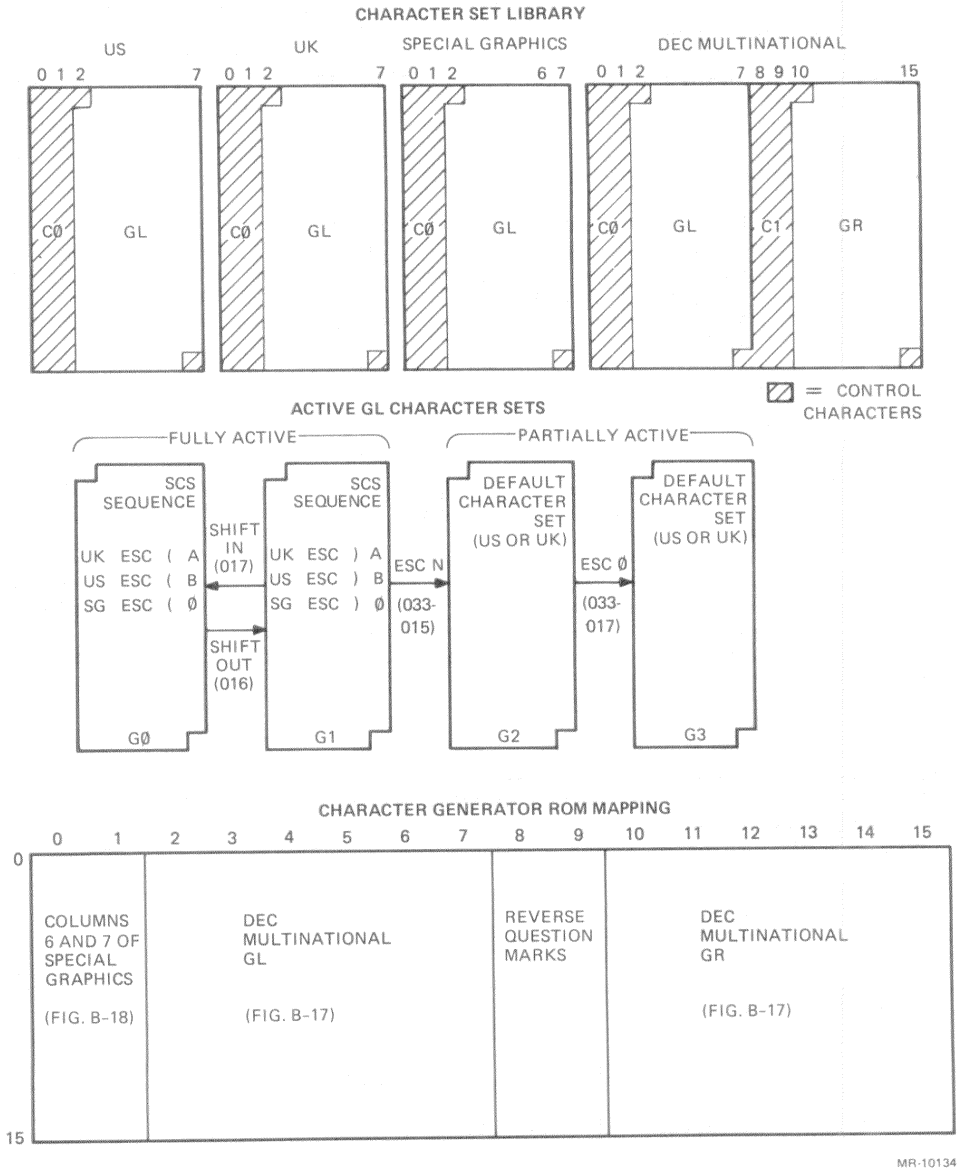


Figure B-19 Character Set Designations

B.8 CHARACTER SET SELECTION

A GL character set is selected by using select character set (SCS) sequences. SCS sequences are used to designate two GL character sets as fully active, and two GL character sets as partially active. Once designated, the two fully active GL character sets are selected with a shift-in or shift-out control character. Once a fully active GL character set is selected, all subsequent characters are assumed as belonging to that GL character set until an SCS sequence is again detected.

The two partially active GL character sets are selected with an escape N or escape O control character. Once a partially active GL character set is selected, only the following character is assumed as belonging to that GL character set. All subsequent characters are assumed as belonging to the previously selected, fully active GL character set.

If the MicroVAX workstation firmware does not detect an SCS sequence, it assumes that all characters belong to the default GL character set. The default GL character set is assigned or determined by the value of the twelfth bit that appears on the screen in the set-up mode, under parameter settings (PARAM SET). If this bit is 1, the default character set is UK; if it is 0, the default character set is USASCII.

The SCS sequences and their octal equivalents for selecting the fully active GL character sets (G0 and G1) and the partially active GL character sets (G2 and G3) are listed in Figure B-20. Note that both the G2- and G3-designated GL character sets will always be the default character sets determined in the set-up mode under parameter settings (PARAM SET).

CHARACTER SET	SCS SEQUENCE (NUMBERS IN SEQUENCE ARE IN OCTAL)			
	G0	G1	G2	G3
UK	ESC (A 033 050 101	ESC) A 033 051 101	ESC N 033 115 (IF UK SELECTED IN SET-UP MODE)	ESC O 033 117 (IF UK SELECTED IN SET-UP MODE)
USASCII	ESC (B 033 050 102	ESC) B 033 051 102	ESC N 033 115 (IF US SELECTED IN SET-UP MODE)	ESC O 033 117 (IF US SELECTED IN SET-UP MODE)
SPECIAL GRAPHICS	ESC (0 033 050 060	ESC) 0 033 051 060		

NOTE: ALL NUMBERS ARE IN OCTAL

MR-10135

Figure B-20 Character Set Selection

B.9 DISPLAYING CHARACTERS

All character codes are processed by the MicroVAX workstation firmware in the following order.

1. If no SCS sequences are detected, all character codes are assumed to belong to the default character set.

2. If an SCS sequence is detected, the character code is translated so that, when it is input to the character generator ROM, it produces the proper character display.

   NOTE

   A translation between the character code received and the character code expected by the character generator ROM is necessary because, as shown in Figure B-21, the ROM mapping for graphic characters does not always agree with the character set mapping in Figures B-17 and B-18.

3. All character codes with the eighth bit set are processed as belonging to Digital's multinational GR character set.

Figure B-22 shows the data paths in various modes. When character codes are input to the workstation application programs in console mode, the application software is responsible for transmitting the actual codes for each key pressed. If the escape key is pressed, the applications software must send the workstation firmware the character code for the escape key (27 decimal, 33 octal, or 18 hexadecimal). The same applies to host processor software, when the workstation is in line mode. In local mode, keyboard outputs are transmitted directly to the translation process firmware.

COLUMN		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
b8 b7 b6 b5 / b4 b3 b2 b1	ROW	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
0 0 0 0	0		– SCAN 3	SP	0	@	P	`	p		?		°	À	?	à	?
0 0 0 1	1		– SCAN 5	!	1	A	Q	a	q	?	?	¡	±	Á	Ñ	á	ñ
0 0 1 0	2		– SCAN 7	"	2	B	R	b	r	?	?	¢	²	Â	Ò	â	ò
0 0 1 1	3		– SCAN 9	#	3	C	S	c	s	?	?	£	³	Ã	Ó	ã	ó
0 1 0 0	4		┼	$	4	D	T	d	t	?	?	?	?	Ä	Ô	ä	ô
0 1 0 1	5		┤	%	5	E	U	e	u	?	?	¥	µ	Å	Õ	å	õ
0 1 1 0	6	◊	⊥	&	6	F	V	f	v	?	?	?	¶	Æ	Ö	æ	ö
0 1 1 1	7	±	┬	'	7	G	W	g	w	?	?	§	·	Ç	Œ	ç	œ
1 0 0 0	8	┘	│	(	8	H	X	h	x	?	?	¤	?	È	Ø	è	ø
1 0 0 1	9	┐	≤	)	9	I	Y	i	y	?	?	©	¹	É	Ù	é	ù
1 0 1 0	10	┌	≥	*	:	J	Z	j	z	?	?	ª	º	Ê	Ú	ê	ú
1 0 1 1	11	└	π	+	;	K	[	k	{	?	?	«	»	Ë	Û	ë	û
1 1 0 0	12	┼	≠	,	<	L	\	l	|	?	?	?	¼	Ì	Ü	ì	ü
1 1 0 1	13	─	£	-	=	M	]	m	}	?	?	?	½	Í	Ý	í	ý
1 1 1 0	14	├	·	.	>	N	^	n	~	?	?	?	?	Î	?	î	?
1 1 1 1	15	– SCAN 1	?	/	?	O	_	o	DEL	?	?	?	¿	Ï	ß	ï	
		SPECIAL GRAPHICS		GL						NOT USED		GR					

NOTE: REVERSE QUESTION MARKS (?) ARE POSITIONS RESERVED FOR FUTURE STANDARDIZATION

MR-10136

Figure B-21 Character Generator ROM Displayable Characters

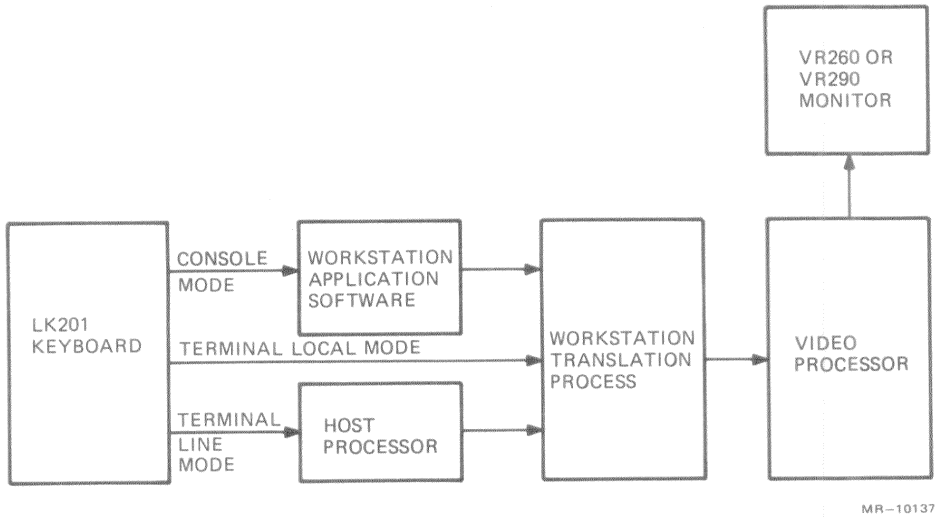


Figure B-22 Keyboard Output Processing

C.1 GENERAL DESCRIPTION

This Appendix provides information on Digital's serial mouse, shown in Figure C-1. A mouse is a hand-held pointing device used to move a cursor on a video display screen. As the mouse is moved about on a tabletop, the cursor makes corresponding movements on the screen. Once the cursor is positioned, one of the mouse buttons is pressed to make the desired selection.

Using an ergonomically designed pointing device like a mouse can make software easier to learn and use. Pointing is often faster, easier, and more accurate than with keyboard typing. Some advantages of using a mouse as a pointing device are

- Very fast positioning
- High accuracy
- Nonfatiguing
- Items on the screen are not obscured
- Stays where it was last placed
- Small, light-weight, and inexpensive.

Digital's serial mouse uses optomechanical technology to detect movement, with a resolution of 200 counts/in. A dedicated microprocessor inside the mouse converts quadrature X-Y pulses to relative X-Y coordinates, which are then transmitted as serial data to the host computer. A rubber-coated ball on the underside of the mouse provides excellent tracking on most desktop surfaces.

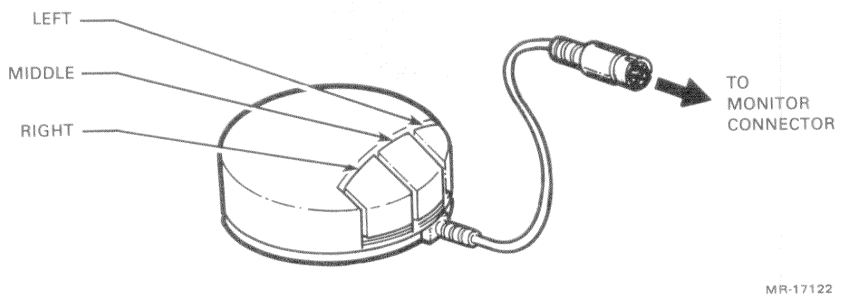


Figure C-1 Three-Button Mouse

C.2 INSTALLATION

C.2.1 System Hook-up

The mouse is shipped fully assembled and ready to operate. To connect the mouse, plug its cable into the mouse/tablet input jack of the computer or terminal on which it will be used. The mouse/tablet input jack is normally located on the rear panel of a video monitor.

C.2.2 Installing/Removing the Mouse Ball

The rubber-coated ball on the underside of the mouse can be removed for cleaning or replacement without special tools. To clean the ball, use water and, if necessary, a mild soap. Do not use organic solvents such as toluene; they damage the ball's rubber coating.

Clean the ball when the cursor fails to track the mouse smoothly. In an average office environment, cleaning the ball every six months should be sufficient.

To replace the mouse ball, turn the mouse upside down to locate the ball housing cover. Remove the cover by turning it counterclockwise, as shown in Figure C-2. To reassemble, insert the ball into its housing, replace the cover, and secure it by turning the cover in a clockwise direction.

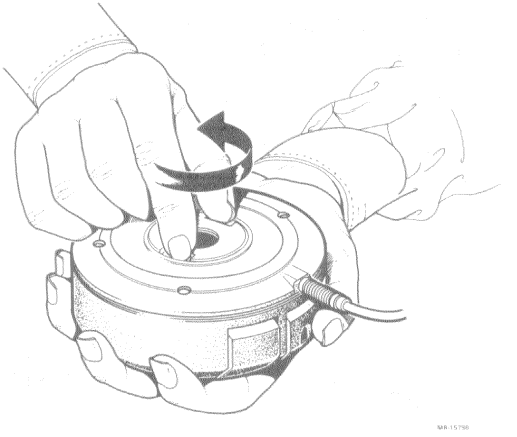


Figure C-2 Mouse Ball Removal

C.3 USING THE MOUSE

The mouse is normally used in contact with a desktop, to the right or left of the keyboard. To operate the mouse, take hold of it and allow your index finger to rest lightly on its buttons. Its cable should lead away from you. Move the mouse on the desktop to verify that the cursor responds correctly: it should go left when the mouse is moved left; right when the mouse is moved right; up when the mouse is moved away from you; and down when the mouse is moved toward you.

By experimenting, each user will find the most comfortable method of holding and moving the mouse. The mouse buttons may be either pressed from the front or down from the top. Most users find it easiest to make small movements of the wrist. Note also that the mouse can be picked up and repositioned at any time.

The buttons on the mouse are commonly referred to as left (L), middle (M), and right (R), as shown in Figure C-1. The functions of the mouse buttons depend on the software used. (Refer to applicable software documentation.)

C.4 MOUSE SPECIFICATIONS

PHYSICAL

Size	8.8 cm (3.5 in) diameter, 4.0 cm (1.6 in) high
Weight	170 g (6 oz) including cable
Switches	Three tactile-type switches, actuating force approximately 85 g (3 oz)
Cable	1.5 m (5 ft), round 0.375 cm (0.15 in) diameter, 6-conductor #26 AWG stranded, shielded high-flexibility design
Connector	7-pin micro-DIN type (male)
Temperature	Operating: +10°C to +40°C (+50°F to +104°F)
	Nonoperating: -40°C to +66°C (-40°F to +150.8°F)

ELECTRICAL

Power	+5 Vdc +/- 5% at less than 150 mA -8 V to -13 V at less than 20 mA
Interface	RS-232 voltage-level-compatible, output from the mouse capable of driving a load of 3 kilohms to ground (mark less than -6 V, space = 4.6 V min.)
FCC/EMI	Class B certified

PERFORMANCE

Resolution	0.125 mm (200 counts per inch)
Tracking Speed	0.76 m/s (30 in/s)
Accuracy	+/- 3% 0 - 25 cm/s (0 - 10 in/s), +/- 15% 25 - 50 cm/s (10 - 20 in/s), +/- 30% 50 - 75 cm/s (20 - 30 in/s) in any direction
Acceleration	0.5 g (5 m/s/s)

OPERATING

Modes	Incremental Stream Prompt (power-up default)
Report Format	Serial asynchronous (8 data bits, odd parity), report consists of 3 bytes indicating button state and relative displacement
Sampling Rate	55 reports/s in incremental stream mode, up to 95 reports/s when polling
Baud Rate	4800 baud

C.4.1 Signal/Power Cable

The connector wiring pin assignments and functions are listed in Table C-1.

The mouse cable male connector pins are numbered as shown in Figure C-3.

Table C-1 Mouse Connector Pin Assignments/Functions

Pin Number	Function
1	GND (signal and power return)
2	TXD (serial out from mouse)
3	RXD (serial in to mouse)
4	-12 V
5	+5 V
6	Not used
7	Device present (shorted to pin 1)
Shell	Protective ground

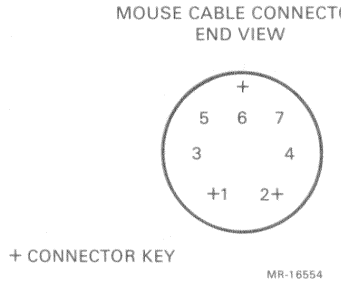


Figure C-3 Mouse Cable Connector Pin Numbering

C.4.2 Spurious Outputs

The mouse is designed to avoid reporting movement when it is not tracking on a surface. Lifting the mouse from a horizontal surface will not report movement of more than 20 counts of resolution.

Noise problems are suppressed where the output varies by one count when the mouse is not being moved.

C.5 ELECTRICAL INTERFACE

C.5.1 Interface Signal Levels

The mouse transmits and receives RS-232-compatible signals.

Transmit: Mark < -6 V, space > 4.6 V
Receive: Mark -15 V to 0.8 V, space 2.8 V to +15 V

Minimum dc load resistance is 3000 ohms to ground.

C.5.2 Power and Voltage Considerations

To operate, the mouse requires +5 V +/- 5% at 150 mA. On systems where a long extension cable is used between the mouse and its power supply, it may be necessary to regulate a higher voltage (5.2 V) at the system to overcome the voltage drop of the cable. Digital's local service representative should be consulted to determine the adequacy of the supplied voltage when adding a mouse extension cable.

There is no fuse inside the mouse housing due to its small size and low voltage requirements. If developing hardware to power the mouse, a fuse or other current-limiting method is recommended as a safety precaution.

The rise time of the +5 V supply should be less than 100 ms to ensure that power-up reset occurs.

C.6 MOUSE OPERATION

C.6.1 Serial Interface Operation

The mouse communicates with its host via an asynchronous, serial interface at 4800 baud (+/- 2%). Each character consists of one start bit, eight data bits, one parity bit (odd), and one stop bit. The mouse ignores incoming parity and the most significant bit of each byte (bit 7) on receive.

The mouse supports only half-duplex communication. If a byte is sent to the mouse while it is transmitting, the mouse will abort the data being transmitted (force a break) and process the new command immediately (except during self-test). If a byte is received between the characters of a multibyte report, the mouse is still considered to be transmitting and will abort the current report.

C.6.2 Report Format

Data is transferred in 9-bit bytes (8 data, plus odd parity). Though the mouse transmits odd parity, it ignores parity errors on receive. The mouse transmits a 3-byte position report, as shown in Figure C-4.

The X and Y values in the mouse position report give the movement in units of resolution since the last report. If the X or Y values overflow, the maximum movement is reported.

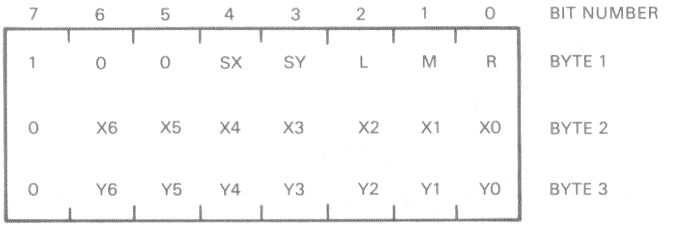


Figure C-4 Three-Byte Position Report Format

C.6.3 Operating Modes

The mouse has two operating modes that determine when, and how often the mouse transmits a position report:

1. Incremental Stream Mode
2. Prompt Mode (power-up default).

In incremental stream mode, the mouse generates reports at 55 Hz intervals any time there is movement, or a change in button position since the last report. If the mouse is motionless and no buttons have changed, no report is generated. The report rate under continuous movement or button change is 55 reports/s.

In prompt mode, the mouse generates a report only in response to a request mouse position command. The mouse responds to over 95 position requests/s.

C.6.4 Summary of Mouse Commands

All mouse commands are printable ASCII characters and are summarized in Table C-2. The mouse ignores invalid commands.

Table C-2 **Mouse Command Summary**

ASCII	Hex	Function
R	52	Select incremental stream mode
D	44	Select prompt mode
P	50	Request mouse position -- The mouse responds with a position report. The P command also switches the mouse to prompt mode.
T	54	Self-test and identify -- The mouse responds with its self-test report (see Paragraph C.6.5). Self-test leaves the mouse in the reset or power-up state. When a self-test command is issued, it is invalid to send any data to the mouse until the last byte of the self-test report is received. The mouse ignores any data received during self-test.
Zx	5A XX	Reserved functions for testing or quality control that are completed within one second. The previous mouse state is not disturbed.

C.6.5 Power-Up Self-Test and Identification

Upon command from the computer or mouse power-up, the mouse automatically checks its internal logic and circuits, and transmits a self-test report consisting of a 2-byte identification code and a 2-byte code describing the health of the electronics and firmware.

The 4-byte self-test report shown in Figure C-5 is transmitted at power-up.

The bytes shown in Figure C-5 are described below.

Byte 1, Firmware ID: Bit 7 through bit 5 (101) indicates the start of a self-test report. R3-R0 is the revision number.

Byte 2, Hardware ID: A single-byte code for manufacturing and device identification. M2-M0 is Digital's manufacturing location ID. Bit 3 through bit 0 is the device code; 0010 indicates mouse data. Figure C-6 shows the ID code format.

Manufacturing location ID assignments are reserved for Digital's future definition.

Device ID assignments:

0010 - Indicates mouse data
0100 - Indicates tablet data

Byte 3, Error Code or Zero: Error code ASCII ">" (3E hex) indicates a RAM or ROM checksum error. If there is no checksum error, ASCII "=" (3D hex) indicates a button error. Codes of 20 (hex) or greater are considered to be fatal hardware problems, codes less than 20 (hex) are nonfatal.

Byte 4, Button code or Zero: The code will be the same as the low three bits of of the first byte of the mouse position report, indicating which, if any, buttons are down or have failed.

Example: 04 (hex) would mean a left button error.

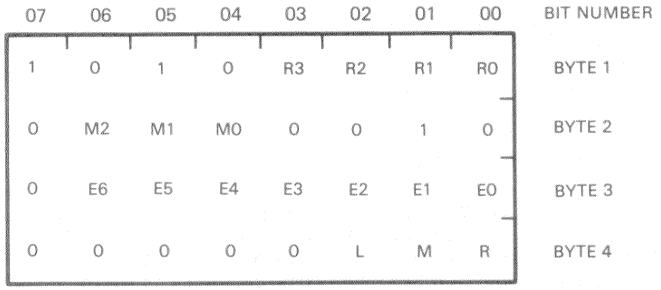


Figure C-5 Four-Byte Self-Test Report Format

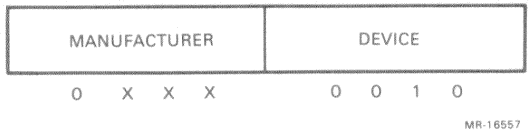


Figure C-6 ID Code Format

The switches in the mouse have two contacts, and thus, four possible states:

1. Button up
2. Button down
3. Both contacts open (switch missing)
4. Both contacts closed (short circuit).

The left and right switches report a button error for any state other than button up. The middle switch reports an error for button down, or both contacts closed.

The mouse ignores any data received during self-test until the last byte of the self-test report is transmitted.

C.6.6 Report Synchronization

Bit 7 of the first byte of a multibyte report is 1, and can be used for software synchronization. Bit 7 of the succeeding bytes of a multibyte report are always 0s.

The following starting bit sequences are defined (bit 7, bit 6, and bit 5 of the first byte of a multibyte report).

100 - 3-byte relative position report (mouse)
101 - 4-byte self-test report (mouse or tablet)
110 - 5-byte absolute position report (tablet)
111 - Reserved for Digital's future definition

C.6.7 Response Time

The mouse completes all valid commands, except self-test and Digital's reserved functions, within 50 ms. It processes a self-test command within 500 ms. At power-up, the mouse completes its self-test within 1 second from stable power.

C.7 PROGRAMMING CONSIDERATIONS

Digital's serial mouse does not support the XON/XOFF flow control protocol because there is no data buffering inside the mouse. This results in a more effective link between mouse and cursor.

The serial mouse uses half-duplex communication. Half-duplex does not interfere with normal mouse operation because there is no reason to send a command to the mouse while it is transmitting. When polling, the previous 3-byte report is always read before requesting another report. The only other commands to the mouse cause it to reset or change operating modes.

The mouse powers up in prompt mode to avoid sending position reports to the host before it is ready to receive them. The mouse's self-test report is transmitted at power-up to notify the host of its presence.

C.7.1 Initialization

Although Digital's serial mouse defaults to prompt mode at power-up, host software should select the protocol and operating mode explicitly for compatibility with other mouse devices. A complete power-up sequence typically includes the following steps.

1. Initialize the host's serial port for the mouse for 4800 baud (8 data, odd parity, 1 stop bit).

2. Read the power-up self-test report to confirm the mouse's identification and function. If the self-test was successful, go to step 4.

NOTE

The self-test report may indicate the pointing device is a digitizing tablet. (See Appendix D.)

3. If the self-test report is not received in 1 second:

   Check that the mouse is correctly attached and send ASCII T (54 hex) to invoke self-test. If the mouse responds, check identification and function. If there is no response, assume a defective or alien mouse.

4. Send ASCII S (53 hex) to select the report format. (Currently, there is only one format.)

5. Send ASCII R (52 hex) to select incremental stream mode, or ASCII D (44 hex) to select prompt mode (power-up default).

C.7.2 Incremental Stream Mode vs. Prompt Mode

The mouse has two operating modes that determine when, and how often the mouse transmits a position report.

In incremental stream mode, the mouse tests for movement or button action 55 times/s, and sends a report only when a significant event occurs. This mode allows the host to track the cursor smoothly, with a minimum of data overhead.

An effective way to use incremental stream mode is to set the alarm level on the universal asynchronous receiver transmitter (UART) to interrupt only when its first-in/first-out (FIFO) buffer is full. On each vertical interrupt, the host polls the UART for available data. If data is waiting, the host reads the report and updates the cursor position. Otherwise, no action is necessary. The host rarely receives UART interrupts.

The mouse can also be polled to allow the host to control when the mouse position is updated. Since the buttons are not latched (only debounced), the host must poll frequently enough to detect any button action.

The intended use of polling is to synchronize mouse updates with video refresh in order to minimize interrupt overhead. On each vertical interrupt, the host reads the previous 3-byte report from the mouse UART, and then sends the command to poll the mouse. If the UART is buffered (as many are), allowing it to hold three bytes, the host never needs to respond to mouse interrupts.

To move the cursor absolutely smoothly under worst-case conditions, it is necessary to update the cursor position every frame time. For normal 60 Hz refresh, the 55 Hz sampling rate is sufficient to provide good tracking. For the smoothest possible tracking, or higher frame rates, polling at the frame refresh rate is recommended.

C.7.3 Button Use

Mouse buttons are commonly referred to as left, middle, and right. The left button is frequently used for the "select" function, and most closely corresponds to the single button available on other mouse devices.

For 2-button applications, the recommended convention is to treat the right and middle buttons as identical. The left and right buttons are used for compatibility with 2-button documentation. The left and middle buttons are used for comfort and easy identification by touch.

C.7.4 Tablet Support

By adding minor extensions to the mouse software, the mouse input port can accommodate Digital's serial tablet as well. For detailed information on the tablet, refer to Appendix D. With appropriate device drivers, it is possible to swap the mouse and tablet transparently while an application is in progress. The tablet uses the same connector and serial interface format as the mouse, and recognizes the same commands (with some additions to accommodate its extended capabilities).

The tablet's power-up self-test and identification sequence uses the same format as the mouse's, but some values differ. Device code 0100 is used to indicate tablet data. By convention, tablet error codes (byte 3 of the self-test report) indicate success for values less than 32 (20 hex), and error for values of 32 or greater.

The tablet transmits a 5-byte absolute position report, with the starting bit sequence 110 used for synchronization.

Since the tablet is an absolute pointing device, it is generally undesirable for applications to move the cursor away from the position last reported by the tablet. For compatibility with the tablet, applications may wish to avoid repositioning the cursor when using the mouse.

While the mouse has three buttons, the digitizing tablet can use either the 4-button puck or the 2-button stylus. The tablet puck has its buttons arranged in the form of a diamond. (See Appendix D.) By convention, puck button B1 relates to the mouse's left button, puck button B2 to the mouse's middle button, and puck button B3 to the mouse's right button. The tablet puck's button B4 is ignored. When using the 2-button stylus, the barrel button B1 corresponds to the mouse's left button, and the tip button B2 corresponds to the mouse's right button.

APPENDIX D
DIGITIZING TABLET SPECIFICATION

D.1 GENERAL DESCRIPTION

This Appendix provides the specifications that define the characteristics of a digitizing tablet system interfacing with a VAXstation II/GPX workstation. The digitizing tablet system consists of a 27.5 cm (11 in) square digitizing tablet, a 4-button puck, a 2-button stylus, and a 5-foot power/signal cable, as shown in Figure D-1.

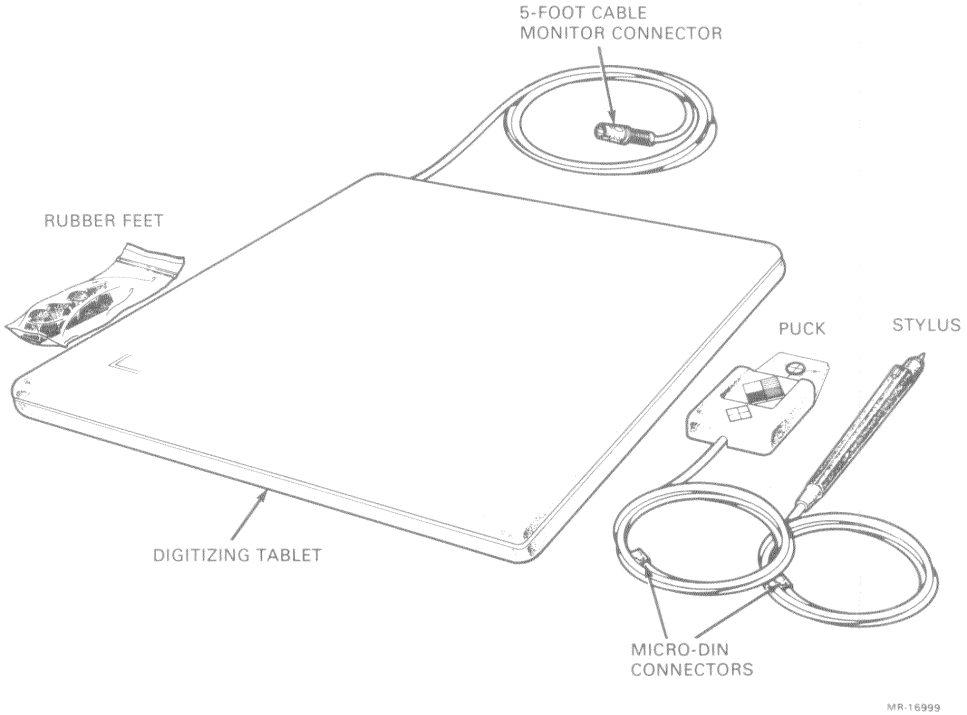


Figure D-1 Digitizing Tablet

The digitizing tablet is a computer input device that sends X-Y coordinates to a computer in order to indicate the position of a stylus or puck on the tablet's surface. The tablet has a resolution of 0.0125 cm (0.005 in) or 200 counts/in. The active area of the tablet is 27.5 cm (11 in) square. The 4-button puck and the 2-button stylus pen use the same connector at the rear of the tablet, with only one connected at any given time. The tablet cable is supplied with a 7-pin micro-DIN type connector (male), which is assigned the signals listed in Table D-1. A 5-foot, 5-conductor, flexible signal/power cable connects the tablet to the MicroVAX workstation system. Figure D-2 shows the tablet cable connector numbering scheme.

Table D-1 Tablet Cable Connector Signal Assignments

Connector Pin	Signal Assignment
1	GND
2	TXD (serial out from tablet)
3	RXD (serial into tablet)
4	Not used
5	Not used
6	+12 V
7	Device present (connected to pin 1)
Shell	Protective ground (ESD shield)

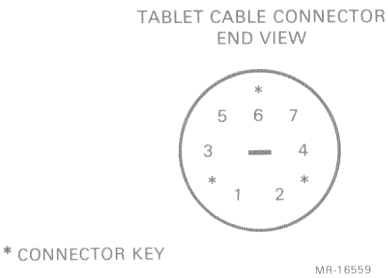


Figure D-2 Tablet Cable Connector Numbering Scheme

D.2 ELECTRICAL SPECIFICATIONS (POWER RATING)

The electrical requirements of the tablet are +12 Vdc +/-10% at less than 300 mA.

D.3 COMMUNICATION SPECIFICATIONS

D.3.1 Serial Interface

The tablet communicates with its host via an asynchronous, full-duplex, serial interface at 4800 baud (+/- 2%) and 9600 baud (+/- 2%). Bytes are framed by one start bit and one stop bit. The data byte contains 8 bits of data and 1 parity bit (odd parity). The default baud rate is 4800 bits/s.

D.3.2 Electrical Signals

The tablet transmits and receives RS-232-C-compatible signals as follows.

```
Transmit: Space = +5 V to +12 V,  mark = -4 V to -12 V
Receive:  Space = +3 V to +12 V,  mark = -3 V to -12 V
```

Output from the tablet is capable of driving a load of 3000 ohms to ground.

D.3.3 Tablet Position Report

The tablet transmits a 5-byte position report format as shown in Figure D-3.

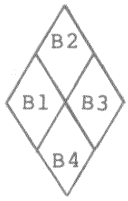
In explanation of Figure D-3:

Bit 7 = Start of frame synchronization

Puck:

B1 = Button #1: 0 = up, 1 = down
B2 = Button #2: 0 = up, 1 = down
B3 = Button #3: 0 = up, 1 = down
B4 = Button #4: 0 = up, 1 = down

Puck
Button Layout



Stylus:

B1 = Barrel button: 0 = up, 1 = down
B2 = Tip button: 0 = up, 1 = down
PR = Proximity: 0 = in proximity,
1 = out of proximity

X0-X11 = X coordinate bits; X0 is LSB, binary format.
Y0-Y11 = Y coordinate bits; Y0 is LSB, binary format.

BYTE	BIT 7	6	5	4	3	2	1	0
1	1	1	0	B4	B3	B2	B1	PR
2	0	0	X5	X4	X3	X2	X1	X0
3	0	0	X11	X10	X9	X8	X7	X6
4	0	0	Y5	Y4	Y3	Y2	Y1	Y0
5	0	0	Y11	Y10	Y9	Y8	Y7	Y6

MR-16560

Figure D-3 Tablet Position Report Binary Format

D.4 TABLET OPERATION AND COMMANDS

D.4.1 Report Rate

The tablet report rate is software-selectable by host command. Report rates are 55, 72, and 120 reports/s. In order to transmit 120 reports/s, the baud rate must be increased to 9600 bits/s. The default or power-up report rate is 55 reports/s. Report rate is selected by sending the following ASCII characters.

K	(4B hex)	Sampling rate of 55 reports/s
L	(4C hex)	Sampling rate of 72 reports/s
M	(4D hex)	Sampling rate of 120 reports/s

D.4.2 Baud Rate Command

The default baud rate of the tablet is 4800 bits/s. The baud rate is changed to 9600 bits/s by sending the ASCII character B (42 hex) to the tablet. Upon receiving this command, the tablet completes any report in progress, change baud rate, and switches to request point mode.

The tablet is switched back to the default baud rate by sending a BREAK (minimum of two character times) or by requesting a self-test of the tablet. Both these commands invoke the self-test and return all functions to the default conditions.

D.4.3 Request Point Mode

Request point mode allows the host to select when a coordinate pair is to be sent. This mode is selected by sending the ASCII character D (44 hex). A coordinate pair is requested by sending the ASCII character P (50 hex) to the tablet. The P command also switches the tablet to request point mode. If the request is made while the stylus or puck is not in proximity with the tablet, the last valid coordinate pair is transmitted without the off-proximity flag being set. Upon completion of power-up or self-test, the tablet is placed in this mode.

D.4.4 Incremental Stream Mode

In incremental stream mode, the tablet generates a report when the puck or stylus is moved more than 0.0125 cm (0.005 in). Reports continue while the puck is in motion. A report is also generated upon button depression or release. Incremental stream is selected by sending the ASCII character R (52 hex) to the tablet.

D.4.5 Self-Test

Upon command from the host computer or upon tablet power-up, the tablet automatically checks its internal logic circuits, and transmits a 2-byte identification code, plus a 2-byte code describing the health of the electronics and firmware.

A 4-byte self-test report is transmitted at power-up, as shown in Figure D-4. Each report contains the absolute X and Y positions of the puck or stylus from the origin of the tablet. The origin of the tablet is the bottom left corner of the active area. The first byte of the report indicates which button is pressed.

07	06	05	04	03	02	01	00	BIT NUMBER
1	0	1	0	R3	R2	R1	R0	BYTE 1
0	M2	M1	M0	0	0	1	0	BYTE 2
0	E6	E5	E4	E3	E2	E1	E0	BYTE 3
0	0	0	B4	B3	B2	B1	PR	BYTE 4

KEY:
BIT 7 = FRAME SYNCHRONIZATION
R3-R0 = REVISION NUMBER
M2-M0 = MANUFACTURING LOCATION ID
E6-E0 = ERROR CODE
B4-B1 = BUTTON CODE
PR = PROXIMITY

MR-17121

Figure D-4 Four-Byte Self-Test Report Format

The four bytes are explained below.

Byte 1, Firmware ID: A0 (hex) for initial release.

Byte 2, Hardware ID: A single-byte code for manufacturing and device identification, as formatted in Figure D-5.

Manufacturing location ID code assignments are reserved for Digital's future definition.

Device ID code assignments:
0010 - Indicates mouse data
0100 - Indicates tablet data

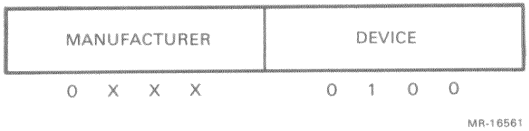


Figure D-5 ID Code Format

Byte 3, Hardware: Status	This byte states the health of the tablet's circuitry. Codes of 20 (hex) or greater are considered to be fatal hardware problems, codes less than 20 (hex) are successful. This byte will be one of the following codes. 3E (hex) - RAM or ROM checksum error 3A (hex) - Analog or digital error 3D (hex) - Button down error 13 (hex) - No puck or stylus connected 11 (hex) - Stylus is connected 00 (hex) - Self-test passed
Byte 4, Button Code: or Zero	The code is the same as the first byte of the 5-byte code of the tablet position report. It indicates which buttons are down, if any. Bits 6 and 7 of this byte are 0s.

The self-test command leaves the tablet in the reset or power-up state. When a self-test command is issued, it is invalid to send any data to the tablet until the last byte of the self-test report is received. The tablet ignores any data received during self-test.

The host commands for self-test are the ASCII character T (54 hex) or a BREAK (minimum of two character times).

D.4.6 Default Conditions

Upon completion of power-up or after a self-test command, the tablet is placed in the following state.

- Request point mode
- Baud rate of 4800 bits/s
- Report rate of 55 reports/s

D.4.7 Report Synchronization

Multibyte reports are atomic (the bytes of the report are sent in sequence without interruption). Bit 7 of the first byte of a multibyte report is a 1, and can be used for software synchronization. Bit 7 of the succeeding bytes of a multibyte report are always 0s.

When a command is received, the tablet completes the current report transmission before executing the new command.

D.4.8 Recovery from Invalid Commands

The tablet ignores invalid commands.

D.4.9 Summary of Digitizing Tablet Commands

Table D-2 lists the commands that can be executed using the digitizing tablet. All tablet commands are single-byte, printable ASCII characters.

Table D-2 Summary of Tablet Commands

ASCII	Hex	Function
BREAK		Invoke self-test
B	42	Change baud rate to 9600
D	44	Request point mode command
K	4B	Set sampling rate to 55
L	4C	Set sampling rate to 72
M	4D	Set sampling rate to 120
P	50	Position request command
R	52	Incremental stream
T	54	Invoke self-test
Zx	5A XX	Digital's reserved functions for testing or quality control, completed within 1 s. The previous tablet state is not disturbed.

D.5 PERFORMANCE SPECIFICATIONS

D.5.1 Resolution

The resolution of the tablet is 0.0125 cm (0.005 in) or 200 counts/in in any direction.

D.5.2 Accuracy

The tablet does not produce outputs in error by more than +/- 5 counts of the position report's LSB.

D.5.3 Spurious Outputs

The tablet avoids reporting movements of 1 LSB or when reporting has been disabled. Picking the puck or stylus straight up from the tablet surface does not report movement of more than 10 counts of resolution. When the puck or the stylus is not in the proximity of the tablet, positional reports to the host are suppressed.

D.5.4 Response Time

The tablet processes all valid commands (except self-test and vendor-reserved commands) within 100 ms. Within 500 ms, the tablet must respond to a self-test command. The tablet completes its power-up self-test within 1 s after stable power.

D.5.5 Initialization

A sample initialization for the tablet is as follows.

1. Initialize the host's serial port for the mouse/tablet to 4800 baud data format (8 data, odd parity, 1 stop bit).

2. Request that the tablet do a self-test and identification sequence by sending ASCII T (54 hex) to the tablet.

   If the tablet driver switches the baud rate to 9600 baud, the break character should also be used (break followed by an ASCII T). This prevents the tablet's staying at the higher baud rate if the system is reset. Care must be taken when switching to the higher baud rate.

3. The tablet replies with the self-test and identification sequence. Device identification should be checked for device type, along with any error report. If self-test was successful, go to step 5.

4. If self-test report was not received within 1 s:

   - Check that the tablet is correctly attached.
   - If no response is received, the tablet is either defective or alien.

5. The system now should set the tablet to the proper operating mode. Default set-up is for prompt mode.

D.6 ENVIRONMENTAL SPECIFICATIONS

TEMPERATURE RANGE	
Operating	+10°C to +40°C (+50°F to +104°F)
Nonoperating	-40°C to +66°C (-40°F to +150.8°F)
HUMIDITY	
Operating	10% to 90% relative humidity, noncondensing, with a maximum wet bulb temperature of 28°C (82.4°F) and a minimum dew point of 2°C (35.6°F).
Nonoperating	5% to 95% with a maximum wet bulb temperature of 46°C (82.8°F).
ALTITUDE	
Operating	2.4 km (8,000 ft) maximum for 4 hours minimum.
Nonoperating (packaged)	9.1 km (30,000 ft) maximum for 1 hour minimum.

Address processor chip
A custom LSI chip (microsequencer) that generates the CRT timing signals and bitmap memory addresses required for screen refresh, scrolling, and raster operations. The address processor chip also supplies microinstructions and data used by the video processor chip (datapath).

Application program
A program designed to meet a user's specific need, such as the monitoring of a manufacturing process.

ASCII
American Standard Code for Information Interchange.

Asynchronous multiplexer
A device that controls and alternates the transmission of signals so that more than one signal can be transmitted over a single communication line. The data being transmitted contains the necessary synchronizing information in the form of start and stop signals.

Backplane
The connector block into which all the MicroVAX workstation's printed circuit boards are inserted.

Backup
The continuing process of making copies of a disk drive's stored data for purposes of recovery in case the main record is lost. Backups are made on RX50 diskettes or TK50 magnetic tape cartridges, which are then safely stored.

Baud rate
The speed at which signals are serially transmitted along a communication path. One baud equals one bit/s.

Binary
The number system that uses only 2 digits: 0 and 1. These digits are usually represented in MicroVAX workstation circuitry by 2 voltage levels.

Bit
A binary digit; the smallest unit of information in a binary system of notation, designated as a 0 or a 1.

Bitmap
Memory used to store an image as an array of bits, one bit for each pixel. Bitmaps can be physically overlayed to form multivalued pixels.

Bootable medium
A fixed disk, diskette, or magnetic tape cartridge that contains software, such as an operating system, which the bootstrap program can load into the MicroVAX workstation's system memory and begin memory execution.

Bootstrap program
A program started upon power-up of the MicroVAX workstation. The program loads into memory software contained on a fixed disk, diskette, or magnetic tape cartridge. The workstation then halts execution of the bootstrap program and starts execution of the software in memory. The software usually loads an operating system or other software into memory to enable operation of the MicroVAX workstation.

Bug
An error in the design or implementation of hardware or software system components.

Bus
A flat, flexible cable that consists of many transmission lines or wires and that interconnects computer system components to provide communication paths for addresses, data, and control information.

Byte
A group of 8 binary digits (bits). A byte is one-quarter the size of a MicroVAX workstation word (that is 32 bits, 4 bytes, or 2 words, or 1 longword).

Central processing unit (CPU)
The part of a MicroVAX workstation system that controls the interpretation and execution of instructions.

Clipping rectangle
A rectangular mask applied to the destination raster during a rasterop. Pixels outside the boundary of the clipping rectangle are unmodified.

CMR
Color map register.

Color map
A random-access memory that translates a pixel value into a color or intensity.

Command
An order entered on the MicroVAX workstation keyboard by a user.

Communication line
A cable path along which electrical signals are transmitted.

Computer system
A combination of, for example, MicroVAX workstation hardware, software, and external devices which performs specific operations or tasks.

Console terminal
The MicroVAX workstation terminal used for installing software and running diagnostic programs.

Controller
A MicroVAX workstation component, usually a printed circuit board, that regulates the operation of one or more peripheral devices. Controllers are often called "interface units."

Control panel
The part of the desktop, floorstand, or rack-mounted BA23 or BA123 enclosure that contains the control switches and indicators for monitoring the operation of the MicroVAX workstation.

CPU
Central processing unit.

Cursor
An arbitrary 2-color shape with an arbitrary point; a blinking or non-blinking block or underscore displayed to indicate the screen or data entry position.

DAC
Digital-to-analog converter.

Data
A formalized representation of facts, concepts, or instructions, suitable to human or mechanical communication, interpretation, or processing.

Data transmission
The movement of data in the form of electrical signals along a communication line.

Debug
To detect, locate, and correct errors (bugs) in a system's hardware or software.

Device
The general name for any entity connected to the MicroVAX workstation that is capable of receiving, storing, or transmitting data.

Diagnostic program
A program or several programs that detect and identify abnormal MicroVAX workstation hardware operation.

Directed-beam refresh
See "Graphics CRT display."

Direct-view storage tubes
See "Graphics CRT display."

Disk
A flat, circular platter with a coating on which data is magnetically stored in concentric circles (tracks); for example, an RD53 fixed disk, and RX50 disk(ette).

Disk drive
A device that contains a fixed disk or one or more diskettes. The drive contains mechanical components that spin the disk(ettes) and move the pickup heads for reading and writing data.

Diskette
A flexible disk packaged in a square paper envelope.

Display coordinates
The address of a pixel in Cartesian coordinates referenced to the upper left corner of the display. X increases to the right, Y increases downward.

DMA
Direct memory access.

DUART
Dual universal asynchronous receiver/transmitter.

EIA
Electronic Industries Association.

Error message
A message displayed by the MicroVAX workstation to indicate detection of (a) hardware, software, or data fault(s).

Fast mode
The writing mode in which data is written into the destination raster in 16-bit increments. Fast mode can be used when one edge of the raster is aligned with the X-axis.

FCN registers
Logic function registers in the video processor chip that can be coded to perform logical operations on displayable data.

FIFO
First-in/first-out (buffer control).

File
A collection of related information treated as a single item by MicroVAX workstation.

Formatted data
Data arranged in a particular pattern to conform to a predetermined structure or architectural protocol. Its organization is dictated by the system software.

FRU
Field-replaceable unit, an individual component or assembly of a system. An FRU can be replaced while the system is operational. The VCB02 video subsystem's FRUs are the VCB02 base module, the first VCB02 4-plane memory upgrade module, the second VCB02 4-plane memory upgrade module, the keyboard, and the mouse or optional tablet.

Graphics CRT display
One of three basic forms of graphics CRT displays: raster scan, directed-beam refresh, or direct-view storage tubes.

The raster scan form is used in commercial television. It operates by varying the intensity of a beam that periodically scans left to right and top to bottom along the fixed number of a screen's scan lines.

In directed-beam refresh displays, the beam moves from point to point on the screen to produce the desired image, rather than scanning periodically. The image is maintained by being completely redrawn at the periodic refresh rate.

Direct-view storage tube devices use the same method as directed-beam displays to draw an image. Since the image is stored in the phosphor of the screen, periodic refresh is not necessary. This type of display features a "write through" capability, which writes a beam-refreshed image of lesser intensity over the permanent image on the screen.

Full color is available only on the raster scan display devices.

Hard-copy terminal
A terminal that displays information on paper, rather than on a video monitor.

Hardware
The physical (mechanical and electrical) components assembled in a MicroVAX workstation.

Head
The part of a fixed disk drive, diskette drive, or tape drive that reads, records, and erases data.

Hole
A pixel within the boundaries of a destination raster that is not mapped by the normal raster operation algorithm.

Hole-fill
A modification of the raster operation algorithm used to fill holes in the destination raster.

I/D interconnect
The instruction/data bus that connects the address processor chip to one or more video processor chips and associated display hardware; used to transfer instructions and data.

Input device
A system assembly used to transfer data into the MicroVAX workstation; (for example, a keyboard).

Input/output (I/O) device
A system assembly that accepts data for transmission to (input) or from (output) a MicroVAX workstation; for example, a terminal.

Interactive
The method of communicating with a MicroVAX workstation system in which a command is entered at the keyboard, the system executes the command, and indicates its accomplishment of the command with a displayed screen prompt.

Interface
A hardware device or software instruction that allows the components of a MicroVAX workstation to communicate with one another.

I/O
See "Input/output device."

Kbyte
Abbreviated form of "kilobyte;" 1,024 bytes.

LEDs
Light-emitting diodes used as indicators on the control panel and vital elements of a printed circuit board.

Linear pattern
A repeating, 2-dimensional, rectangular pattern whose origin and alignment correspond to an arbitrary vector in the bitmap.

LK201
One of Digital's LK200-series keyboards.

Load
To move software from a peripheral device into system memory; to physically place a disk into a disk drive, or tape into a tape drive.

Local processor
The processor that controls the VCB02 video subsystem.

Locator
An absolute point on the display represented as (X, Y), with the X coordinate in the high 16 bits, and the Y coordinate in the low 16 bits.

LSB
Least significant bit.

Magnetic tape
A long plastic strip coated with magnetic oxide, used for storing data; for example, the tape contained in a TK50 magnetic tape cartridge.

Mbyte
Abbreviated form of "megabyte;" 1,048,576 bytes.

Memory
The area where a MicroVAX workstation stores data and finds the instructions to perform commanded tasks.

Memory coordinates
The address of a pixel in Cartesian coordinates referenced to the start of bitmap memory. X increases to the right; Y increases downward.

Menu
A displayed list of options, typically commands a user can enter.

OCRs
Operand control registers in the video processor chip that can be coded to control the transfer of displayable data between functional registers in the video processor chip and the bitmap or I/D interconnect.

On-line
Pertaining to equipment, devices, and events that are in direct communication with the MicroVAX workstation system.

Operating system
A collection programs that control the operation of the MicroVAX workstation and perform such tasks as:

- Data and program memory assignments
- Processing requests
- Scheduling tasks
- Controlling input/output operations.

Output device
A device that extracts data from the MicroVAX workstation; for example, a printer.

Peripheral device
Any device that provides the CPU with added memory storage or communication capability; for example, a disk(ette) drive, video terminal, printer.

Pixel
A picture element, the smallest addressable component of a displayable image. Associated with each pixel is a value, and a bitmap address in a 2-dimensional Cartesian coordinate system.

Pixel field
The bits in the X direction that represent a pixel. At highest resolution, the field is one bit wide. The width is two, or four, bits at half, or one-quarter, resolutions, respectively.

Pixmap
An N-plane (pixel) rectangle, where N is the number of planes provided by a display.

Power-up sequence
A series of ordered events that functions to properly energize a system.

Printer
A peripheral device that provides paper copies of information stored in a MicroVAX workstation.

Program
The complete sequence of instructions necessary for a MicroVAX workstation to perform a task.

PROM
Programmable read-only memory.

Prompt
A symbol or word(s) the MicroVAX workstation displays to indicate a task is completed or that a command should be entered to perform one.

Puck
A 4-button input device used with an optional digitizing tablet.

Raster
The set of pixels within a parallelogram.

Rasterop
A raster operation on the contents of bitmap memory that transforms a rectangular source of pixels into a destination raster of any size or orientation.

Raster scan
See "Graphics CRT display."

Read-only memory (ROM)
A memory that does not allow modification of its contents.

Reboot
To restart a MicroVAX workstation system. Pressing the restart button on the control panel reboots the workstation system if the halt enable/disable switch at the rear of the enclosure is in the up (enable) position.

Record
A set of related data that a program can treat as a unit. A "file" consists of a number of records.

ROM
Read-only memory.

Run
A single, continuous execution of a program.

Slow mode
The writing mode in which data is written into the bitmap one pixel at a time. Slow mode is used for Z-axis operations and when the destination raster does not have one edge aligned with the X-axis.

Software
Programs executed by a MicroVAX workstation system to perform chosen or required functions.

Software package
A set of related programs that together perform a specific task.

Storage medium
Any device capable of recording information; for example, a diskette.

Store
To enter data into a storage device, such as a disk drive, or into memory.

Subplane
The bit plane that is created by reducing the X resolution of a bitmap by a factor of one-quarter or one-half. In this case, X resolution is sacrificed to obtain added color or intensity levels.

System
A combination of, for example, MicroVAX workstation hardware, software, and external devices that performs specific processing operations.

System management
Tasks performed by the operating system that control the overall operation of the MicroVAX workstation.

Terminal
Generally, an input/output device that allows a user to communicate with the MicroVAX workstation system. Terminals are either video or hard-copy devices.

Tile
A linear pattern whose origin coincides with the screen origin and whose edges are parallel to the X- and Y-axis. Conceptually, a tile pattern covers the entire bitmap address space.

VCB02 video subsystem
A 4- or 8-plane video subsystem contained on 2 or 3 quad-height modules, respectively. It provides 1024 (horizontal) by 864 (vertical) pixel resolution on a 19-inch monochrome or color monitor when installed in a MicroVAX workstation (BA23 or BA123 enclosure). The 4-plane subsystem displays a total of 16 colors or shades of gray, whereas the 8-plane subsystem displays a total of 256.

Video terminal
A terminal that displays information on the screen of a cathode ray tube (CRT).

Video processor chip
The component (data path) of the VCB02 video subsystem that operates on displayable data as instructed by the address processor chip.

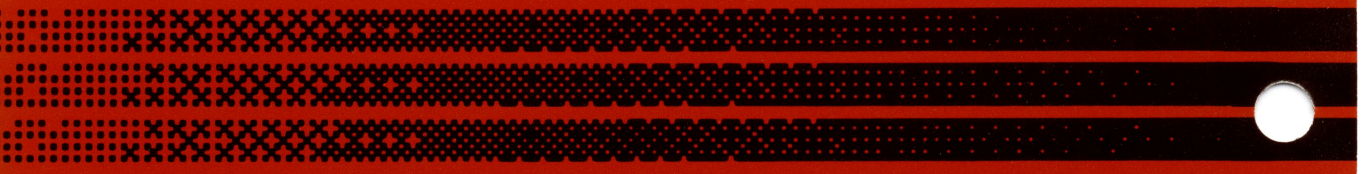
Winchester disk
A hard disk permanently sealed in a drive unit to prevent contaminants from affecting the read/write head.

Word
The largest number of bits (32) that a MicroVAX workstation can accommodate in any one operation. The MicroVAX workstation can also process longwords (two words, or 64 bits).

Workstation
A single-user system that offers high-performance, high-resolution graphics, and can function in a network environment.

Write-protect
To protect a disk, diskette, or other storage medium against the addition, revision, or deletion of its information.

Write-protect notch
The small notch in the side of an RX50 diskette envelope, covered with an adhesive label or tab to write-protect the diskette.



Digital Equipment Corporation • Maynard, MA 01754