DRAGON VIDEO SYSTEM
HARDWARE SPECIFICATION

June 25, 1985
Revision 4.2

DRAGON VIDEO SYSTEM HARDWARE SPECIFICATION

June 25, 1985
Revision 4.2

COMPANY CONFIDENTIAL

RESTRICTED DISTRIBUTION - DO NOT REPRODUCE

The specifications contained in this document are subject to change without notice. For further information or to obtain additional copies contact:

Bob Rose	HLO2-1/J12	225-6136	SEAMOS::ROSE
Ned Forrester	PKO3-1/K90	223-7015	REGINA::FORRESTER

CONTENTS

1 INTRODUCTION

This document describes the hardware of the Dragon Video System that will be used in many future terminals and display systems. These video terminals may support virtual terminals for the control of several different processes by one user, making them DEC's next generation replacement for the VT100 and VT200 series. New display features may include larger screen sizes, variable pitch fonts, rectangular screen regions, and bitmap graphics.

In this specification, the Dragon system is described in its most general form without regard to which features or screen formats will actually be used in any products. Some features may be reserved for future extensions and not be implemented in the initial product offering.

This specification is the complete description of the Dragon system. The second pass Adder chip (DC323B) and Viper chip (DC322B) have some new features, not in the first pass chips. The specification describes only features of the second pass chips; first pass bugs and differences are listed in the appendix. No product design should use the first pass chips.

Under no circumstances should this document be construed as a commitment by Terminals Engineering to produce any terminal related to the enclosed specification. This specification should only be used for training personnel and designing Dragon based products. Due to the confidential and variable nature of the enclosed information, please do not copy this document or allow uncontrolled circulation. Extra copies may be obtained from the authors.

1.1 Definitions

Throughout this document, the terms "Z axis", "Z dimension" or "color" refer to variations in intensity, color or blink as provided by multiple bitmap planes and does not refer to the third spatial dimension. "X" and "Y" do refer to the horizontal

and vertical spatial dimensions respectively.

The term "local processor" refers to the processor that controls the Dragon system. This processor may have non-video tasks such as providing the communications interface of the product to the outside world, or even running the user's application.

"Dragon" or "Dragon system" refers to the entire video system (with or without a local processor). "Chips" or "Dragon chips" refers only to the Adder and Viper chips. "Hardware" refers to all of the Dragon system EXCEPT the local processor and its firmware.

A "viewport" is the place on the screen to which an image is mapped from a "window" in the users data. The term "region", refers to the implementation of viewports in the hardware.

Differences between this specification and revision 3.4 are highlighted by change bars in the left margin.

A "<" symbol at the start of a paragraph indicates detailed information that readers may wish to skip during their first reading.

1.2 Related Documents

This specification provides general coverage of the hardware and software concepts of the Dragon system. The following documents provide additional detail of interest to product implementers:

1. Dragon Video System Hardware Summary Specification - A 17 page summary of this specification.

2. Dragon Firmware Programmers Guide - A tutorial in Dragon programming. Seriously out of date, but still potentially useful.

3. DC323 Adder Chip Specification - Functional and parametric specification of the Adder chip.

4. Functional Description of the DC322 - Functional specification of the Viper chip.

5. DC322B Video Processor (Viper) Chip Specification - DC and AC parametric specifications of the Viper chip.

The following documents provide implementation examples:

1. DRAGONLIB.C - A library of drawing routines written in the C language. Along with associated calling programs, the library provides a set of examples for using most Dragon functions.

2. Pegasus Viper and Adder Board - These schematics show an implementation of a four plane Dragon system that is programmable to run any memory bus width mode and any resolution mode.

3. Pegasus etch board schematics - Simplified half page Dragon system.

4. QDSS Base Subsystem and Memory Upgrade Modules - A full page, 8 plane system with capability to DMA Dragon commands from a Qbus.

2 SYSTEM OVERVIEW

The Dragon Video System displays on a raster scanned cathode ray tube driven by a bitmap to facilitate flexible formatting of text and to provide graphics capability in all video terminals. Appropriate modularity allows a range of display and memory sizes to be produced, including full page or traditional partial page displays and with one or more planes of display memory. Monochrome and color displays are available. Display regions are implemented to facilitate the use of different portions of the screen by different processes. The system allows high speed text (or graphics) update of the bitmap by the local processor to support editing.

2.1 Principal Features

1. A bitmap display provides many well known advantages over a cell display like the VT100. Examples include: variable character size or positioning and inherent graphics capability.

2. The full page version of the display provides about 850 thousand pixels refreshed on the screen at 60 Hertz, avoiding the flicker and smear of an interlaced display. (Faster systems may also be possible.)

3. The display format is programmable to allow different height, width and refresh rate combinations.

4. 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 though scrolling is in progress.

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

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

7. Multiple bitmap planes (up to 24) support applications requiring gray scale, color or control planes. All planes can be manipulated simultaneously.

8. Any of the planes can be subdivided into two half resolution or four quarter resolution planes. This allows Z depth (color) to be traded for X resolution.

9. Z axis addressing allows an alternate form of memory access that transfers the bits from each plane (or sub-plane) 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.

10. The basic bit manipulation hardware provides a fast character writing rate of 20 thousand characters/second, if fully supported by the local processor (about 200 msec to fill an average page).

11. Text attributes may be extended beyond the VT100 with additions like: arbitrary character size, arbitrary angle, italics, sub/superscript and variable pitch fonts.

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

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

14. Another mode of rasterop 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/sec.

15. The use of 64Kx1 or 64Kx4 RAMs and custom VLSI control circuits will lower the cost of the system to achieve a good cost/performance ratio.

2.2 Hardware

A brief overview of the hardware will aid understanding of the functions presented in sections 3 and 4. Basic timing information is also provided to enable calculation of performance. Additional hardware is introduced in section 3 as it is required. Additional information about the Adder and Viper chips and other Dragon hardware components is contained in section 5; the complete internal specification of these chips is covered by separate documents, see section 1.2.

Figure 1 shows the Dragon system hardware. The design uses two, custom, VLSI chips to reduce the cost and size, and increase the speed of the system. The system comprises six logical components: the local processor (optionally, with a DMA

controller), the Adder (address processor) chip (3 micron, ZMOS), the bitmap memory planes, the Viper (video processor) chips (4 micron, HMOS), the color map and shift register circuits, and the high speed timing logic. The Adder chip (DC323, formerly upper chip) is responsible for functions that are common to all planes, such as: local processor interaction, all rasterop computations, bitmap address generation, clipping, screen refresh, scroll control and monitor sync generation. The Viper chip (DC322, formerly lower chip) provides the data path and control elements that are 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 Viper chip operations during rasterops.

< It is possible to synchronize the screen displays of multiple Dragon systems that are running from the same clock generator; no buses may be shared by separate Dragon systems except the local processor bus. Generally, up to 24 Viper chips may be used with each Adder chip; however, if the loading limitations for the I/D bus specified in section 5.4 are not exceeded, up to 32 Viper chips may be used.

2.2.1 Buses

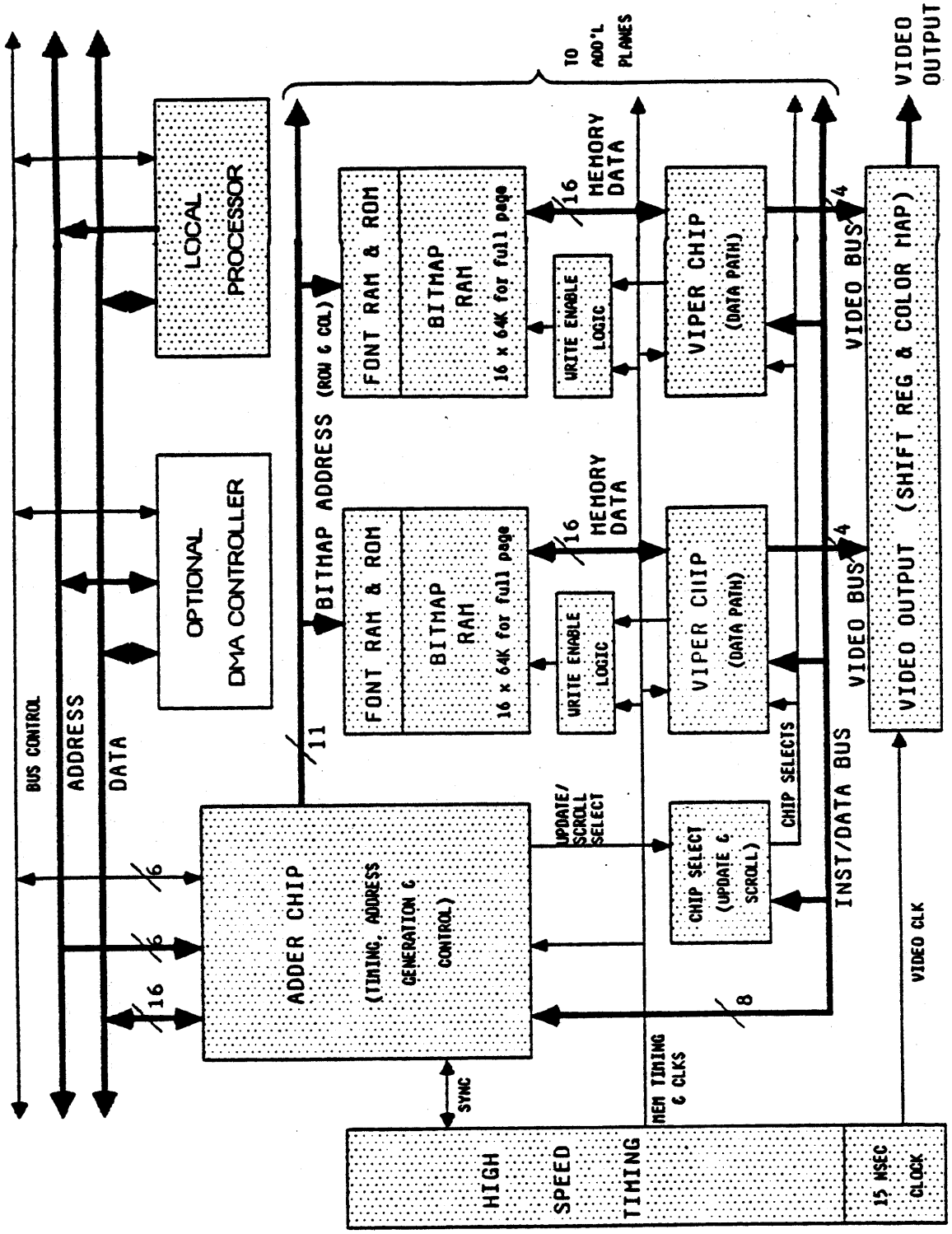
The Adder chip communicates on three buses:

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

   a. the six address lines are provided for direct access by the local processor through its normal memory or I/O addressing modes;

   b. an address counter in the Adder chip allows a DMA controller to pass data and control words to sequential Adder chip registers via a single bus address.

2. The bitmap address bus uses 11 bits each of multiplexed row and column address to the bitmap RAM. This allows an area of up to 8K by 8K pixels to be addressed for update when a 16 bit memory data bus is used (limitations on the displayable portion of this are outlined in sections 2.2.2, 2.2.3, 3.1, 4.1.1.5 and 5.6). All addresses for screen refresh, scroll write back, and bitmap read and update are provided on this bus.



DRAGON VIDEO SYSTEM BLOCK DIAGRAM

Figure 1

3. Finally, the instruction/data (I/D) bus controls the Viper chips and provides for all data flow to and between Viper chips and to/from the local processor (via the Adder chip). Each cycle of the 8 bit I/D bus has four states that provide a 16 bit instruction from the Adder chip to the Viper chips and 16 bits of data from one I/D bus device to the others. These cycles are used to configure the Viper chip registers, to regulate data transfers during rasterops, and to load any external device on the I/D bus, such as a color map. A chip select mechanism, external to the Viper chips (and possibly controlled by the I/D bus) is provided to allow one or more Viper chips to be involved in a transfer.

The Viper chips use three buses:

1. The I/D bus described above.

2. The memory data bus is used in the transfer of all data into or out of the memories during screen refresh, scroll write back and for memory update. This bus is either 16, 8 or 4 bits per Viper chip, depending on the screen size required.

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.

Readers primarily interested in Dragon programming rather than hardware can skip to section 3.

2.2.2 Bitmap Memory

The bitmap memory is configurable to three sizes to allow a reduced cost for displays with fewer pixels. This is accomplished by selecting the appropriate bus width mode for the system and then using a minimum of 16, 8, or 4 64Kx1 memories per plane for full page, half page and quarter page displays respectively. Up to four memory chips may be attached to each Viper chip data pin (limited by capacitive loading) to increase the memory of a full page plane from 1 Mbit to 4 Mbit (for off screen memory) and still operate at the full specified speed; more memories can be added if the speed is appropriately derated.

The following table lists some data about the three modes. Instantaneous pixel time is the fastest that the memory and Viper chip can operate; average pixel time allows 25% of time for the monitor to be in horizontal or vertical retrace. Approximate

number of displayable pixels is given for two popular frame rates; these numbers could vary by +/- 10% depending on the exact numbers of pixels per scan and scans per frame.

Page Name	Bus Width Mode	Number of 64K RAMs	Num. of Pix. in Memory	Inst. Pixel Time	Ave. Pixel Time	Num. of Pixels @ 60 Hz	Num. of Pixels @ 80 Hz
Full	16	16	1024 K	15 ns	20 ns	840 K	620 K
Half	8	8	512 K	30 ns	40 ns	420 K	310 K
Quar.	4	4	256 K	60 ns	80 ns	210 K	150 K

NOTE

< The Dragon system is designed to use 64Kx1 RAMs that have half the normal page mode access time; this speed is provided by fast static column address circuitry on the RAM. Other approaches to obtain the required speed from 64K density parts might be: 8Kx8 RAMs, 16Kx4 RAMs, and 64Kx1 RAMs with other forms of fast page mode output. Some of these options are discussed in section 5.6.3.

2.2.3 Timing

All timing specified in this document is referred to a standard full page pixel period of 15 nsec. This time represents approximately the fastest system speed allowed by the design (the actual specification is 14.25 nsec). Slower speeds are allowed within the limitations of dynamic storage in the Adder and Viper chips and the memories; it will be possible to operate the hardware at half of this speed (assuming a screen format that provides for bitmap memory refresh).

NOTE

< It is likely that some of the process distribution of the chips will be able to operate at greater than specified speed, but this will have to be determined when the parts are characterized. If such chips can be selected, it will be possible to build some systems that have the 60 Hz screen sizes listed above but that operate at 80 Hz, or larger screen formats at 60 Hz; of course, it will be necessary to have memories, monitors and other system components that also operate at the increased speed.

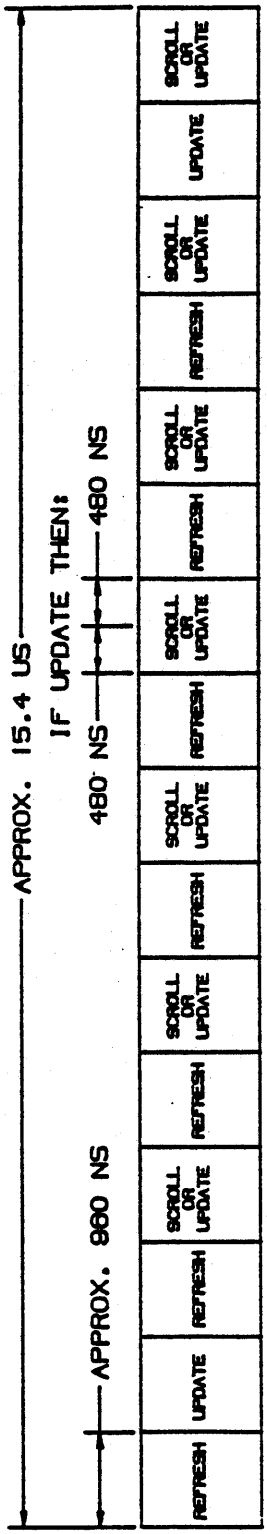
All bus timing, with the exception of the video output bus, is independent of the bus width mode (screen size) and is based on multiples of a 30 nsec clock. The video bus operates at 60, 120 or 240 nsec in a 16, 8 or 4 bit system, respectively; the video bus is always 4 bits wide.

The local processor bus on the Adder chip is asynchronous to the rest of the hardware because its timing is controlled by the local processor (or DMA controller, if present). Internal propagation delays in the Adder chip may require wait states to be added to Adder chip access cycles when used with some processors.

There are two kinds of bitmap memory cycles. A major cycle (960 nsec nominal) is used to read or write 8 words (128 bits in a 16 bit system) to or from the Viper chip for screen refresh or scrolling. Any unused major cycle is subdivided into two minor cycles (also called update cycles, 480 nsec 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 roughly (dependent on screen size parameters) every 0.5 msec (except during vertical retrace, which is less than 1.5 msec) by the screen refresh reads.

< Figure 2 shows an example of a scan which has been divided into 16 major cycles, 7 of which are used for screen refresh, 7 may be used for scroll writeback if any of their words are contained in a scrolling region, and the remaining 2, 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 Viper chips operates concurrently to transfer 8 words (128, 64 or 32 bits) during a major cycle, and to read and return a modified word during a minor cycle.

< Each scan can be programmed to last an integer number of major cycles ($N \geq 2R+2$, where N is the number of major cycles/scan and R is the number of major cycles used to read the data that is refreshed on the screen). Enough screen refresh memory reads must be executed on each scan to display 512 pixels horizontally, regardless of the bus width, to guarantee dynamic memory refresh (not all of these pixels need to be displayed, but they must be read). No more than 2048 pixels may be contained in one scan (including horizontal retrace time); no more than 1024 pixels can be displayed unless extra memory is added to each plane beyond the minimum requirement. The positioning and width of horizontal blank and sync are programmable with respect to memory cycles. Any number of scans per frame may be programmed, up to 2048 (including vertical retrace time); the number of displayable scans is limited by the amount of memory used (for a minimum full page, 992 scans could be displayed; this is the memory height of 1024, less 32 for down scrolling; see section 3.3.1). The position and width of vertical blank and sync are



MEMORY CYCLES DURING ONE SCAN

Figure 2

programmable.

The I/D bus operates continually at the minor cycle rate (480 nsec nominal), whether major or minor cycles are in progress and in sync with the memory cycles, during which it transfers two instruction bytes and two data bytes.

The computation cycle of the Adder chip is two major cycles (1920 nsec, nominal). In 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 (see additional conditions in section 3.5.1.3). 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, eg. 3 sources and 3 destinations, 6 destinations, etc.). This allows optimum use of update cycles. No source and one 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 in addition to the rasterop execution time, although 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.

All performance data in the remainder of this document will be for a full page (16 bit) system only. The performance of the smaller systems can be calculated from the above data.

3 OPERATIONAL DESCRIPTION

This section describes the overall capabilities, functions and performance of the Dragon video hardware in general terms; most of the description covers the Dragon chips with reference to support expected from the local processor. Section 4 will present the interface and registers of the Adder and Viper chips in detail.

3.1 Memory Organization

The design of the Adder 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 to the local processor, and the Adder chip does not perform rasterop manipulations in the local processor's memory; but, requests may be made for data exchange between processor memory and the bitmap.

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] to (but not including) the address [X limit, Y limit] (these limits are programmable in the Adder chip; this includes all memory that is read (but not necessarily displayed) by the screen refresh process. 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 off screen memory is all the rest of the memory; it is "L" shaped in the general case. (See the description of the X limit and Y limit registers in section 4.1.1.5 for more information.)

The dimensions of the bitmap memory are implementation dependent. Generally, the height and width are powers of two; for minimum 16, 8, and 4 bit bus widths the height is 1024, 512 and 256 respectively, while the width is 1024 in each case. When additional memory is used, rectangles of these sizes can be added in either the X or Y direction out to a limit of 8Kx8K pixels. It is possible to use hardware external to the Adder chip to alter the power of two restriction.

3.2 Dragon System Control

The hardware is controlled by the local processor through the loading of command and data registers in the Adder chip. Registers control the configuration of the Adder and Viper chips for: screen format, video sync, scrolling and rasterops.

The processor can load any of the writable registers directly. Several status bits can be read by the processor to know if the Adder chip is ready to accept (or provide) data; these include: rasterop initialization complete, rasterop computation complete, address buffer empty, pause complete (programmable frame sync), scroll service required and vertical blank (fixed frame syncs), ID bus data ready, and various clipping occurrences (the use of each bit is described in section 4.1.1.1). Any of these status bits may be enabled to assert an interrupt request pin on the Adder chip.

< If the local processor is not dedicated to video service or is providing complex video services, it may not wish to wait for the appropriate Adder chip status before proceeding with its next request. To allow decoupling of local processor computations from Adder chip execution, an interface for a DMA controller has been provided. A request pin may be programmed to assert on any of the same conditions that are available as flags or interrupts to the local processor; this request will fetch the next data word from the DMA controller. When loading data and command registers, the DMA controller writes to a special address in the Adder chip that is associated with an internal address counter. If the MSB of the data word is clear, the counter addresses the Adder 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 ID bus data registers contain 16 significant bits of data; to allow arbitrary data to be loaded to these addresses, the MSB does not cause the address counter to be loaded if it is pointing to either of the ID 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 inadvertently left pointing to an ID data register). The register addresses are assigned such that common repetitive functions only need access one group of consecutive registers, in addition to the command register.

3.3 Viewport Support

To assist the implementation of multiple viewports on the screen, the Dragon chips provide clipping to and scrolling or dragging of rectangular regions (the term "region", as used in this document, 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 4 pixels from the left side of the screen. Figure 3 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 so that 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 and will continue to contain

any data placed in it when it was previously contained in a region. The whole update region may be either scrolling or not, but a region should not be updated if only part of it is scrolling (eg. region 11 should not be updated if region•5 is being scrolled).

NOTE

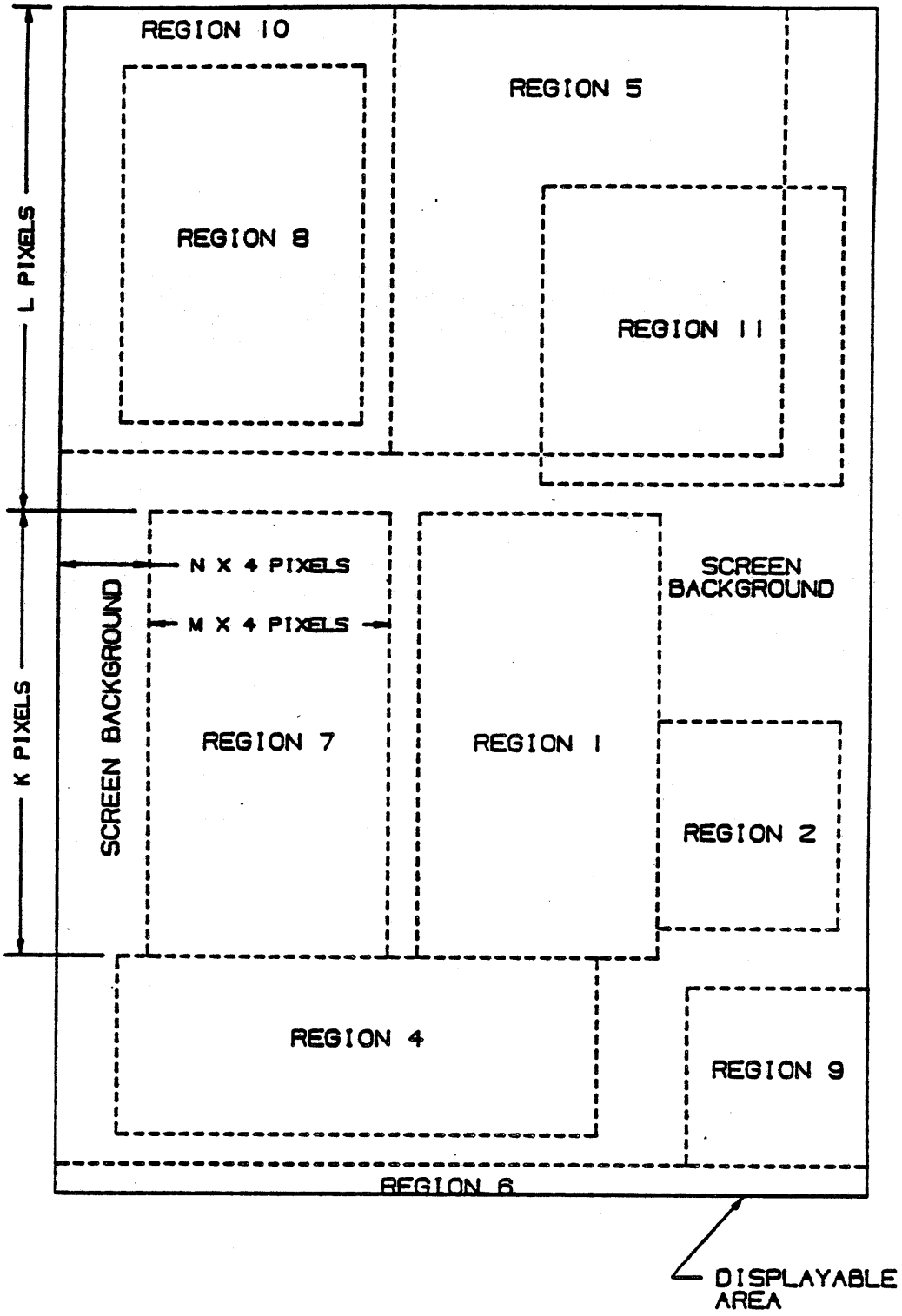
< It is possible to have the left and right edges of a region specified to single pixels, but this requires many chips external to the Adder chip. Also, clipping status would not correspond to these edges; and, it would be impossible to use memories that are organized by four bits (because these will have only one write enable for all bits). For more information, see section 5.7.1.1.

When updating a region, clipping allows the image outside of the region to still be computed but with writing disabled. The Adder chip provides status to the local processor to indicate if any rasterop was completely or partially clipped while writing or whether any rasterop was not completely clipped (to aid a picking algorithm). These clipping status bits are accumulated as rasterops progress, and may be read or cleared at any time, but the results are only predictable if they are read or cleared between rasterops.

To the greatest extent possible, the regions are allowed to be independent. In some ways, this is not 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). Unless large amounts of color map and a controller are provided, all regions must share one color map and agree on the resolution mode settings. Unless large amounts of extra bitmap are provided, regions probably share the off screen areas used for symbol storage. Obviously, overlapping regions cannot be independent of each other.

3.3.1 Scrolling

Scrolling movement can be up, down, left or right (not diagonally) at a rate from 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, about 30-40% of the non-scrolling update time is still available for source/destination operations, depending on screen format (50-80% of non-scrolling time is available for destination only or source only operations because these operations are primarily compute



REGION EXAMPLE

Figure 3

bound when not scrolling).

NOTE

< Actually, the distance that may be moved during an upward scroll is limited only by the number of scan times allowed for vertical retrace, as this time is used to write the fill color back into the vacated memory (if the region extended 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 of 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 any size region, 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 want to write portions of the same word to more than one memory location. Second, down scrolling cannot be accomplished in the obvious way because this would require writing data into memory locations not yet read by the refresh/scrolling process, thereby destroying screen data.

< Instead, down scrolling moves all of the data outside the scrolling region up, and then offsets the memory address at which screen refresh starts reading the bitmap, thereby creating the illusion that the scroll region moved down. (Proper synchronism 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 down scrolled in one frame time.

2. The consumption of writing time is as if the whole screen were up scrolling. 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 sideways motion of a region.

4. Management of the Y Offset adds extra work for the local processor.

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

To prevent interference between the loading of Viper chip registers for scrolling and for update, a separate ID command and data path (including chip select control for the Viper chips) is provided for scrolling; 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. The use and control of the index registers is somewhat more complicated and is described in section 3.3.4.1.

NOTE

< To conserve die area in the Adder chip, scroll registers are only buffered 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, a smooth effect cannot be achieved by this technique, anyway, so it should not matter if the abruptness of jump scrolling of the two regions is increased. If the local processor can service an interrupt request to load four registers in 200-500 usec (depending on screen format), then it will be possible to scroll different regions in successive frames without the extra buffering of the index registers. (Normally, the local processor may take an entire frame time to load any buffered register.)

3.3.2 Dragging

The scroll function can be used to smoothly drag a region 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 only be defined on four pixel increments in the horizontal direction, horizontal dragging must stop on four 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 following 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 like the scroll boundaries.

3.3.3 Clearing A Region

The Adder 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 will be 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 and without waiting for the next frame time, use of the erase mechanism will provide a very clean appearance because it is synchronized to occur between two display frames.

3.3.4 Drawing In The Scrolling Region

The blank space that is created by scrolling needs to be filled with new data. It is desirable to draw while scrolling continues so that the smooth effect of the scroll is not disturbed. The indexing mechanism of the Adder 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 in 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 that the clipping region cannot be used as a "drawing function" to control the shape or extent of objects being drawn into a moving region, 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, (to fill the blank areas created by scrolling), the pause register and the interrupt or request enable bits can be set to interrupt the local processor or request further data from a DMA controller when a specific point on the screen has been passed by screen refresh. Normally, the pause would be set, by the scroll service routine to allow the update process to continue after the top or bottom of the scroll region has been displayed; this allows a full frame time to fill the top or bottom edge of a region that is up or down scrolling, before the blank space would become visible.

3.3.4.1 Indexing

To allow updating from the local processor or DMA controller into a region that is scrolling (or has scrolled) without requiring the local processor to change the coordinate values in its display list, there are index registers in the Adder chip that 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 will become the new values at the start of the next frame. Between frames, the Adder 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); and 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 local processor thus, in general, if the update process is driven by a DMA controller, the update process may not change the update region to one that is scrolling because a frame may end between the time that the local processor places index values in the data list for the DMA controller and the time that these values are loaded into the Adder chip. Thus it is best not to send region changes through a

DMA controller.

< When the local processor changes the update region, the following interlock with the scroll process is recommended:

1. The local processor 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 what 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 Adder chip registers. The index registers must be loaded in the order: pending, new and then old. If the instruction(s) that is used to move each index value to the Adder chip is 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 will be awakened by the scroll service interrupt from the Adder 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 will be available to the update process.

6. The scroll process should load all six index registers in the Adder 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 Adder 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.

3.4 Multiplane Support

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

The number of planes is limited, by the fanout of the I/D and address buses, to 24 Viper chips (buffering of the address bus is required; the 400 pf maximum load on the I/D bus may permit 32 Vipers if a close layout is maintained). Z axis operations are limited to addressing 4, 16 bit words of Z data which allows a maximum of 64 planes or subplanes; this corresponds to half the maximum number of subplanes that would otherwise be possible from 32 Viper chips.

3.4.1 Resolution Mode

The resolution of any or all planes may be set to 1, 2 or 4 bits/pixel. In one bit mode, each of the four bits appearing on the video bus of a Viper chip defines two possible states for each of four pixels on the screen; in two bit mode, each of the two pairs of bits defines four possible states for each of two pairs of horizontally adjacent pixels on the screen; and, in four bit mode, each output to the video bus defines 16 possible states for four horizontally adjacent pixels. One of the two or four divisions of a low resolution plane is referred to as a subplane.

The actual interpretation of the video bus data as multiple states for adjacent pixels must be provided in the color map; but, the Viper chip provides modifications to the use of data and masks, the interpretation of barrel shift values, and the effects of Z axis addressing that aid the manipulation of low resolution planes (see section 4.2.1 for an complete description of the effects of resolution mode). Resolution mode in the Viper chips affects only the writing of new data so that different regions can have different resolution settings; however, additional external hardware would be required to actually provide different color map interpretations to different regions.

< Resolution mode has the limited application of allowing a tradeoff between Z depth and X resolution in one or more planes; it is not intended to provide more general purpose planes than there are Viper chips. Note that subplanes are NOT independent of each other as are planes implemented with separate Viper chips. Some of the restrictions on subplanes are:

1. Subplanes share update settings for the logic function and control store ram in the viper chip.

2. Complement mode writing will not work in general because each of the bits in a low resolution pixel will be addressed separately, but both bits will be written during each access. Some but not all of the problems with complement writing can be avoided by programming a mask register in the Viper chip to permit writing to only one bit within a low resolution pixel.

3. Writing speed per low resolution pixel will be proportionately reduced because The Adder chip still addresses the memory as if it were full resolution, accessing each bit within a low resolution pixel. Of course, there are fewer low resolution pixels on the screen, so the time to draw a given fraction of the screen is the same as for full resolution.

4. Fonts and patterns loaded off screen may have to be scaled up horizontally, depending on the intended use.

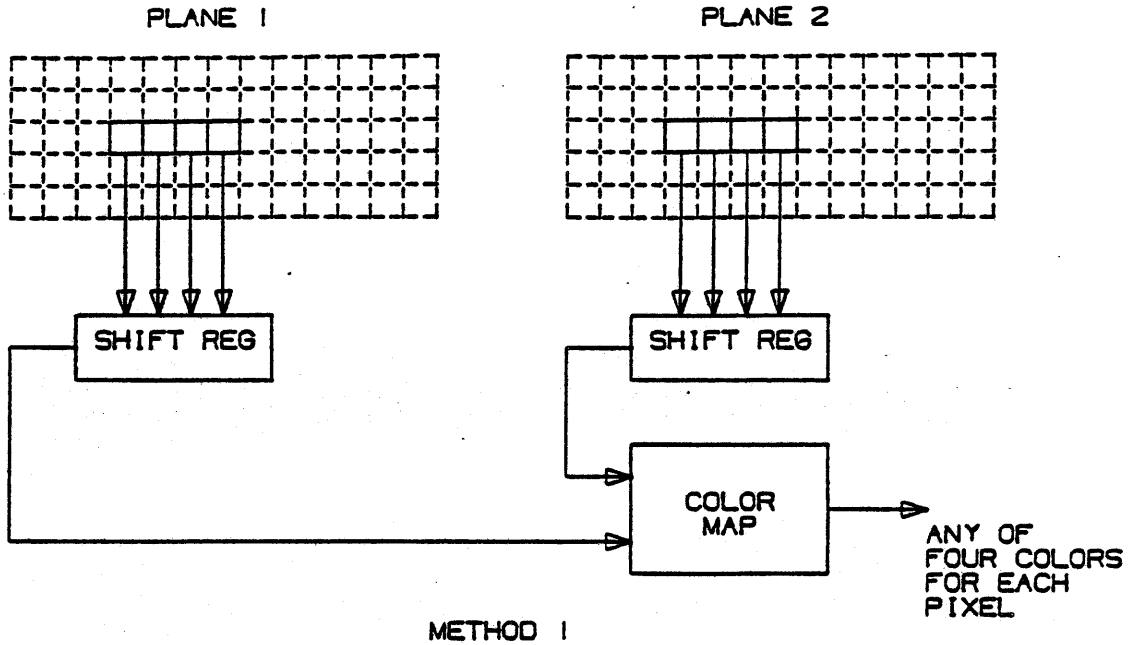
5. Subplanes cannot be scrolled independently. When horizontally scrolling a region that contains subplanes, the region should only be moved by multiples of the lowest resolution subplane to preserve the phase of low resolution data. The maximum horizontal scroll speed (pixels/frame time) would be proportionately reduced in low resolution planes because the pixels are larger.

6. When subplanes are enabled in a Viper chip, the low resolution pixels are of longer duration; enabling subplanes does not increase the data rate through a Viper chip.

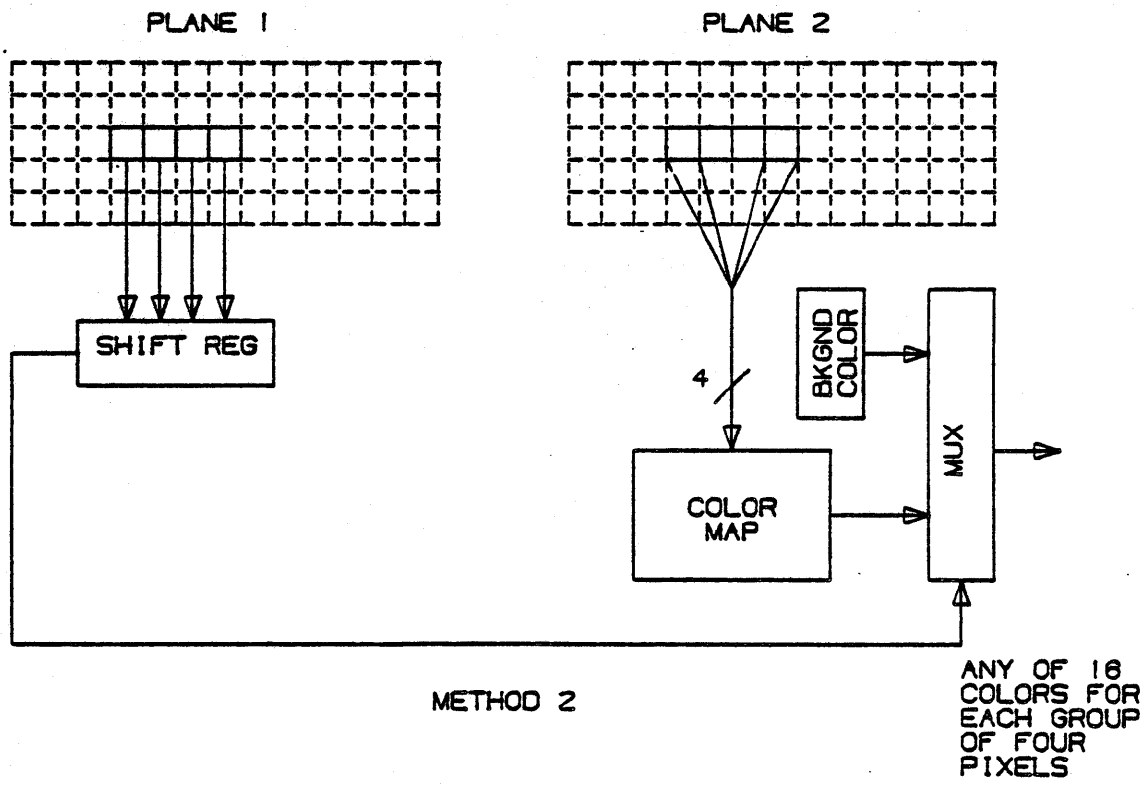
< Figure 4 shows two examples of the use of resolution mode. In the upper example, two planes are used, both in 1 bit mode, whose video output buses are serialized by shift registers to form a two bit code that accesses one of four values from a color map to determine the color of each pixel. In the lower example, one plane is used in 4 bit mode to choose one of sixteen foreground colors for four adjacent pixels while the 1 bit mode plane makes an independent choice of this foreground color or a screen-wide background color for each pixel in the group of four.

3.4.2 Z Axis Addressing

Z axis operations allow the exchange of data between the Adder and Viper chips on the I/D bus by using the 16 bits of a data word to transfer one bit to/from each of 16 planes or subplanes. This can be used for both the exchange of data between the local processor memory and the bitmap, and the loading of appropriate Viper chip data registers.



METHOD 1



METHOD 2

ALTERNATE USE OF MEMORY PLANES FOR COLOR

Figure 4

< Each Viper chip can be programmed with a six bit plane address; the lower four bits specify the bit within an I/D bus data word to which a Viper chip will respond during a Z axis operation; and the upper two bits specify the Z block (separate I/D bus transaction) if more than 16 planes or subplanes need to be transferred. Alternatively, chips selects may be used to enable groups of viper chips to form Z blocks.

< A 1 bit resolution plane may have any plane address, and transfers the addressed bit on the I/D bus; a 2 bit resolution plane must have an even address, and transfers the addressed bit and the next higher bit; and a 4 bit resolution plane must have an address that starts on a nibble boundary, and transfers the whole nibble whose lowest bit corresponds to the plane address. No two planes may have the same address; and no other plane may have a plane address equal to the addresses of any of the two or four bits accessed by a low resolution plane (this must be ensured by the local processor).

During local processor/bitmap data exchanges, data in a normal rectangle from the bitmap is transferred to/from a DMA controller or the local processor. If Z mode transfers are requested, each word transferred will be the color of one pixel. For more information see sections 3.5.3 and 4.1.2.3.2.

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 Viper chips. The source register is one input to the logic units in the Viper chip, the foreground and background registers are selected on a bit-for-bit basis by the output of the logic unit and loading them with a Z axis command can program the Viper chips for the colors with which subsequent elements will be drawn. The fill registers in the Viper chips define the color that will be written into the new areas of scrolled regions.

3.5 Basic Address Calculation And Data Path Hardware

The Adder and Viper chips contain bitmap manipulation hardware to execute the commands provided by the local processor. 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 Adder chip handles all word and bit addressing, clipping, and control. The Viper chip provides bit alignment, data exchange and logical combinations. There are three rasterop modes: normal, linear pattern, and fill. This section will describe the normal mode; the linear pattern and fill modes will be described in section 3.7.

The contents of the Adder and Viper 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 content of data registers may be changed by rasterops.

3.5.1 Adder 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 being combined with the destination. No curve algorithms are implemented because all of 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 Adder chip.

The destination is a parallelogram and may be a 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 non-square pixels. As discussed below, rotation and scaling may not be suitable during complement mode drawing operations.

The sequence of source/destination operations is controlled by the Adder chip command register; three bits select one of the eight possible combinations of sources and a destination (sources: none, first, second and both, destination: on, off; not all of these combinations are useful); 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 Viper 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 Adder chip use 14 bit, two's complement numbers for each of X and Y. Beyond the 14 bit limit, calculations will 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) because these values are multiplied by two in some internal calculations; however they are still specified as 14 bit, two's complement numbers. 13 bits of the X address (including specification of the pixel within a word) and 13 bits of the Y address are available at the output of the chip (in 16 bit bus width mode). The large address output of the Adder chip is provided to allow bitmap writing for a laser printer, which will require about four times the resolution in both X and Y as will a video display.

< A six 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 8 operations can be performed during each scan of scrolling if the screen format allows); and, if source/destination operations are required, then three (or two for double source) operations can be stored.

In the following discussion of Adder chip computation functions, it may be helpful to refer to the Adder chip computation address path diagram, shown in Figure 5.

3.5.1.1 Destination - Rotation

The destination is a parallelogram defined by two vectors and an origin (see Figure 6). 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; and then, using the next pixel on the path of the other vector (the slow vector) as a new origin for another fast vector with its same DX and DY; fast vectors are scanned until all of 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 non-square pixels.

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 component 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 because then 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 because this would restore a pixel to

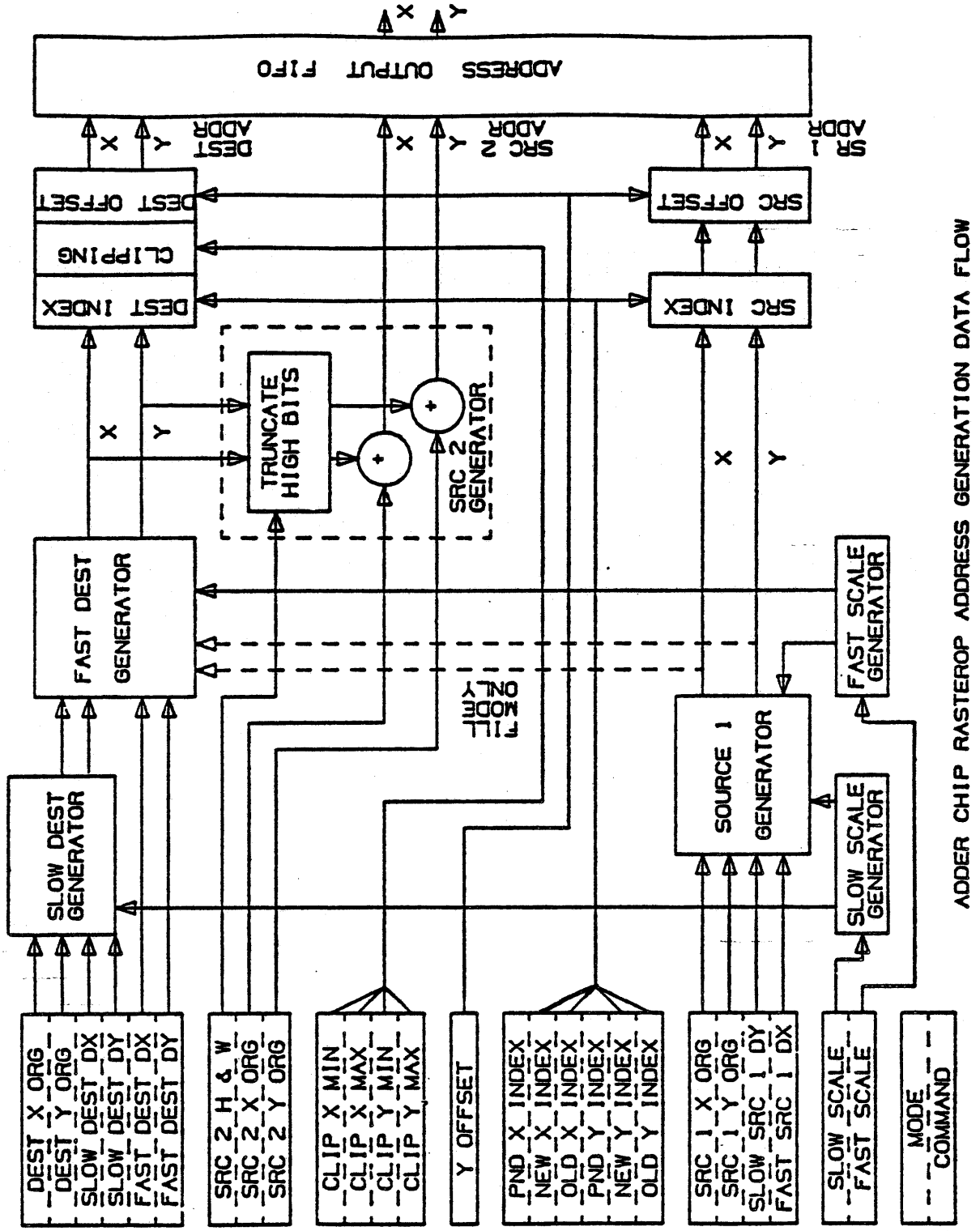
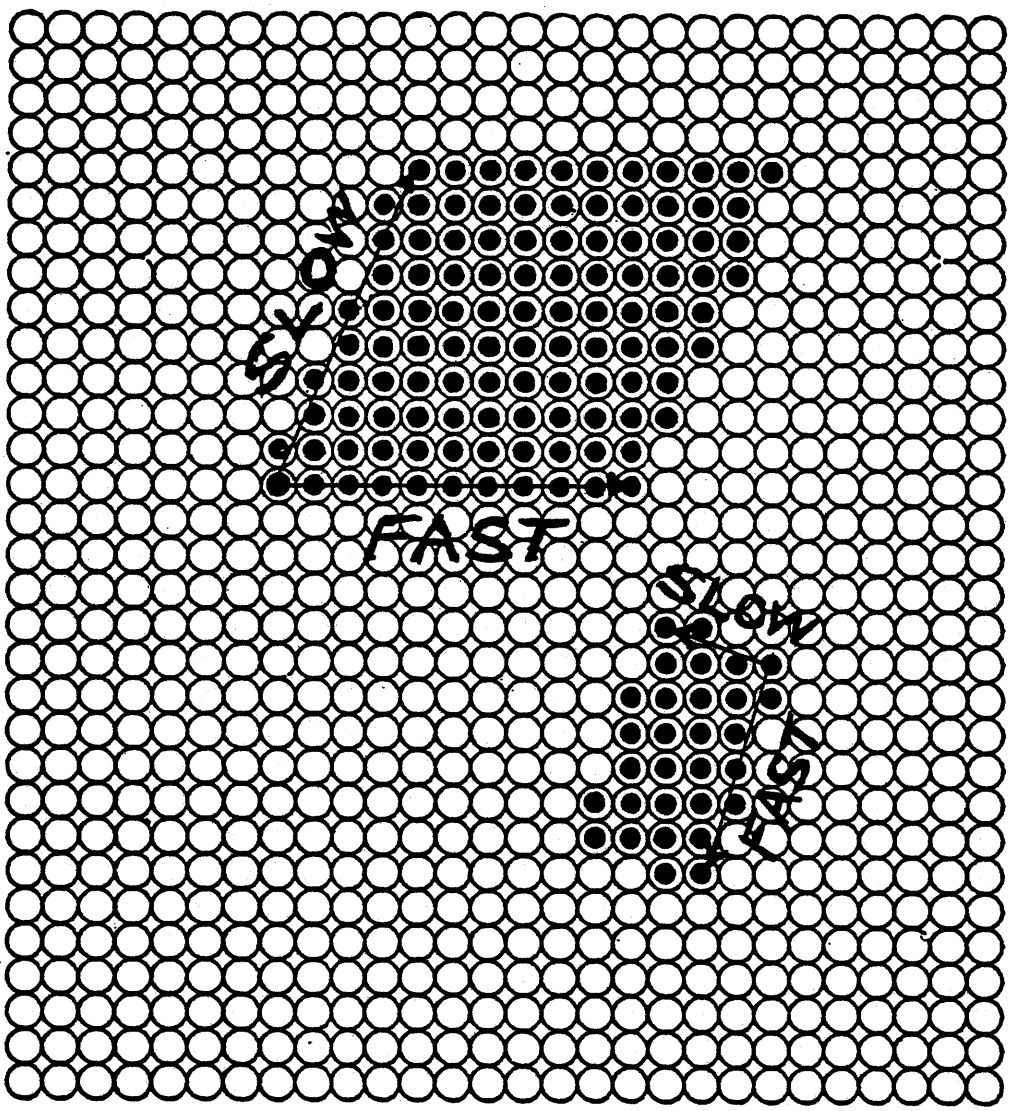

ADDER CHIP RASTEROP ADDRESS GENERATION DATA FLOW
ADDRESS OUTPUT FIFO
X
Y
DEST ADDR
SRC 2 ADDR
SR 1 ADDR
DEST OFFSET
CLIPPING
DEST INDEX
SRC OFFSET
SRC INDEX
TRUNCATE HIGH BITS
SRC 2 GENERATOR
FAST DEST GENERATOR
FILL MODE ONLY
FAST SCALE GENERATOR
SOURCE 1 GENERATOR
SLOW DEST GENERATOR
SLOW SCALE GENERATOR
DEST_X_ORG
DEST_Y_ORG
SLOW_DEST_DX
SLOW_DEST_DY
FAST_DEST_DX
FAST_DEST_DY
SRC_2_H_&_W
SRC_2_X_ORG
SRC_2_Y_ORG
CLIP_X_MIN
CLIP_X_MAX
CLIP_Y_MIN
CLIP_Y_MAX
Y_OFFSET
PND_X_INDEX
NEW_X_INDEX
OLD_X_INDEX
PND_Y_INDEX
NEW_Y_INDEX
OLD_Y_INDEX
SRC_1_X_ORG
SRC_1_Y_ORG
SLOW_SRC_1_DY
FAST_SRC_1_DX
SLOW_SCALE
FAST_SCALE
MODE
COMMAND

Figure 5

its original state during complement writing. Because the destination 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.



Example Destination Rasters

Figure 6

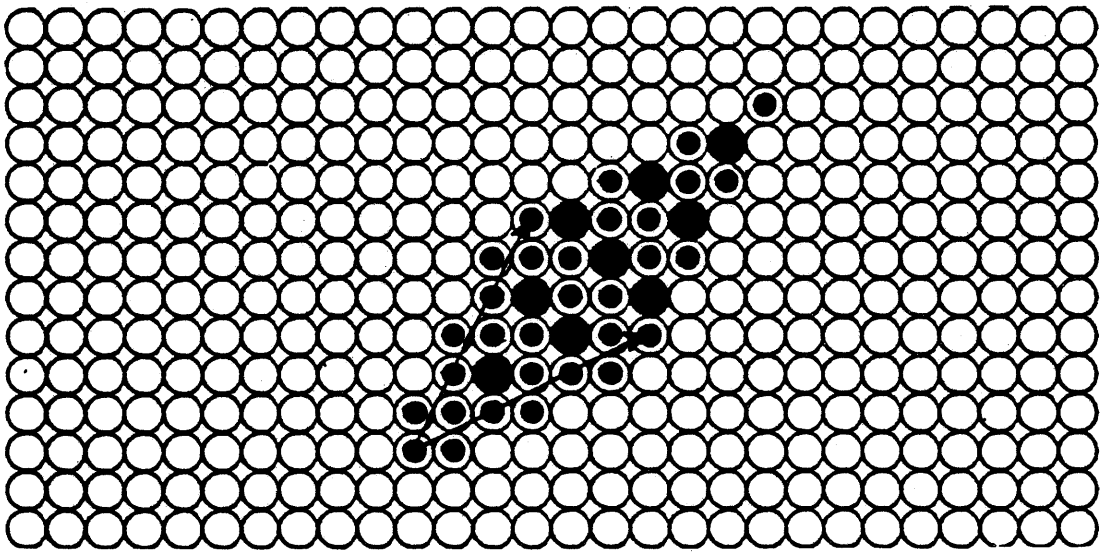
3.5.1.1.1 Holes And Duplications

< 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 some vector combinations will cause both effects or neither effect within the parallelogram. The following combinations of fast and slow vectors will create duplications (Figure 7 shows an example that creates duplication):

1. If both vectors point in one of the eight cardinal directions (0, 45, 90, etc.), then 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, then duplications will only occur if the other vector is in one of the two adjacent octants or one of the two octants adjacent to the opposite cardinal direction.

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



Example of Fast and Slow Vectors that Duplicate Pixels

Figure 7

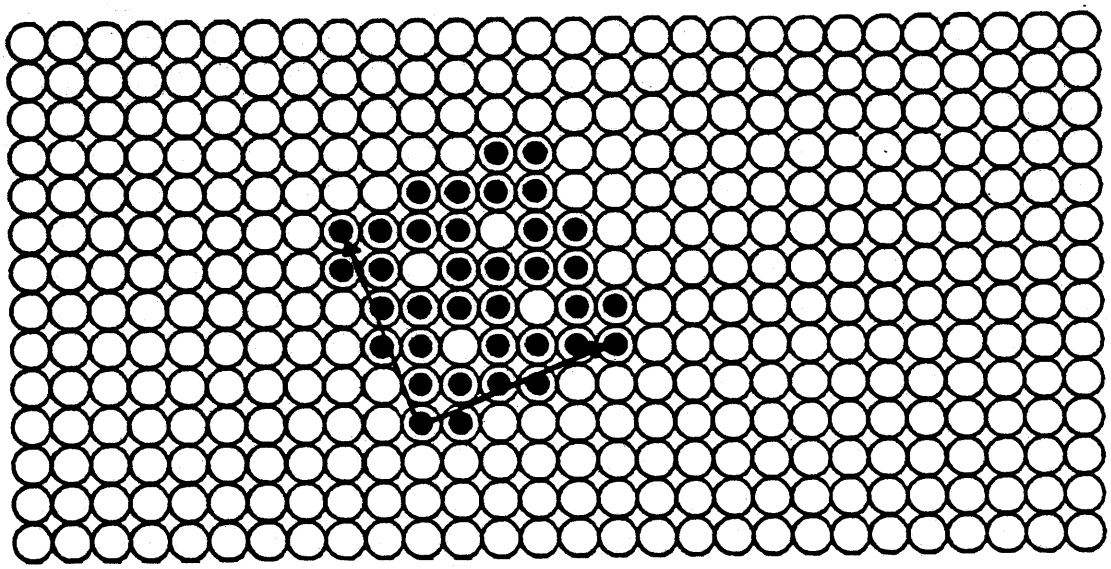
Note that no RECTANGULAR area of any orientation (on a screen with square pixels) will cause duplications. Duplicated pixels will cause problems for complement operation, but one would expect some overlapping of data when an image is compressed by any of the above conditions or by scaling down an image. These complement problems can be alleviated by doing any transformations in off screen memory without complement mode, and then copying the result to the visible screen without a transformation; two sets of control registers are provided in the Viper chip to support repeated two-step operations.

< The following combinations of fast and slow vectors will create holes (Figure 8 shows an example that creates holes):

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

2. If neither of the vectors is parallel to the X or Y axes, then holes will only be generated 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.



Example of Fast and Slow Vectors that Leave Holes

Figure 8

3.5.1.1.2 Bresenham Error Computation

< Bresenham's Algorithm is implemented in the Adder chip by unconditionally incrementing the position register for the major axis, but only incrementing the position 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, and 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 content of the Error 1 register is added to the destination slow vector error register, and the Error 2 register is added to the destination fast vector error register. In fill mode, the Error 1 register is used 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 half a 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. These error registers are normally set to zero.

3.5.1.2 First Source - Scaling

The first source is an unrotated rectangle defined by a fast vector (a signed width, the fast vector of the source is 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 determined by the length (major axis)

of the fast vector of the destination times 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 two's complement source DX is not significant in normal mode, it is used for linear pattern and fill modes described in section 3.7). Similarly, the magnitude of the source slow vector is determined by the length of the destination slow vector times the slow scale factor; and the sign is determined by the sign of the source DY register.

< A fast or slow scale factor is a number less than one with an indication of up or down scaling, stored in a 14 bit register. The MSB (bit 13) indicates up or down scaling and 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 Adder 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 always incremented, vice versa if down scaling is specified. One pixel from the smaller vector may map to more than one pixel in the larger vector; this results in non-uniform sampling of the smaller vector during the scaling process. The scaling of fast and slow vectors is independent. Down scaling will cause pixels in the destination to be written multiple times; if complement writing is used, two 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 will ensure that the first pixel is not read one time too many. However, the Adder 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 in the scale factor, inaccuracies can exist in large scale factors. This problem becomes significant when the number of significant bits in the scale factor is reduced (by having a large scale factor) 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); and 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 (see section 3.5.2.2). The first source is most commonly used to move objects in the bitmap such as characters, or to move areas of the screen.

3.5.1.3 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 one, and the source and destination fast vectors are in the same direction (same sign) (the last condition is not necessary if the first source is disabled), the hardware will operate on words the size of the bus width to increase the speed of writing; this "fast mode" is invoked automatically under the appropriate conditions.

Fast mode is disabled if any of linear pattern mode, fill in Y mode, or Z mode (used for processor/bitmap transfers) are enabled. These modes are described below.

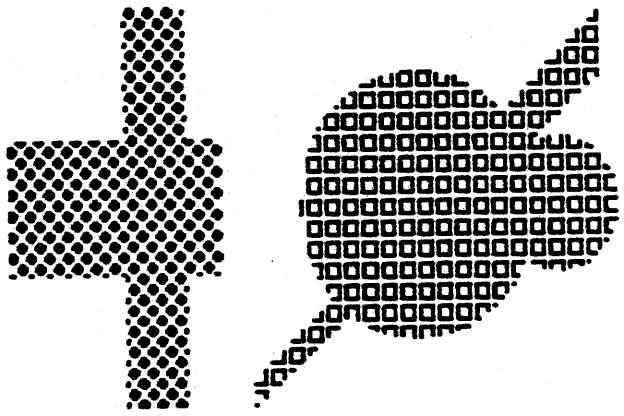
3.5.1.4 Second Source - Tiles

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 just destination). No scaling or rotation is provided between the second source and the destination, although the first source may still have these properties, of course. 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 that are stored in off screen memory (see section 3.1 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 two dimensional rectangular pattern. An important property 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 9, is as if copies of the pattern rectangles were used to tile the whole screen and, when a patterned object is written, these tiles are exposed to view throughout the interior of the object. The Adder chip achieves this effect by restricting the height and width of a pattern rectangle to integer powers of 2 and adding just 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. The effect of the source 2 origin is more like a pointer to the origin of the tile than like an offset, when tiling is enabled. 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, but one can still have 4 and 8 wide tiles in 16 bit mode by repeating the tile elements within the tile cell. Tile origins must be on word boundaries in the memory. The 512 maximum tile size allows up to at least one inch tiles on a laser printer.



Tiled Areas

Figure 9

Like the general second source, no scaling or rotating of tile patterns is provided because the Adder 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 local processor and using linear patterns (see section 3.7.2) to write one scan of the patterned area at a time.

3.5.2 Viper Chip - Data Manipulation

The Viper chip performs the actual manipulation of the memory data accessed by the Adder 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 Viper chips, and modifying the destination data with the logic unit. In the following discussion of Viper chip functions, it may be helpful to refer to the Viper chip rasterop data path diagram, shown in Figure 10.

3.5.2.1 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. In narrower bus width modes, these widths are reduced proportionately. 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 and the other always with the second source. The output of the barrel shifter can be directed to the source or mask registers on the Viper chip and/or to the I/D bus for broadcast to the other Viper chips or the Adder chip. Only one Viper chip can drive the I/D bus during any source operation.

< The Adder 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, if selected for the destination. If a Viper chip is set to two bit or four bit resolution mode, the least significant one or two bits of the barrel shift constant, respectively, are truncated in that Viper chip; this prevents a rasterop from shifting the bits of a low resolution plane by less than a bit pair or nibble.

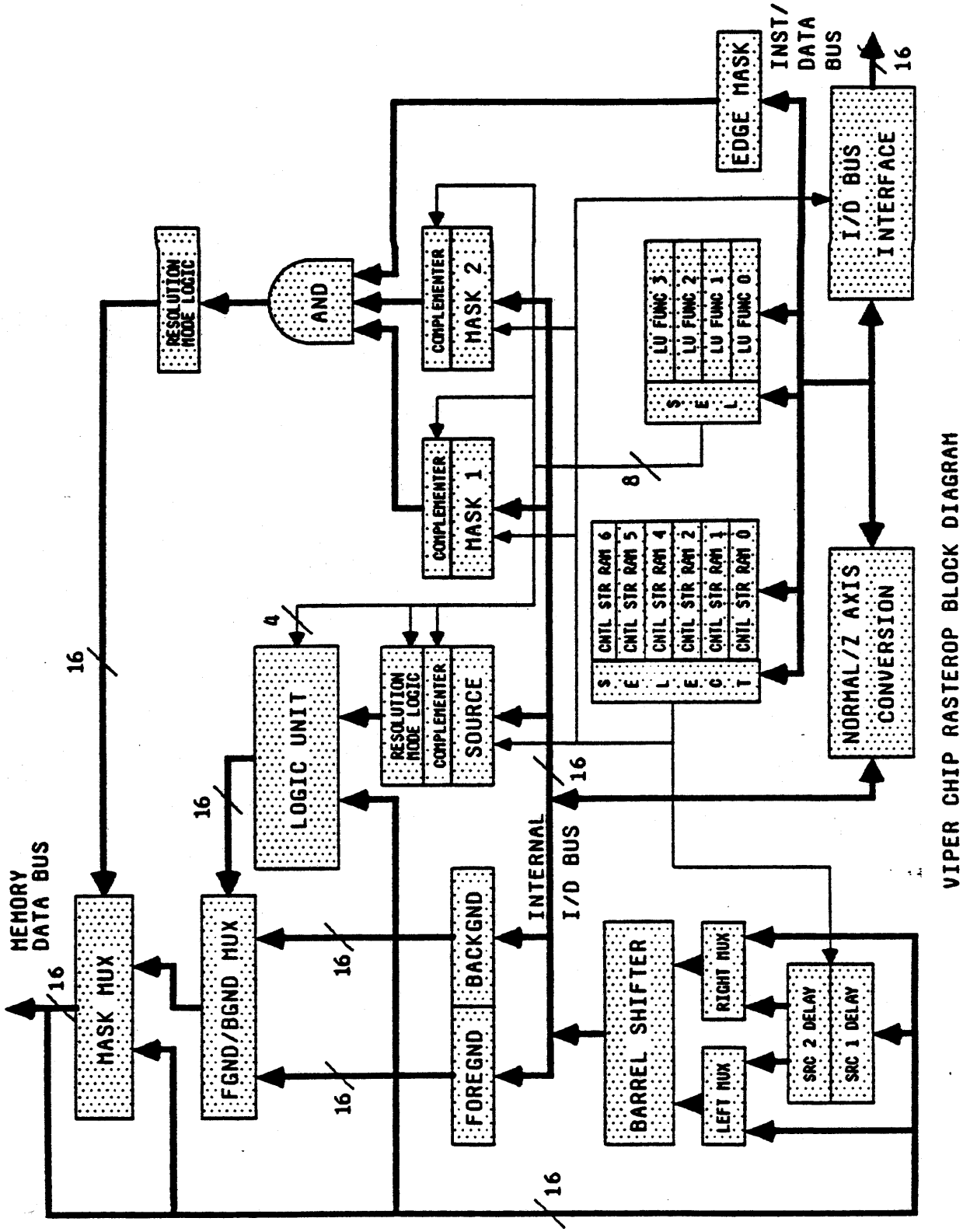

VIPER CHIP RASTEROP BLOCK DIAGRAM
MEMORY DATA BUS
16
MASK MUX
FGND/BGND MUX
16
16
16
LOGIC UNIT
4
16
RESOLUTION MODE LOGIC
COMPLEMENTER
SOURCE
16
INTERNAL I/D BUS
16
BACKGND
FOREGND
16
BARREL SHIFTER
RIGHT MUX
LEFT MUX
SRC 2 DELAY
SRC 1 DELAY
RESOLUTION MODE LOGIC
AND
COMPLEMENTER
MASK 2
COMPLEMENTER
MASK 1
8
SEL
LU FUNC 3
LU FUNC 2
LU FUNC 1
LU FUNC 0
SEL ECT
CNTL STR RAM 6
CNTL STR RAM 5
CNTL STR RAM 4
CNTL STR RAM 2
CNTL STR RAM 1
CNTL STR RAM 0
EDGE MASK
INST/ DATA BUS
16
I/O BUS INTERFACE
NORMAL/Z AXIS CONVERSION

Figure 10

3.5.2.2 Logic Unit, And Source And Mask Registers

During a bitmap memory read-modify-write cycle, the logic unit combines the content of the source register (loaded with a constant or during a preceding source read cycle) with the present 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 local processor using a Z axis load, or 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, it is necessary to spread adjacent pattern bits 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 two or four bits (in two and four bit modes, respectively) so that if any of the bits in the group is a "1" then all two or four bits of the group will be one at the input of the logic unit. In order that either the ones or the zeros in the source can be so spread, there is a programmable complementer between the source register and the resolution mode logic. 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 a similar fashion to the source register. 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. A special mask (called the edge mask) is provided by the Adder chip, 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 of these mask words are ANDed together so that only bits selected by all three masks will be modified; bits not addressed by the Adder chip as being in the range of a rasterop cannot be modified. When no mask is being used, mask 1 should be loaded with all ones and both complementers turned off to prevent masking of destination bits. If only the 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.

< The combined masks are processed by resolution mode logic, similar to that used for the source, which ORs the adjacent mask bits according to the setting of resolution mode. This forces all of the bits of a low resolution pixel to be modified together. The resolution mode logic for the masks CANNOT be disabled.

3.5.2.3 Control Store RAM And Function Registers

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

1. where to send the read data from the barrel shifter (source, masks, I/D bus),
2. where to send incoming I/D data (source, masks), and
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 controls transfers during a source 2 read (and specifies the control of the other delay register for the second source), and the third controls the destination read-modify-write. The two banks allow easy switching between two functions without reprogramming the Viper chips (such as, when a function that must be performed with two rasterop steps is executed repeatedly). The Adder chip selects which bank to use (as specified by the local processor) 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 Viper chips often do different things at the same time; for example: during text operations one Viper chip will be reading the font and transmitting it on the I/D bus while the others will be receiving the font from the I/D bus.

NOTE

< Some of the functions of the control store RAM are vestiges of an earlier design and may no longer be required. It is not clear that there will ever be a time when a delay register is NOT used; the delay register control bits are probably always enabled in the two source CSRs. Furthermore, there is probably no use for a destination CSR to ever contain a value other than zero (NOP).

The other memory 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, and 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 Adder chip during rasterops, as specified by the local processor. These registers permit convenient transitions between tasks requiring different logic selections such as writing normal and reverse text. Four registers allows two orthogonal functions to be programmed.

3.5.3 Processor/Bitmap Transfers

Processor/bitmap transfers (PBTs) are similar to rasterops, except that either the source or destination is the local processor bus. Processor to bitmap transfers are referred to as PTBs; Bitmap to processor transfers are referred to as BTPs; and transfers of either type are referred to as PBTs. All data for a PBT is transferred through the six deep I/D data FIFO. There are two modes for PBTs, X mode and Z mode. The second source is not available; no scaling or rotation are available.

In an X mode PBT, data from one plane is transferred to/from the local processor bus with 16 bits adjacent in X occupying one word (in 16 bit bus width mode) followed by additional X words and then by additional scans. In 8 and 4 bit bus width modes, only the low 8 or low 4 bits, respectively, are significant (there is NO packing performed by the Adder chip to build words from bytes and nibbles).

In a Z mode PBT, all the bits from one pixel location of up to 16 planes (or subplanes) are transferred to/from a local processor bus word, followed by additional pixel data along the fast vector and then by additional scans. If a system has more than 16 planes, additional 16 plane blocks (Z blocks) can be transferred with additional PBT commands.

A PBT transfers a continuous stream of data words to/from the local processor memory, either through the local processor itself or through a DMA controller; 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, in the usual fashion. Depending on the sophistication of the transferring device, these words can be all sequential in the local processor memory, or each fast vector's data could be indexed to different points in the processor memory. The latter action could be used to access a rectangle of data within a larger rectangle already formatted in local processor memory; such indexing is an externally supplied function, not a part of the Adder chip.

In general, executing PBTs while scrolling is possible. 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 down scrolling) must operate continuously, adhering to a maximum time between words transferred. If the transfer becomes delayed, address calculations for scrolling will become invalid and data will transfer to/from the wrong screen location. Generally this requirement can only be met by an uninterruptable processor or a DMA controller. There is no such problem for PBTs in off screen memory. For more information, see section 4.1.2.3.

< It is recommended that the local processor compute exactly the number of words that will be transferred with a PBT command by the hardware. For a Z mode transfer, this will simply be the width times the height (in pixels) of the area being transferred; for a X mode transfer, the width calculation will be dependent on the width (in pixels), the bus width mode and the indexed X origin address. If exact calculations are not possible, too much local processor data is accommodated by harmless additional reads or writes to the FIFO; however, if too little data is transferred, the Adder chip will not be satisfied, requiring a cancel command (or more data) to continue operation. The ability to cancel a BTP operation is particularly useful if the local processor wishes to scan the bitmap data for a particular set of values, stopping when such is found; this is useful for a polygon flood operation.

3.5.4 Performance

When no scrolling is in progress, destination-only and single source rasterops are compute bound in the Adder chip; so they operate at approximately 500K cycles/sec (compute cycle rate). Rasterops with two sources are memory cycle bound, and operate at approximately 400K cycles/sec (depending on screen format). Cycles operate on either single pixels or words depending on whether the conditions for fast mode exist; if words are possible, they are either 4, 8 or 16 bits, depending on the bus width. Four compute cycles are required by the Adder chip 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 300K-400K cycles/sec, single source rasterops operate at approximately 200K cycles/sec, and two source rasterops operate at approximately 150K cycles/sec; all depending on the screen format. Partially scrolling screens (except down scrolling) will have update speeds intermediate between the non-scrolling and full scrolling speeds.

< There is one additional performance limitation that applies

to updates to a region that is in motion (scrolling) or 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 computer 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 about 8 memory cycles, depending on screen format, are required to write or read all the data at each Y address, eg. vertical vectors), the update process will not be able to pass the refresh process, thereby limiting progress to that of the refresh process. To eliminate this effect, it is recommended that tall, skinny rasterops, in regions whose height is a significant portion of the screen height, be written from bottom to top, rather than from top to bottom. 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, this limitation does not apply.

3.6 Application To Text

Normal rasterops are used to write characters. The Adder chip contains no additional hardware to assist in the display of text, but much of the standard hardware already described was inspired by text requirements. This section will describe common text procedures.

3.6.1 Font Storage And Access

Fonts are stored in an undisplayed portion of the bitmap. Ordinarily, a font need be stored only in one plane and is transmitted from that plane to itself and others when a character is written. This will result in writing characters in the bitmap that are monochromatic (all of one foreground color and one background color), as is normally desired. If multi-colored or gray-scale (for anti-aliasing) characters are desired, then the fonts can be stored across many planes. Some extra RAM will normally be available in the off screen portion of the bitmap for soft fonts; but, if ROM or additional RAM is required, this can be added to a plane because there are more X and Y address bits available than are needed for a full page display. ROM storage of fonts in the bitmap is straight forward only if a product is restricted to operation in one bus width mode or if separate ROMs are provided for each mode; this is due to the different mapping of data and address bits in the various bus width modes.

< Characters may be stored anywhere in the bitmap (though normally in off screen memory); however, a character does take up a two dimensional space in the memory, described by a height, width and an X and Y origin. Thus, font storage in the memory must be allocated in a two dimensional fashion. Normally the local processor will keep a table for the characters in each font where it can look up their origins and widths. While character origins may be on any pixel, faster operation (20-30%) will result if characters are left justified in bitmap memory word cells (16, 8 or 4 bit words), because only left shifting will be required, resulting in fewer source reads.

3.6.2 Normal Text

The display of unrotated, unscaled, fixed-pitch text could use the following example settings in the Adder and Viper 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 Viper chip 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 Viper chip logic functions, foreground, background and (possibly) source and mask registers are set for some of the desired character attributes (see further discussion of attributes, below).

4. The Adder 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 is 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 one.

The following operations could now be performed to write each character in a string:

1. The local processor would load the command register in the Adder chip to start a source 1/destination rasterop with the desired logic unit function register and CSR bank selected.

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

3. After waiting for the Adder 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 (if the same as the previous command), so that the next character will start 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 by 15 high cells on a 16 bit bus, about 20K chars/sec can be written, if fully supported by the local processor. The speed will range from 15K to 25K chars/sec for other useful character sizes. Because a full page screen contains about 6000 characters, 300 msec will be required to fill the screen with characters. Adding about 100 msec to clear the screen, but considering that an average text page really only contains about 3000 characters, about 250 msec will be required for a "new page" of average text. Obviously, if the local processor is not able to process characters as fast as the hardware can take them, then the processor will determine the overall performance.

3.6.3 Variable Pitch Text

The local processor may use any technique it wishes to compute the character spacing. Obviously, fixed spacing is useful in many applications. Variable spacing can be used to justify a line of text by spreading any extra pixels over the line; one such computation is to find the "real number" (integer and fraction) of pixels for each character (or space), and then, for each character, spacing 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 (as may be useful for compatibility with dot-matrix printers when dumping the bitmap).

A variable pitch font would be displayed with a different spacing for each character (read by the local processor from its width table). This may be accomplished by changing the destination origin and fast DX or just the destination origin alone. The former mode allows the font to be stored compactly, with the characters only occupying their width in the memory; but, the speed will be somewhat reduced because the local processor must wait for the previous rasterop to be completed before reloading the DX register and starting a new rasterop. The latter mode allows the same speed of operation as fixed pitch text, but the characters would probably need to be stored in cells the size of the largest character so that portions of unwanted characters would not be copied to the screen; of course some loss of speed will result 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 character 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 could be done by clearing the area of the text line before writing the character string.

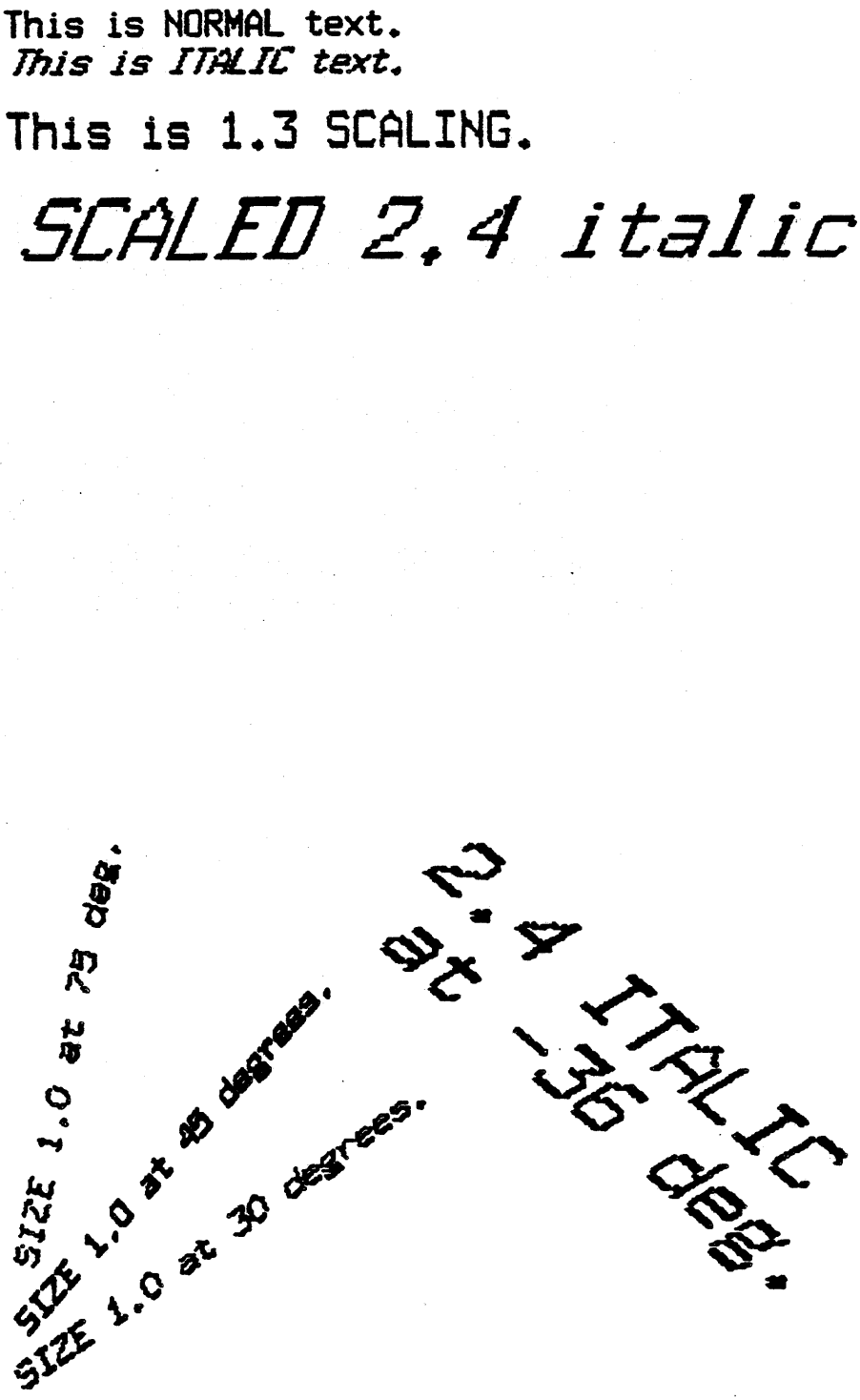
The performance of variable pitch text is similar to that described above for fixed pitch text.

3.6.4 Rotated And Scaled Text

Display of rotated or scaled text is similar to 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 want to be accounted for). An example of scaled and rotated text is shown in Figure 11.

< To insure proper tracking of an arbitrary base line, real destination origin updates could be made in the manner discussed for variable pitch text, but with an integer and fractional part for both X and Y that correspond to the correct base line angle. Use of real position updates will allow 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/second because of the varying number of pixels/character. Using figures provided in sections 2.2.3 or 3.5.4, these characters will be written at 500K pixels/sec and it



Text Scaling and Rotation Examples

Figure 11

will take 1.7 sec to fill a 840K pixel screen.

3.6.5 Character Attributes

To set the character attributes, appropriate Adder and Viper 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; although, if a permanent display list is being maintained, it may be desirable to reset all attributes periodically (maybe at 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 Adder chip command word to address one of the four logic unit function/mask complement registers in the Viper 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. 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. Another technique would be to overstrike each character with another character containing 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 Adder chip mode register may be used to disable writing, or the local processor can just skip 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 and are implemented through the color map. To enable writing the appropriate color in the bitmap planes or subplanes, a Z axis load of the Viper chip foreground and/or background registers would be accomplished with a Z axis I/D bus command; this will set all of the F/B bits in each plane or subplane to either 1s or 0s. Source patterns would then be loaded (via the Viper CSRs) into a Viper chip mask register to accomplish overstrike writing, or into the source register to

accomplish replace mode writing. Alternately, 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 and the fast and 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, sub and superscripts may be nested and angled baselines may be accounted for.

3.7 Application To Graphics And Additional Graphics Support

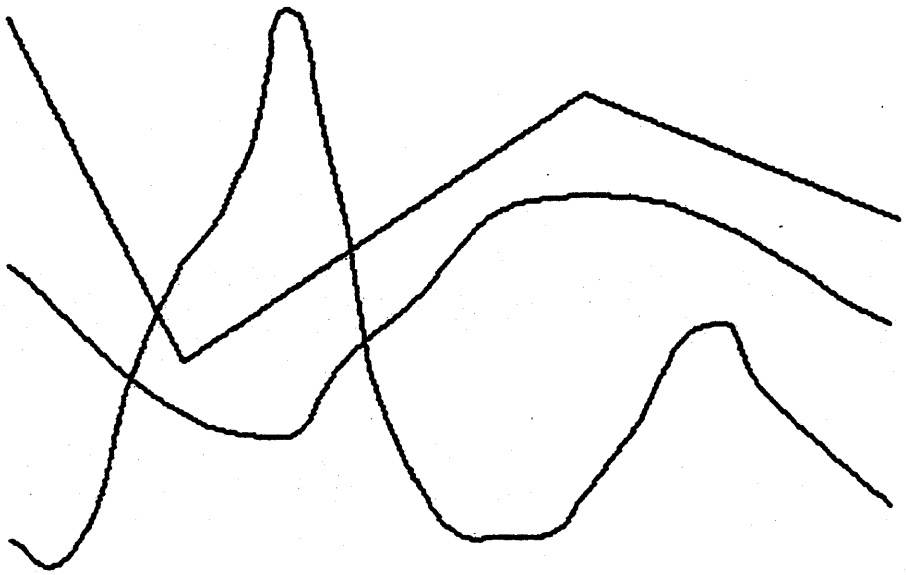
Most graphics functions use the features introduced above. Linear patterns and polygon fill add some additional features.

3.7.1 Points And Vectors

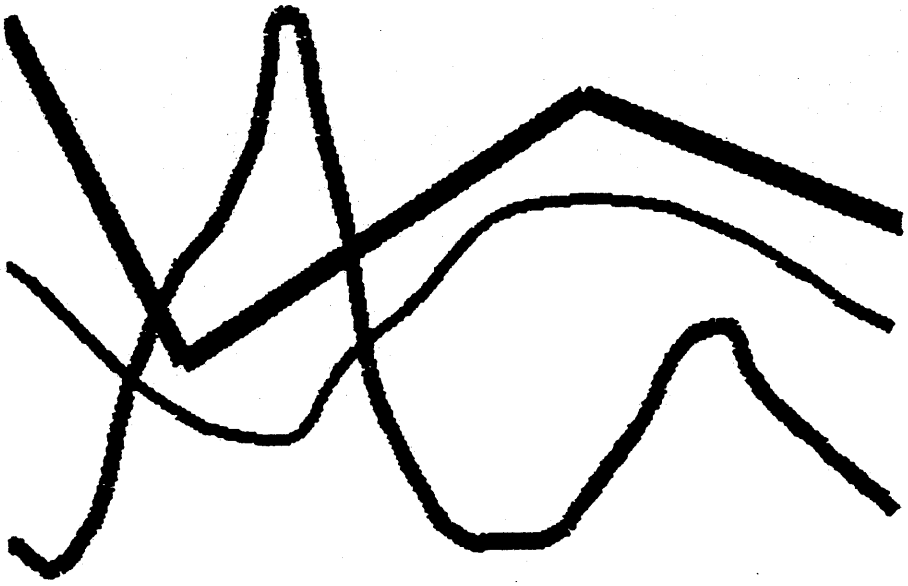
Solid color (monochrome) points and vectors are plotted as specific parallelograms, without any source operations. The source or foreground/background registers in the Viper chips are loaded with constants according to the desired color (usually with a Z axis I/D bus command). For a point, the location is loaded into the destination origin registers, the destination fast and slow vectors are loaded with a length of one (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 six deep data FIFO can always accommodate one to six words without being subject to local processor delays. While a PTBZ could be used to write one pixel, this is not possible while scrolling because the processor might not supply the data quickly enough after the PTBZ command; the single pixel rasterop described above is recommended to avoid this problem. 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. See section 3.7.4 for a discussion of applicable techniques.

Figure 12 shows examples for the various vector types described in the following paragraphs.



Single Width Vectors



Broad Vectors

Figure 12

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 one, 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 of 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).

If a broad vector is desired, the set up would be the same except to turn hole fill on and load a slow vector with a length equal to the desired thickness and a direction perpendicular to the fast vector. Care must be taken at corners to prevent gaps (seen in the figure) between vectors. Gaps may be 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. Simulation of the area traced by a rectangular brush may be accomplished by decomposing each vector segment into two parallelograms with 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.

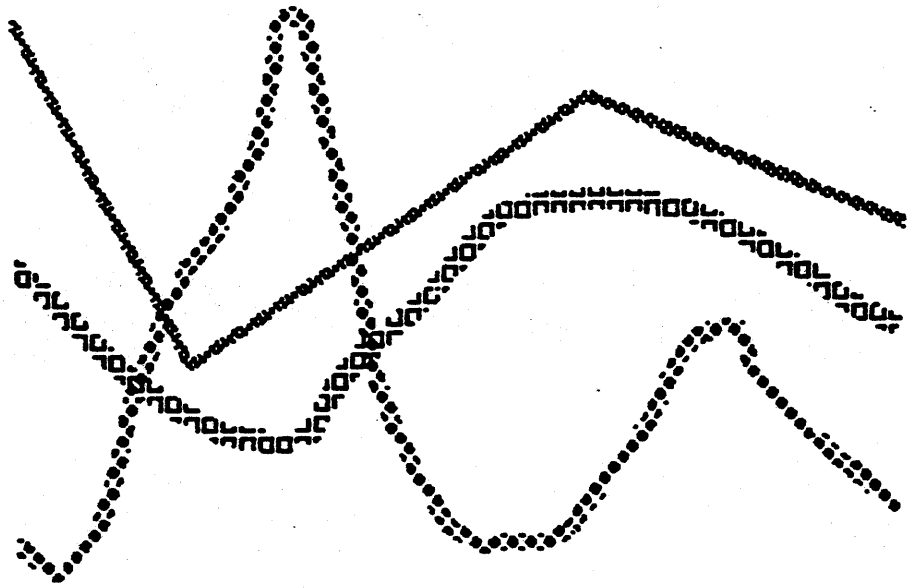
3.7.2 Shading Of Vectors - Linear And Tile Patterns

Vectors may be shaded in one of two ways (Figure 13 shows examples of both types):

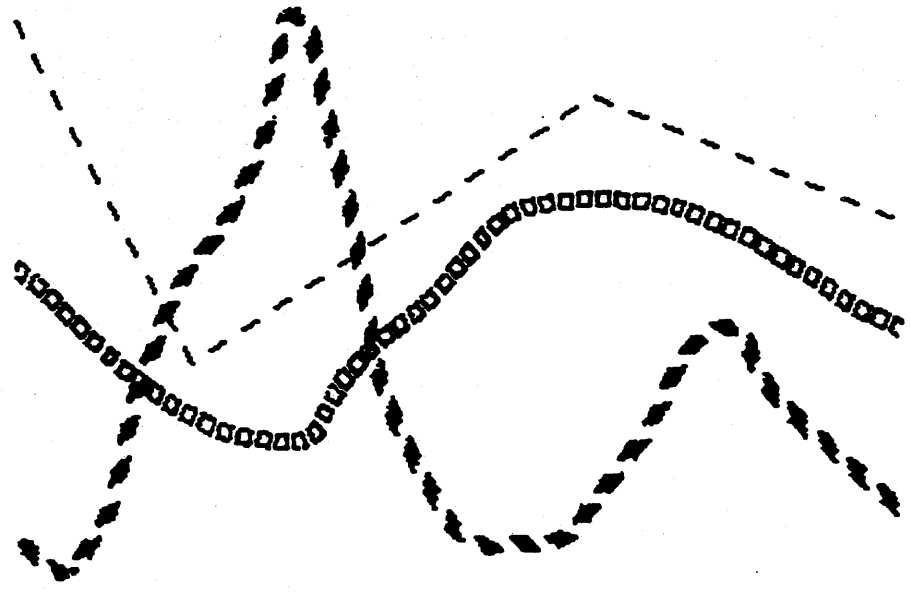
1. Inclusion of a first source set for linear pattern mode will provide linear patterns (like dashed lines, bullets or stripes) that align themselves with the direction of a vector.

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

Linear patterns are created by setting the Adder 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.



Tiled Vectors



Linear Patterns

Figure 13

Linear patterns operate in the following manner:

1. As the destination scans its fast vector, the source will provide 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 will reinitialize its fast vector at the same origin (slow vector point) last used; thus, the source will scan 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 will be incremented (as determined by the slow scaling).

4. Both the source and destination fast vectors will be 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 will be reset to the source 1 origin, and the whole source will repeat 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 will cause the source and slow scale to be initialized, also.

The result is that a two dimensional pattern is scanned by the source and transferred to the position, direction and scale of the destination. If it is desired to scale the pattern so that 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 normally are 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 prevents 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 will require the phase of the linear pattern to be reset between each vector command (see next paragraph).

< 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 are 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 (described above) and the destination are enabled; the area under the destination will be tiled as previously described. Appropriate tile patterns for texturing vectors normally will be fine grained and approximately isotropic (no stripes).

When patterned broad vectors are drawn, the same consideration must be given to gaps between vectors as discussed above for solid vectors (Figure 13 contains gaps). In addition, the phase of linear patterns may need to be adjusted; no additional consideration is required for tiled vectors.

3.7.3 Fill Mode - Polygons

In general, a fill operation places a solid or textured object on the screen whose shape is described mathematically, such as by a list of vertices. This is contrasted to a flood operation described in section 3.7.4.

To assist in the filling of 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 only be filled with solid colors (destination only), or with the second source. Through the use of 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; and, 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 will reverse at the crossing and continue 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, as described in section 3.7.3.1). This is similar to a normal rasterop, for which the first but not the last point is plotted.

To fill a whole polygon, it must be subdivided into pieces that can be directly filled by the Dragon hardware. Such a piece will be intersected by every line that is parallel to the fill axis exactly twice (except at edge intersections or when an edge is parallel to the fill axis). Another way to put this is that if the border of a piece is traversed from the top most (left most) vertex, for an X (Y) fill axis, there will be only one reversal of direction along the Y (X) axis. Figure 14 shows an example of a polygon meeting this requirement for an X fill axis.

Assume a polygon that meets the preceding conditions for an X fill axis. To fill such a polygon (with an example provided by Figure 14):

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

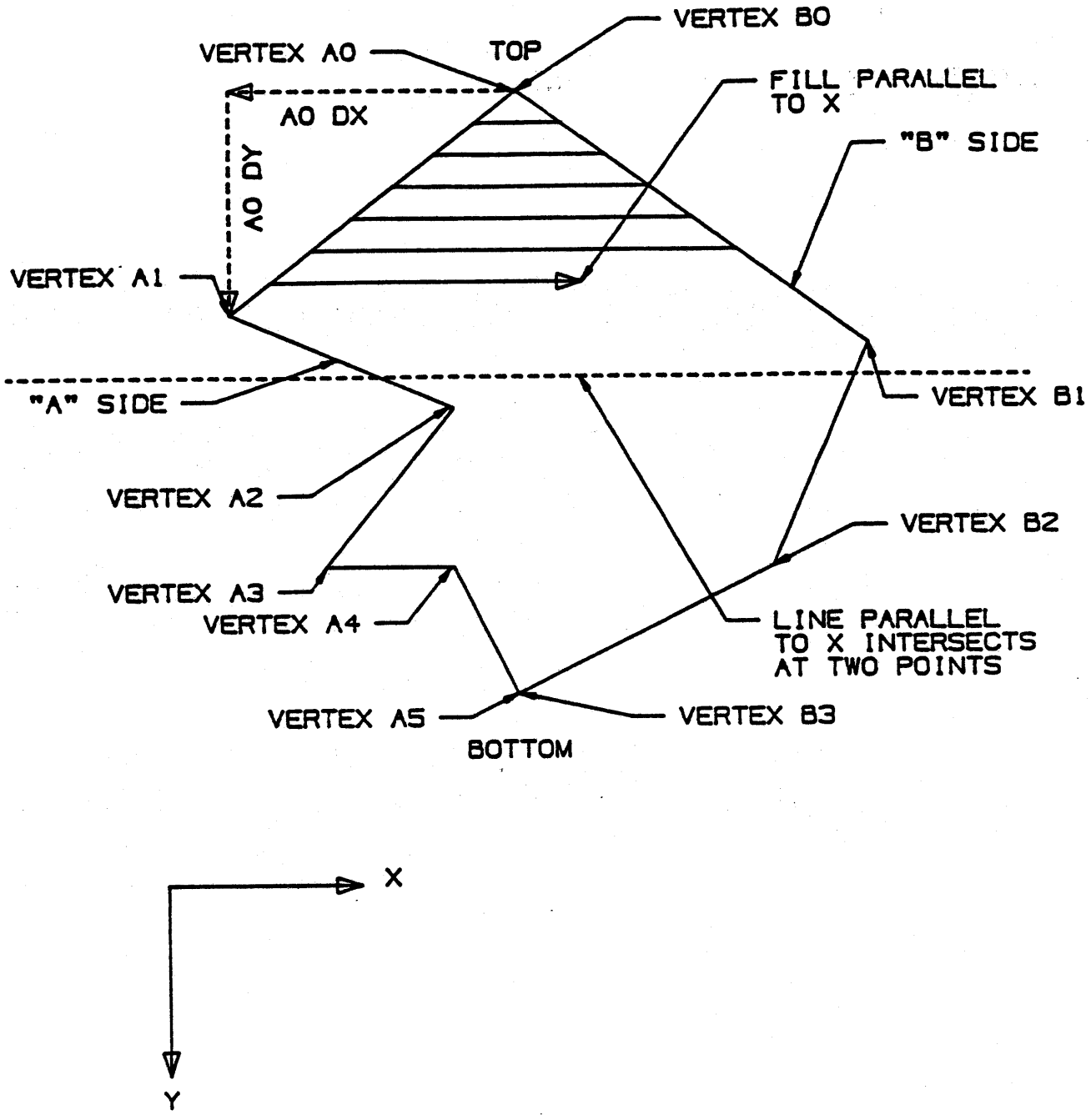
2. One side of the polygon (from the upper most point "TOP" to the lower most point "BOTTOM") is designated the A side and the other side is designated the B side. It does not matter which side is designated A or B.

3. The vector strings that represent each side of the polygon are expressed as DX and DY pairs to the next point (just like ordinary vectors), see the A0 example. After an initial origin load, all vectors are relative.

4. The vertices from both sides are sorted as a single group by increasing 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 as long as they both have the same Y coordinate, the source 1 Y origin register is



POLYGON FILL EXAMPLE

Figure 14

ignored for X fill, the roles of the two source 1 origin registers is 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 Adder 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. Then 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 Adder chip signals that it needs more data (rasterop complete flag), the local processor loads the next vector or vector pair in the list into the appropriate vector registers (A to destination slow, and B to source 1), and then loads the Adder chip command register again, continuing with the previous step until all required vectors are plotted.

11. The process is terminated when the local processor loads new data into the destination origin registers (which will cause the positions to be reinitialized) or switches the Adder chip out of fill mode.

3.7.3.1 Fill With Complement Mode

The above procedure will plot all of the points that would have been plotted by the edge vectors, plus all the points in between. However, some of the points will be plotted more than once, creating potential problems for complement mode operations (any operation that XORs or uses the complement of the destination). Fortunately, some extra work by the local processor can fix this problem, for most of the important cases, by forcing all pixels that would be written an even number of times to be written an odd number of times (ordinarily, writing doubled pixels a third time).

< Doubling will occur each time an A or B vector is loaded, AND the previous vector or the vector on the other side (whether or not it has also expired) was (is):

1. "inbound" (approaching the other vector), and

2. has its major axis parallel to the fill axis, and

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

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

   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 above conditions, any vector that expires and has an inbound slope greater than or equal to 2:1 (major:minor) with its major axis parallel to the fill direction will always cause doubling, and any other expiring vector will never cause doubling. This can be seen by substituting major/2 for the minor axis in the above equation. For a vector that is not expiring, doubling will always occur for slopes greater than or equal to 2:1, doubling will never occur for slopes less than or equal to 1:1, but the equation must be evaluated 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, then this fix is applied to both vectors.

< This extra procedure for complement mode will correct 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); and, if an inbound vector of the stated slope is followed by an outbound vector that also meets the slope condition, the pixels that are common to the vectors but which lie outboard of the vertex will be doubled. If necessary, these conditions can be computed by the local processor and corrected later by tripling.

< As an example of correction of doubling of pixels, refer to the previous example in Figure 14 and assume that the 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, and 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 polygons are butted together so that one shares some edges with a previous polygon, the adjoining edges will be written twice; but, because all of 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 just the edge vectors again (in the same direction as when filling), including the last point on the last vector.

3.7.3.2 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 base line 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 non-zero. 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 two-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. In addition, if the vector path crosses the baseline and then reverses direction (parallel to the baseline, such as when filling a circle), the baseline will be written twice, as if two polygons had been butted together, and the baseline will need to be drawn again as a single vector to correct for the even number of writes.

3.7.4 Polygon Flood

A flood operation colors or textures the interior of an area whose outline is previously described in the bitmap. This is contrasted to a fill operation described in section 3.7.3. No explicit support for a polygon flood function is provided by the Adder 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 Adder chip, but only the Viper chips see the memory data.

Flood can be implemented using the following commands:

1. the local 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 Adder chip,

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

5. the flood algorithm would then step to a point on the next scan or continue at a previous seed point, in the normal manner.

The speed of this operation is determined by the BTP scanning time plus the overhead time of the local processor (which could be significant if it takes more than 2 usec for each comparison or if the flooded area is narrow). The drawing time is generally not significant compared to scanning and decision making. If a one plane Dragon system is used, if the outline of the flood area is described in a single plane, or if the boundary is described by a single color, the bitmap can be scanned with an X mode bitmap to processor transfer, thereby checking 16 bits (or the width of the bus) on each transfer; using this technique, an empty full page screen (1024 x 864) has been flooded in less than 2 seconds.

3.7.5 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 all of a rectangle is copied. The source 1 is indexed and offset and so may be from either on or off screen. If a source 2 is used it must be from off screen.

To move just an object, that is to alter only the destination pixels within the boundary of the object, it is necessary for the hardware 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 previous two paragraphs described the movement of color objects (objects defined in more than one plane). It is also possible to move monochromatic objects (ones specified by just a foreground shape), assigning a foreground and background color to them at the time that they are moved. This type of operation has already been described for the writing of text.

4 DRAGON CHIP REGISTERS AND COMMANDS

This section defines the interface from the local processor to the Adder and Viper chips in detail. General concepts and terminology were introduced in section 3. There are basically two types of commands to the hardware: direct commands to the Adder chip, and I/D bus commands that are passed through the Adder chip, to the Viper chips and other I/D bus devices.

4.1 Adder Chip Registers And Commands

All Adder chip commands are the result of Adder chip register loads by the local processor or an external DMA device. Before presenting the Adder chip commands and interface, it is appropriate to define the Adder chip registers.

4.1.1 Adder Chip Registers

The Adder chip contains many loadable registers that hold parameters for rasterops, set system timing, and control the operation of the chip. Unless otherwise specified, registers are write only and numerical registers contain 14 significant bits (including sign, the two MSBs of a word are ignored, except for test purposes) in two's complement representation. The loading of some registers causes specific additional actions to be taken by the Adder chip; these actions will be described with the corresponding register.

There are different types of coordinates implied by the various position registers. "World coordinates" refers to the coordinate system of the image as viewed by the local processor 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 increasing X to the right and increasing Y 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.

Because the registers may be loaded by a DMA controller through the register address counter, thereby loading 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 device's data list.

The register addresses (or the address of the first of a group of registers), in HEX and enclosed in [], precede the

register descriptions in the following list. Where specified, bit assignments (or the LSB of a group of bits) are enclosed in <>.

4.1.1.1 Interface Control Registers

[0] ADDRESS COUNTER (ADCT) - This address controls a counter that provides indirect access to the Adder chip registers for use by a standard DMA controller. 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 six bits of the data replaces 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 pointed to by the address counter is accessed and the counter is always subsequently incremented (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 to assert the request or interrupt output pins. A one in either of these registers enables a status bit; if a bit in the status register is a one and that bit is enabled by one of the enable registers, the output pin corresponding to the enable register will be asserted. Both enable registers are cleared by assertion of the -INIT pin on the Adder chip.

The request pin and register are provided to control data requests to a DMA controller that may be present on the local processor's bus. In general, the DMA controller would probably only set one of these request conditions at a time because it will be waiting for one specific event before passing the next data word.

The interrupt pin and register are provided to control interrupt requests to the local processor.

After receipt of an interrupt, the local processor will need to read the status register to determine the interrupting condition(s), if more than one interrupt is enabled. Actually the request and interrupt functions are implemented identically so, if a DMA controller is not used, both outputs may be used as separate interrupt requests.

[3] STATUS (STAT) - The status register provides various indications of the internal progress of the Adder chip. The conditions that set and clear each bit are identified in the following list. This register is readable 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 zero at this bit position; unaffected by a one 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 by the local processor even if the second event occurs before the first event is detected. 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 (see section 4.1.1.5. The -INIT pin will clear 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 zero at this bit position; unaffected by a one at this bit position or by values of other bits. This bit differs from the Vertical Blank bit in that the latter is set at the beginning of the vertical blank interval.

<2> Rasterop Initialization Complete - Set when initialization of a rasterop or processor/bitmap transfer (PBT) is complete, indicating that it is safe to load the source 1 and destination origin registers, or to load a new (but not different)

rasterop (but not PBT) command (see section 4.1.2.2. Cleared when any rasterop or PBT command is loaded. Set by a cancel command or the -INIT pin.

<3> Rasterop Complete - Set when rasterop or PBT address calculations are complete and no further command is pending, indicating that 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 PBT) command (see section 4.1.2.2. Cleared when any rasterop or PBT 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 or PBTs have been used, indicating that 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 will not go high until the IDD FIFO is also empty, indicating that all data transferred by the BTP has been picked up by the local processor or DMA controller. Cleared when any rasterop or PBT command is loaded. Set by a cancel command or the -INIT pin.

<5> I/D Data Receive Ready - This bit is set if the I/D data FIFO has an data word available for reading; it is cleared if the FIFO becomes empty. When a rasterop, PBT or cancel command is loaded to the command register, or bus initialization occurs, the FIFO is initialized and this bit will be cleared. Note that this bit will be set any time that data is in the FIFO, such as during PTB 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 Adder chip:

a. When a rasterop or PBT command is loaded to the command register, the FIFO is initialized and this bit will be 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; it is cleared if the FIFO becomes full. (While this bit responds during rasterop commands, I/D data is not used in these commands and any data loaded will remain 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, and it is cleared when an I/D command is loaded to the command register. This bit should also be checked before loading another command following an I/D command; however, if at least three major cycles (one major cycle plus a sync cycle, about 3 usec) passes 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 will be 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 or PBT is not in progress (see IDD register), or the -INIT pin is asserted, the FIFO is initialized and this bit will be set.

<7> I/D Scroll Data Ready - This bit is set if a new data word may be loaded to the I/D scroll data register, and it is 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 a picking algorithm by indicating that a rasterop was at least partly inside the clipping rectangle. The accumulation of clipping information during a rasterop will be

different for different bus width modes, although the end result will be the same in all cases. Clipping information for a rasterop will be 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 zero in each corresponding clipping bit position to be cleared and a one in positions to be unaltered; zeros or ones in other bit positions will have no effect on these status bits. Clearing of the clipping status bits during a rasterop will give unpredictable results; these bits should only be cleared after the address complete status bit is asserted.

<D> Vertical Blank - Set at the start of the vertical blank interval. Cleared by writing a word to the status register with a zero at this bit position; unaffected by a one 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 blank 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 and would force a load to the IDD if this address were accessed through the address counter.

[7] I/D DATA (IDD) - When a rasterop, PBT or cancel command is loaded into the command register, or the -INIT pin is asserted, the six deep FIFO at this register address is cleared. During PBT 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 bus 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 left most pixel in the word.

Whenever a load to the I/D data register occurs AND the address output complete status bit is set (a PBT or rasterop command is NOT in progress), the FIFO is cleared as the data word is loaded so that 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 will continue 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 following the mode register; the duplicate address is provided to ease access through the address counter because the command register is frequently written following 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
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) - Adder chip command register. See section 4.1.2 for a detailed description of the commands and various effects of loading this register. The command register is used for all commands by the update process. Some combinations of bits will result in undefined operation. When performing a series of operations (eg. 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 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/bitmap 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.

4.1.1.2 Scroll Registers

During active scrolling, scroll registers must be loaded by the scroll process before the START of vertical blank. The Adder chip performs various functions during vertical blank, 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 and would force a load to the IDS if this address were accessed through the address counter.

[C] I/D SCROLL DATA (IDS) - This contains data to be transmitted to the I/D bus during scroll process I/D

commands; the last word loaded to this register will be 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 will be transmitted on the I/D bus without interference with any update activity that may be in progress, assuming that only scroll related registers in the Viper chips are addressed, and that separate chip selects are used. An output pin from the Adder chip can enable an I/D bus chip select register that is reserved for these scroll I/D commands.

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

[F] SCROLL X MAX (PXMX) - Right boundary of 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 only be specified to a multiple of four pixels.

[10] SCROLL Y MIN (PYMN) - Top boundary of scroll region. The upper most pixel in the scroll region.

[11] SCROLL Y MAX (PYMX) - Bottom boundary of 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) - Y device coordinate specifying the scan that, when being displayed, will cause the pause status bit to be set, or a second pause event to be queued (see 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 local processor during down scroll; ranges between 0 and the height of the displayed portion of the bitmap memory (the same value stored in the Y limit register, minus one, see below). The offset value loaded to this register should be that which will be in effect when the following frame is complete, 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 will cause the region selected by the scroll boundaries to be scrolled (by the vertical distance loaded to this register OR the horizontal distance loaded to the scroll constant registers in the Viper chips) during the following frame. If no value is loaded during a frame, no scrolling will occur in the next frame.

<0> 12 Bits. The vertical magnitude (unsigned distance) of the scroll. If a non-zero X scroll constant is specified in the Viper 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 Viper chips should also be set to down and left.

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

4.1.1.3 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 which will be automatically loaded into the new registers at the start of the next frame; the content of these registers is not destroyed by this load so that, if no scrolling will

take 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 will be 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. An appropriate interlock for these two processes is described in section 3.3.4.1. All of 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 and then 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 only be specified to a multiple of four pixels.

[1D] CLIP Y MIN (CYMN) - Top clipping boundary. The upper most 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 and thus are not usable as a drawing function, but merely to contain the image within the borders of the scroll window. They only affect whether or not destination data is written and the setting of the clipping status bits; all rasterop calculations are still performed.

[1F] SPARE

4.1.1.4 Rasterop Control Registers

[20] FAST SOURCE 1 DX (FSDX) - Fast extent for the first source. This can only be in the + or - X direction, so 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, so 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 Adder 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 will reset 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 Adder 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 zero or down scaling if a one. The magnitude is specified in the remaining 13 bits with the binary point preceding bit 12. The Adder chip adds 0.0000000000001B to each scale factor. The scale factors are ignored by the Adder chip unless both the source 1 and destination are enabled. See section 3.5.1.2 for a complete description of scale factors.

[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, and therefore must be off screen. If tiling is enabled, the source 2 origin must be on a word boundary (dependent on bus width) 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). In 8 and 16 bit bus width modes, the tile width must not be set for less than the bus width; of course, one may have a narrower tile by repeating it within the larger

pattern cell.

<3> Reserved. Should be 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.

See section 3.5.1.1.2 for more information on these registers.

4.1.1.5 Screen Format Control Registers

An example program that calculates the contents of the following registers from general timing requirements can be found in the routine VDIsetscan() in the file DRAGONLIB.C, referenced in section 1.2.

[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 Dragon system. Each register determines the time of one vertical event, such as asserting vertical blank; the event is specified by the two MSBs (bit 12 and 13). The low 11 bits determine the time of the event (scans for vertical sync events, and scans minus one for other events), starting from vertical unblank. 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, so that 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 blank is deasserted (set low) when the vertical period is reached. The bits of the Y scan count registers are assigned as:

<0> 11 Bits. Time of event from deassertion of vertical blank, in scans for vertical sync events, and scans minus one for other events. The end vertical period event is set to the number of scans in the frame, minus one.

<B> Reserved, should be zero.

<C> 2 Bits. Event.
00=End vertical period. Set vertical blank low in following scan.
01=Set vertical blank high.
10=Set vertical sync low.
11=Set vertical sync high.

The vertical sync signal is available at an output pin for use in systems that want separate vertical and horizontal sync; this signal also chooses between two events in the X scan counter that provide the variable width horizontal sync pulse needed in a composite sync system. When separate syncs are required, the vertical sync polarity is programmable and the "010" or "011" X event should be removed from the X scan count sequence, so that the horizontal sync width is unchanged. When using composite sync, the vertical sync interval must be delineated by the high state of the vertical sync signal, allowing the longer horizontal sync time to be determined by the "100" or "101" event in the X scan counter (see below).

An example assignment of events for a composite sync application is: YCT0 will set vertical blank high, YCT1 will set vertical sync high, YCT2 will set vertical sync low, and YCT3 will set vertical blank low and restart the vertical counter.

System sync request will be generated by the Adder chip in the scan prior to deasserting vertical blank (scan prior to the first displayed scan); the exact timing of sync request within that scan is determined by the X scan count registers, 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, 64 or 32 in 16, 8 or 4 bit bus width mode, respectively. To guarantee refresh of the dynamic bitmap memory, enough major read cycles must be programmed to display 512 pixels on each scan; this will allow 4 row addresses to be refreshed on each scan in 16 bit bus width mode, 8 row addresses in 8 bit mode, and 16 addresses in 4 bit mode (see bit <8>, below). At typical scan rates, memory refresh will be accomplished approximately every 0.5 msec, plus any gap occurring during vertical retrace. See the discussion of XCT- and XL registers for required relationships to this register.

<6> 2 Bits. Bus width mode must be programmed here and in the Viper 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 Viper scroll boundaries. The assignment of X and Y address bits to the Adder chip output pins is dependent on bus width and the memory configuration bit below. The assignments are covered in section 5.6.1.
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 1024" mode, 8, 16 or 32 row addresses are refreshed on each scan in 16, 8 or 4 bit modes, respectively. In "greater than 512" mode, 4, 8 or 16 row addresses are refreshed on each scan in 16, 8 or 4 bit modes, respectively. The number of pixels read in each scan is the (number of refresh reads programmed in XCON * the bus width * 8).
0=1024 <= (number of pixels read from memory/scan) < 2048
1= 512 <= (number of pixels read from memory/scan) < 1024

NOTE

If 1 Mbit of memory (16 64Kx1 memories) is used on the Dragon system (in 16 bit mode), the maximum addressable range in both X and Y is 1024 pixels (actually only 992 in Y if 32 scans of memory are reserved for down scrolling). To

extend these ranges, it would be necessary to add another 1 Mbit, allowing a 2048H x 1008V or 1024H x 2032V memory range. Of course, still more memory may be added, if desired, up to 8192H x 8192V (or the limits of bus loading for the desired system speed).

[36] X LIMIT (XL) - Width of that portion of the memory READ by the screen refresh process (could be more than that which is 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, 64 or 32 in 16, 8 or 4 bit bus width modes, respectively.

[37] Y LIMIT (YL) - Height of the portion of memory read by the screen refresh process, 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; while the memory between the right edge of the screen and X limit (if any) can be used, this is only possible 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 Dragon system. Each register determines the time of one

horizontal event, such as asserting horizontal blank; the event is specified by three high order bits. The low 11 bits determine the time (in 16ths of a major cycle, nominally 60 nsec, but some events are not allowed to be programmed to this precision) of the event, starting 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:

<0> 11 Bits. Time of event from a reference time 3-1/4 major cycles preceding 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 two events may occur in the same 1/4 of a major cycle; that is, no two events may have times that are equal in all but the low two 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 one 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 zero in this bit.

<F> Test function in XCT6, normally 0.

The period must be set to Nx16-4. The integer multiple "N" must be equal to twice the number of read cycles in XCON plus at least two (an integer number of major cycles); a larger "N" will lengthen the horizontal retrace interval by units of one major cycle. N must be at least 10.

Horizontal blank 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 Adder and Viper chips and in the video output circuitry; the Adder chip produces the 3-1/4 major cycle delay mentioned above, and the Viper chip produces an additional 10 Alpha clock delay (5/8 major cycle). Horizontal blank would be set high after the desired number of pixels have been displayed on a scan.

The polarity of horizontal sync is programmable. Horizontal sync starts when set high by the "101" event (or low by the "100" event). Horizontal sync ends when set low by the "010" event (or high by the "011"), except when vertical sync from the Y scan counter is high, when horizontal sync is set low by the "100" event (or high by the "101" event). The "100" or "101" events will always change horizontal sync; one of these events should be set to occur after the "010" or "011" event when composite sync is generated, and should be used alone (without the "010" or "011" events) when separate syncs are generated.

System synchronization is generated by logic external to the Adder chip "shortly" after the sync request event occurs; sync request always occurs during the scan preceding the end of vertical blank. "Shortly" is dependent on the implementation of the timing logic external to the Adder chip. Usually the sync interval will start at the beginning of next major cycle following sync request; with such an implementation, this event must be programmed in the range Mx32+4 to Mx32+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. This must also be set for a time during the scan when no other X scan counter events will occur during the following sync interval (two major cycles); event times affected are in the range Mx32+21 to (M+1)x32+23, inclusive. The two low bits of the sync request event time are not significant (only 240 nsec intervals allowed).

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

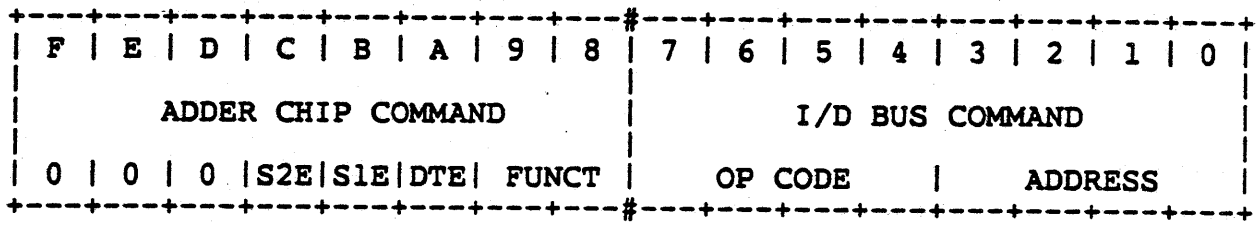
[3F] SYNC PHASE (SYNP) - 11 Bits. The value loaded in the horizontal sync counter at each system sync time (once per frame). This must be set for (M+1)x32, where M is defined for the sync request event, above (this assumes

the required sync interval of two major cycles). The low five bits of this register are ignored (not programmable).

4.1.2 Adder Chip Commands

All commands to the hardware are passed to the two command registers in the Adder chip (except scrolling, which is initiated by loading the scroll constant register, as previously described). 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 three MSBs are reserved for test functions), as shown below:

COMMAND REGISTER FORMAT



The low byte contains the I/D bus portion of any command, consisting of an opcode nibble and a nibble that usually specifies a register address; I/D instructions are fully described in section 4.2.2, but required and optional bit settings are described in this section as they apply to specific commands. The next 2 bits select one of four functions (I/D bus, processor/bitmap transfer (PBT), rasterop or cancel) and the next 3 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; thus, the I/D scroll command will not be further described.

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

All commands (except cancel) involve the Viper chips. Only those Viper chips that are selected by the chip select register will respond to commands on their I/D bus inputs and/or will allow writes to their associated memory planes. (Participation in scrolling is determined by the setting of the scroll enable bit in each Viper chip, as described in section 4.2.1.) When multiple Viper 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 Viper chip is to be active (such as a unique register load), only the one chip select is enabled. A separate chip select register is used for loading Viper chips when using the scroll I/D command register.

Chip select registers are implementation dependent, but normally, they will have an unencoded enable bit for each Viper chip in the system. The chip select registers may be connected to the I/D bus for loading with an "external" I/D bus load (see

section 4.2.2.1), or they may be connected directly to the local processor bus. If the chip selects need to be changed by a DMA process operating through the address counter of the Adder chip, they would have to be connected to the I/D bus.

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 when the first is done. 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 (address output must be complete before a new command is loaded). The ability to queue (one) repeated command is used to reduce register loading overhead between rasterops of a series, such as when displaying a text string. PBT commands cannot be queued in this way because loading a new PBT command will clear the IDD FIFO.

4.1.2.1 Register Loading

Loading of Adder chip registers does not require the execution of any command. These 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, as previously described in the description of the status register in section 4.1.1.1. Appropriate status conditions will be reviewed below, as applicable.

All Viper chip registers and external I/D bus devices are loaded using the Adder chip I/D command. Although any I/D bus instruction could be transmitted with this command, only these register loads are permitted. No reading or active cycles are allowed; these are controlled by the Adder chip as part of rasterops. See section 4.2.2 for a complete description of the enclosed I/D bus commands, but a brief description is given below.

ID BUS VIPER REGISTER LOAD COMMAND

0	0	0	0	0	0	0	1	1	0	0	ADDRESS

ADDRESS specifies the desired Viper chip register. Only those Viper chips whose addressed registers are to be changed should be chip selected. Viper chip registers are described in section 4.2.1.

ID BUS Z AXIS VIPER REGISTER LOAD COMMAND

0	0	0	0	0	0	0	1	1	0	1	0	REG	Z BLOCK

REG specifies the register to be loaded:
00=Source register.
01=Foreground register.
10=Scroll fill register.
11=Background register.

Z BLOCK specifies one of the four possible groups of planes or sub-planes whose values are specified by the 16 bit I/D bus data. All viper chips pertinent to the Z axis load should be chip selected.

ID BUS EXTERNAL REGISTER LOAD COMMAND

0	0	0	0	0	0	0	1	0	EXTERNAL OUTPUT INSTRUCTION

This instruction will be ignored by all Viper 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, but the Adder chip will always source data during the I/D bus data transfer cycles. The I/D bus instruction 00H is reserved for the Adder chip to transmit during idle I/D bus cycles.

Any of these three instructions 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 PBT commands, see below), 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.

In order to avoid interference with update operations, only the scroll process should load the following registers in the Adder 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 Adder chip registers may be loaded by the scroll process with the necessary synchronization between the scroll and update processes, as described in section 3.3.4.1:

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 only load the following registers in the Viper chip:

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

4.1.2.2 Rasterop Command

All rasterop functions were described in section 3. 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 changing any registers; as outlined below. All functions are invoked by the command:

RASTEROP COMMAND

0	0	0	S2E	S1E	1	1	0	1	1	FNC SEL	NOP	CSR	NOP

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 setting of the status bits). The I/D byte specifies the active cycle for the rasterop (see section 4.2.2.2). FNC SEL is a two bit code selecting one of the four logic unit function registers in each Viper chip. CSR specifies the bank of CSRs to be referenced during the source and destination cycles; the Adder chip fills in the last two CSR address bits according to the cycle in progress. The Viper 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; these are fully described under the status

register in section 4.1.1.1, but are reviewed here. 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 (see bitmap to processor transfers, below). Normally, only one of these status bits is enabled at a time.

4.1.2.3 Processor/Bitmap Transfers

Processor/bitmap transfers (PBTs) are similar to rasterops, except that either the source or destination is the local processor bus. They use the same rasterop status bits for control of the process, with the addition of two I/D data buffer bits, described below. A general discussion of PBTs can be found in section 3.5.3.

All data for a PBT is transferred through the I/D data FIFO (IDD). This six deep FIFO is cleared when any PBT (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 PBT 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 PBTs because the Adder chip ignores the scale registers unless both the source 1 and destination are enabled; PBTs never enable both.

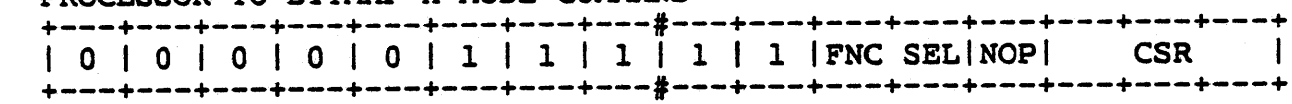
When PBTs are used during scrolling, data must be transferred continuously to prevent computed addresses in the Adder chip from becoming invalid. The local processor or DMA controller must transfer two words every horizontal scan time; this data rate may be a little more conservative than necessary.

4.1.2.3.1 X Mode PBT

In an X mode PBT, data from one plane is transferred to/from the local processor bus with 16 bits adjacent in X occupying one word (in 16 bit bus width mode) followed by additional X words and then by additional scans. The LSB of a word is the left most pixel in the displayed word and the MSB is the right most pixel. In 8 and 4 bit bus width modes, only the low 8 or low 4 bits, respectively, are significant (there is NO packing performed by the Adder chip to build words from bytes and nibbles). The area

of the bitmap that is involved in a transfer of data to or from the local processor is determined by setting the rasterop destination and source to span an unrotated rectangle.

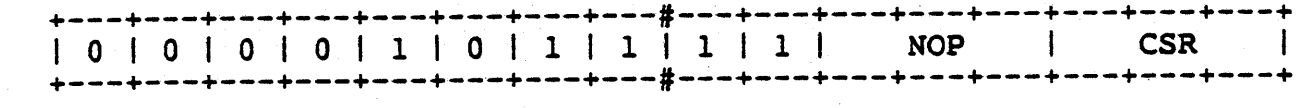
PROCESSOR TO BITMAP X MODE COMMAND



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 plane(s) being transmitted to is controlled by 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 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 will be masked; similar masking will occur 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.

BITMAP TO PROCESSOR X MODE COMMAND



The CSR bit has the same meaning as for rasterops, except that any source CSR location can be used to select the Viper chip that talks on the I/D bus (alternately, if more than one Viper chip is 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 will be determined by the low 4 (or 3 or 2) bits of the destination origin; the first bit of the source will be shifted to the bit addressed by the destination origin (just like any rasterop). Like the masked bits on each fast vector of the PTB command, the corresponding bits (those not in the "destination") will be 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.

4.1.2.3.2 Z Mode PBT

In a Z mode PBT, all the bits from one pixel location of up to 16 planes (or subplanes) are transferred to/from a local processor bus word, followed by additional pixel data along the fast vector and then by additional scans. If a system has more than 16 planes, additional 16 plane blocks (Z blocks) can be transferred with additional PBT commands. The bitmap area is again determined by an unrotated rectangle (does not have to be, but should be) 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.

PROCESSOR TO BITMAP Z MODE COMMAND

0	0	0	0	0	1	1	1	1	0	1	0	F/B	1	Z BLOCK

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.

BITMAP TO PROCESSOR Z MODE COMMAND

0	0	0	0	1	0	1	1	1	0	1	1	0	0	Z BLOCK

The plane address, the Z BLOCK address and/or the chip select register are used to select the Viper chips which 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.

4.1.2.4 Cancel Command

CANCEL COMMAND

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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 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 blank status bits are not affected.

During execution of the cancel command, the "I/D data transmit ready" status bit will be 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 4 major cycles of delay (this will normally be the case).

4.2 Viper Chip (I/D Bus) Commands

The instruction/data bus is used to control all of the Viper chips and any external devices connected to it. The I/D bus also provides for the exchange of data between Viper chips or from Viper chips to the local processor (via the Adder chip). Control of external I/D bus devices is covered in section 5.4.1 of this specification because these devices can be product specific. External devices might include the chip selects for the Viper chips (which control the Viper chip's participation in I/D bus transfers and rasterops), or the video color map. Two chip select registers are selected by the Adder chip, one for the scroll process to access the Viper chips and one for the update process.

4.2.1 Viper Chip Registers

The basic width of a Viper chip register is 16 bits, although many registers require fewer than this. All of the Viper chip registers are writable and none are readable. The register address (or the address of the first of a group of registers), in HEX and enclosed in [], precedes the register descriptions in the following list. Where specified, bit assignments (or the LSB of a group of bits) are enclosed in <>.

[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 Viper chip behave as 1, 2 or 4 planes. The effect of resolution mode on the source register can be disabled.

The resolution mode may be changed at any time so that different regions may have different resolutions; but, the external color map must also be able to change its configuration dynamically as each region is displayed to make resolution changes useful. If different resolution modes are used in different regions, the scroll process will not be able to use a Z axis load of the fill register.

<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 Viper 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 of

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 will only move by a multiple of two bits. The Viper chip acts as two planes for Z axis operations, receiving or transmitting the two bits corresponding to its plane address and the next more significant bit; the plane address must be programmed to an even value.

10=This setting will give 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 of the 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 will only move by a multiple of four bits. The Viper chip acts as four planes for Z axis operations, receiving or transmitting the four bits corresponding to its plane address and the next three more 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 Viper chips to keep the bits in their proper subplanes because the Adder chip cannot supply each Viper chip with a different shift constant (each Viper chip may be programmed for a different resolution). On the other hand, this effect cannot be provided for scrolling because the error caused by truncation must not be accumulated and because the scrolling region may not be the region for which the resolution mode has been set. Horizontal scrolling should only be 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. All Viper chips and the Adder chip must be set to the same bus width. The control of the barrel shifter is adjusted so that, combined with Adder chip actions, the control of all rasterops and scrolling is independent of the bus width (almost, exceptions are: X mode PBTs, source 2 tile origins, and Viper scroll boundary registers; also, the amount of bitmap memory present and the speed of operations in fast mode will change, of course). The video bus speed is adjusted to compensate for the

reduced number of pixels to be displayed in one frame time.

<2> 00=4 Bit bus width. The video bus outputs four bits every 240 nsec.
01=8 Bit bus width. The video bus outputs four bits every 120 nsec.
10=This setting will give undefined results.
11=16 Bit bus width. The video bus outputs four bits every 60 nsec.

[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, the this nibble is the magnitude of the scroll minus one; 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 non-zero Y scroll constant is programmed in the Adder chip, this nibble must be zero. If the region being scrolled contains some Viper chips set for low resolution, the low order one or two bits of this nibble should be zero for left scrolls or all ones for right scrolls.

<4> Horizontal scroll direction.
0=Left. Also use "0" for up, down and disabled scrolling. If a non-zero Y scroll constant is programmed in the Adder chip, this bit must be zero.
1=Right.

<5> 0=Disable scrolling (horizontal or vertical) of all data accessed by this Viper 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 Viper 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, then different update regions could have different plane address arrangements. See section 3.4.2 for additional restrictions.

<0> 4 Bits. Bit address within Z axis block (0 to 15). Addresses the bit (or low order bit in a low resolution Viper chip) on which the Viper 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 Viper chip.

<4> 2 Bits. Z axis block address (0 to 3). This is needed if more than 16 planes or sub-planes are to be addressed in the system. These are essentially the high order bits of a 6 bit plane address, but separate blocks must be addressed with separate Z commands.

[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 Viper 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), after the latter has passed through a complementer and resolution logic, and 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 (see resolution mode register description, above, for discussion of effect).
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 left most pixel in a word, as viewed on the screen and the MSB is the right most pixel. All four of the 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) so that only those bits are modified that are selected by all three masks (a one 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 masking 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 that will be inserted into memory words to fill the blank space created by scrolling. Normally a solid color fill is desired and will be provided with a Z axis load; if loaded directly, care should be taken to set all of the bits corresponding to one subplane to the same value. This register is double buffered; 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 Viper chips must be told which bits of the left most word and the right most word are actually in the scroll region. The Viper chip can be programmed to place the region boundaries on any bit position (for possible future extension), but the Adder chip cannot specify the boundary closer than a multiple of four; therefore, only those boundaries should be selected that select groups of four bits. When loading these registers, only the low 8 or 4 bits of these registers are significant in 8 or 4 bit bus width modes, respectively. These registers are double buffered; 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 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, then only bits from the left edge through the right edge are clear.

[D] RIGHT SCROLL BOUNDARY - 16 Bits.

All bits are clear from the LSB (left edge) through the bit corresponding to the right most pixel that is to be scrolled in the word that contains the right edge of the region (all other bits set), unless the right boundary is between words (LSB not scrolled) or both edges are contained in the same word, in which case all bits are set.

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

[F] FOREGROUND COLOR - Selected by a one 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 left most pixel in a word, as

viewed on the screen and the MSB is the right most pixel. Both of the 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 Viper chip control store RAM controls the transfer of data within the Viper chip and to/from other I/D bus devices during rasterops. Addressing of its contents during rasterops is controlled by the Adder 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 Adder chip command register.

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

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

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

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

[17] Reserved

NOTE

These assignments of CSR functions are defined by the Adder chip. The Viper chip does not assume any 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 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 (ie. 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 that 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.

NOTE

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

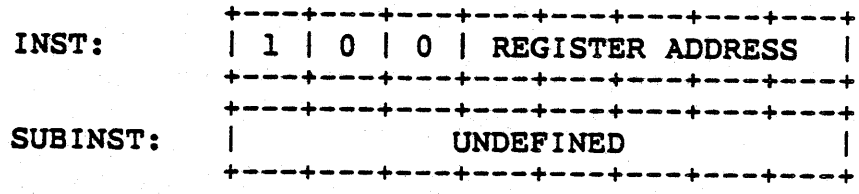
4.2.2 Instruction/Data Bus Instructions

This section is included to help with the design of I/D bus devices and is not required for understanding how to program the Dragon video system. Programming requirements are covered in section 4.1.2.

I/D bus instructions are specified by a one byte code; rasterop and PBT commands are further specified by a subinstruction byte. Most I/D bus instructions are also accompanied by a 16 bit data transfer.

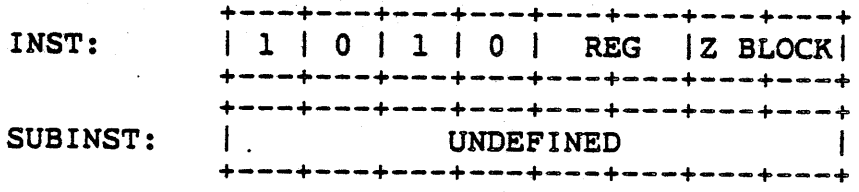
4.2.2.1 Local Processor I/D Instructions

VIPER CHIP REGISTER LOAD



Loads the addressed Viper chip register with the 16 bit data transmitted on the I/D bus and that was contained in the word loaded to the IDD (or IDS) register of the Adder chip previous to loading an I/D bus command. This instruction is always initiated by the local processor; the Adder chip never transmits it without an explicit command to do so.

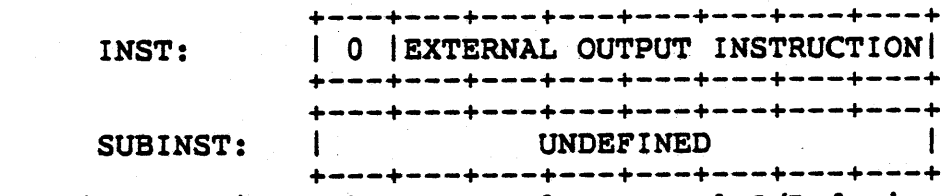
VIPER CHIP Z AXIS REGISTER LOAD



Loads all bits of the addressed register, in each of the 16 planes or sub-planes of the addressed Z block (in the Viper chips that are also chip selected), with the contents of the bit selected by the low four bits of the plane address (programmed into each of the Viper chips). If a Viper chip is set for a low resolution mode, a pair of bits or a nibble is duplicated into each of the 8 pairs or four nibbles of the selected register. Z axis register addresses are defined as:

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

EXTERNAL REGISTER LOAD

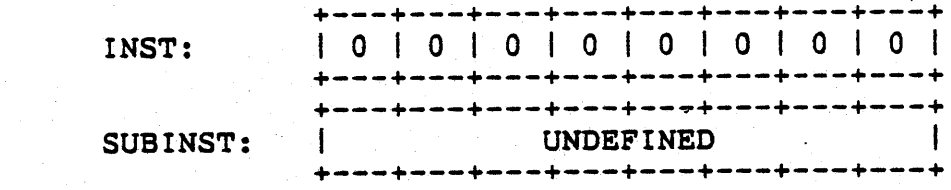


The instructions that control external I/D devices are implementation specific. Information on using external I/D devices is provided in section 5.4.1 of this specification. External I/D instructions are distinguished by bit <7>=0 in the instruction byte. Instruction 00H is reserved for NOP. Any instruction in the range 01H to 7FH may be used for an external device.

4.2.2.2 Adder Chip I/D Instructions

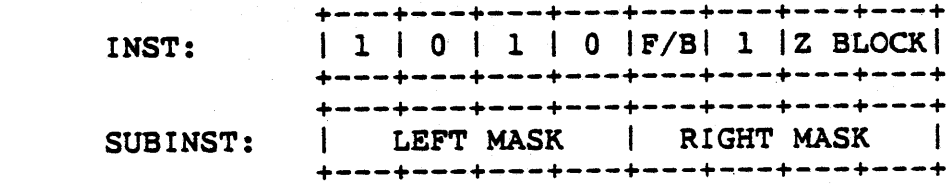
The only direct I/D bus instructions that a programmer will use are the Viper chip and external device register loads. The remaining I/D bus instructions are used by the Adder chip to control rasterops and are listed for the sake of completeness.

NO OPERATION



This instruction is transmitted by the Adder chip when the I/D bus is idling.

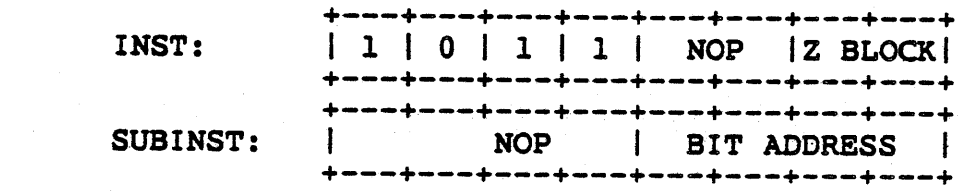
Z AXIS WRITE (TWO INTERLEAVED INSTRUCTIONS)



This instruction is essentially identical to the Viper 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. 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 Viper 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

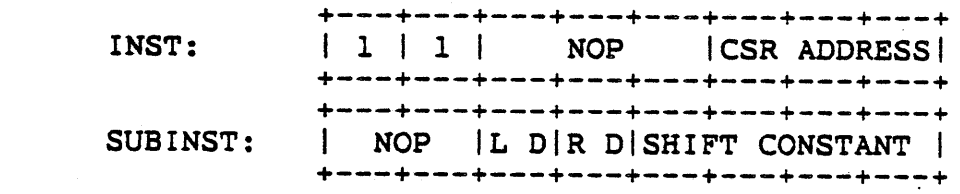
written. However, the Z data being written by the current memory cycle is that which was transmitted during the previous I/D bus cycle, not the concurrent cycle; these are really ordinary Z axis register loads interleaved with destination only rasterop cycles. The right and left mask specifiers are equal for this instruction and are the address of the selected bit; for a complete description of the edge mask see Active Cycles below.

Z AXIS READ



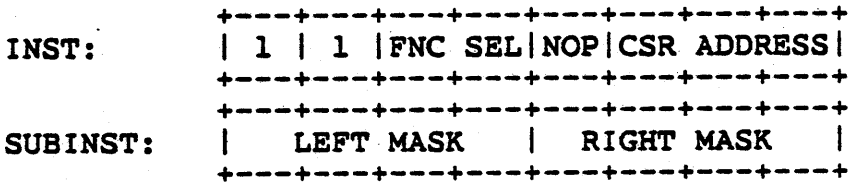
This instruction is used by the Adder chip during a bitmap to processor transfer along with an accompanying memory read cycle. The bit address specifies the bit in the memory word for which the color is being read. (The Viper chip generates a barrel shift constant from the bit address and the low four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

ACTIVE CYCLE FOR MEMORY READ



This instruction is generated by the Adder chip with an accompanying memory read cycle to control the handling of the read data. The CSR address selects the controlling CSR location (and therefore a delay register), and will be 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 left most 16 bits are selected; if the shift constant is 15, the right most 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.

ACTIVE CYCLE FOR READ-MODIFY-WRITE



This instruction is generated by the Adder 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 will be 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 within the word. The left mask is the bit address of the left most pixel to be modified. The right mask is the bit address of the right most pixel to be modified. Therefore, if one pixel is to be modified, both mask values are the same. 0 is the LSB or left most pixel; 15 is the MSB or right most pixel.

5 PHYSICAL CONFIGURATION

For complete specifications of the Adder and Viper chips refer to their respective specifications, referenced in section 1.2. This section describes system wide interconnection, timing and functional issues in general terms. Specific implementation examples are also referenced in section 1.2. The implementation suggestions in this section represent the recommended or intended system configuration but are flexible, of course.

5.1 Adder Chip Pins

The Adder chip is contained in an 84 pin, leaded, square, surface mount package. Pin numbers are indicated in []; for grouped pins, the pin numbers are listed in order, starting with the high order pin.

5.1.1 Processor Interface

- DAT<15:0> (Input/Output) [57, 58, 59, 60, 63, 64, 65, 66, 69, 70, 71, 72, 73, 74, 75, 76]
  Local Processor Data Bus. 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]
  Processor Address. Selects an Adder chip register. Latched on the falling edge of -AS to facilitate connecting the AD and DAT pins on a multiplexed bus.

- -AS (Input) [84]
  Address Strobe. Falling edge latches address on AD<5:0> from local processor bus. -AS is provides the Adder chip select function and should be asserted only for Adder chip bus cycles.

- -DS (Input) [83]
  Data Strobe. During processor write, falling edge latches processor data. During read, the low level enables DAT<15:0> and the rising edge disables DAT<15:0> output buffers.

- RD (Input) [1]
  Processor Bus Read/-Write. Indicates read or write cycle. Latched by -DS.

- -REQ (Output) [11]
  DMA Data Request. Enabled by Adder chip request enable register for DMA controller service request or local

processor interrupt.

- -INT (Output) [10]
  Processor Interrupt. Enabled by Adder chip interrupt enable register for local processor interrupt.

- -INIT (Input) [2]
  Bus Initialize. Resets Adder chip command processor, halts commands, and clears address counter and IDD and address output FIFOs. Disables I/D bus drivers on the Adder chip until the first occurrence of SYNC following deassertion of -INIT.

5.1.2 Memory Address And VIPER 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 bitmap. Addresses are clocked out by ADDCLK; one row and eight column addresses for a major cycle, and one each of row and column for a minor cycle. The sequence of X and Y addresses presented at the row and column times is described in section 5.6.1.

- 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 of the scroll or clipping regions. -WE<0> controls the low order nibble (left most on the screen). A set of values is provided for each column address output by the Adder chip. Only -WE<1:0> are used for 8 bit bus width mode, and only -WE<0> is used for 4 bit bus width mode. 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 right most word in the region. A state output is provided for each column address output by the Adder chip during a screen refresh major cycle.

- ADS (Input) [28]
  Address Disable. When high, allows the Adder chip to complete its current memory cycles and then suspend all update (minor) memory cycle activity. The address bus is NOT tri-stated, minor cycles become dummy 16 bit reads, and major cycles continue unaltered. An external device can take over any of the cycles that it wishes by sensing the 128/-16 and R/-W pins, and supplying its own address, data and control. Refresh and scroll cycles may be lost (if ignored by the external device), but removal of addresses from the address output FIFO is suspended so that updates can continue when ADS is lowered. Scrolling is not recommended when this pin is asserted because addresses held in the address output FIFO may have stale indexes or offsets by the time they are used to access memory.

NOTE

This function is intended to allow a laser printer to suspend Adder chip memory operations so that external hardware can dump the bitmap memory at an appropriate speed for the printer. It may also be used for an external processor or an alternate screen refresh/scroll function to access the bitmap memory directly.

5.1.3 I/D Bus

- ID<7:0> (Input/Output) [22, 21, 20, 19, 18, 16, 15, 14]
  ID Bus Bit. Eight bit bus always transmits during the two instruction cycles; tri-stated during the two data cycles unless a PTB or I/D command is being executed.

- IDCTL (Output) [12]
  Chip Select Control. Selects update or scroll chip select registers. Scroll chip select registers are indicated by a high during execution of commands in the

ICS register.

5.1.4 Monitor Timing

- BLANK (Output) [7]
  Video Composite Blank Signal. High indicates blank.

- CMPSYN (Output) [6]
  Video Composite/Horizontal Sync. Composite sync (or horizontal sync if separate syncs are used). Programmable polarity.

- VRTSYN (Output) [5]
  Video Vertical Sync. Vertical sync; used for separate syncs. Programmable polarity.

5.1.5 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 sync should be initiated by the external timing generator (normally 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 sync interval to reset phase of the Adder chip timing logic. Also, a number of double buffered registers will be updated. Note that this is not a chip reset pin. All clock signals must be held frozen during the entire sync interval (normally two major cycles).

5.1.6 Power

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

Power. Plus 5 volts 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]

5.2 Viper Chip Pins

The Viper chip is contained in a 68 pin, leaded, square, surface mount package. Pin numbers are indicated in []; for grouped pins, the pin numbers are listed in order, starting with the high order pin.

5.2.1 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 right most 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 will be tri-stated, and

the VIPER chip may receive bitmap memory data. When low, all DIO<15:0> and PE output drivers will be enabled.

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

- LTCLK (Input) [66]
Latch Clock. When R/-W and 128/-16 are high, the falling edge of LTCLK is used to latch in bitmap memory data on the DIO<15:0> lines.

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

5.2.2 I/D Bus

- ID<7:0> (Input/Output) [31, 32, 33, 35, 36, 38, 39, 46]
Instruction/Data input/output bus. 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 will be latched and executed. Otherwise, the arriving instruction will be ignored.

5.2.3 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.

5.2.4 Clocks

- ALPHA (Input) [55]
Data on the VID<3:0> lines will be 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 VIPER chip. Period is 1/4 of a major cycle.

- SYNC (Input) [63]
  System Sync. When asserted, synchronizes the phase of internal Viper chip states to the remainder of the Dragon system. Also, a number of double buffered registers will be updated. Note that this is not a chip reset pin.

5.2.5 Power

- VDD 1, VDD 2 [53, 54]
  Plus 5 Volts 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.

5.3 High Speed Timing

The Adder and Viper chip clocks are discussed in section 5.3.1, system synchronization in section 5.3.2, address generation in section 5.6, memory data timing in section 5.7, memory write enable logic in section 5.7.1, and video data and control in section 5.8.

5.3.1 Clock Generation

The Adder and Viper chips require the following clocks, which are described briefly in sections 5.1.5, 5.2.4, 5.1.2, and 5.2.1, and completely in the Adder and Viper chip specifications referenced in section 1.2:

- PHI1, PHI2 - Basic system clocks for both the Adder and Viper. Determines I/D bus timing.
- PHI3, PHI4 - Serial arithmetic clocks for the Adder chip.
- ALPHA - Video output clock for the Viper chip.

- ADDCLK - Address output clock for the Adder chip.
- LTCLK - Data clock for 128 bit cycles (major cycles) for the Viper chip.

For convenience throughout the specifications, clock timing is generally discussed in increments of 30 nsec; however, the actual minimum period for Phi 3 and 4 is 114 nsec, rather than the nominal 120 nsec. The maximum period is at least twice the minimum. All other clocks scale accordingly.

The skew between any two of the above clocks cannot exceed +/- 5 nsec. This can generally be achieved only by deskewing all of these clocks through a common latch, clocked at 30 nsec, because differences in wire delays and loading will consume a significant part of the allowed skew.

The waveforms of all clocks except ADDCLK and LTCLK are fixed and can be found in the Adder and Viper specifications. The waveforms for ADDCLK and LTCLK are dependent on the selection of memory for the bitmap. Considerable range is permitted in these signals; the earliest and latest possible timing is shown in the Adder and Viper specifications, and any placement of transitions between the earliest and latest is permissible. ADDCLK must have nine cycles per major cycle or two per minor cycle. LTCLK must have eight cycles for every major cycle or pair of minor cycles; even though LTCLK does not clock data at the pins during minor cycles, it is still used internally. used for

5.3.2 System Sync

Once per frame (vertical scan of the screen), all system timers are synchronized to each other. Ideally, this should only be required once at system startup, but because loss of synchronization could corrupt all future screen display and writing, periodic synchronization makes the system much more robust to static discharge and other environmental effects. 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 Adder chip), called the sync interval, and asserting the SYNC pin on the Adder and all Vipers during the first part of the sync interval (usually for the first major cycle). 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 as described in section 4.1.1.5. During the sync interval, the clocks must remain in the following states:

- o PHI1 - Low
- o PHI2 - High
- o PHI3 - High
- o PHI4 - Low
- o ALPHA - High
- o ADDCLK - High
- o LTCLK - High
- o Other memory timing signals - as required.

After the sync interval, all clocks must proceed exactly as if the two major cycles of the sync interval had never occurred. Basically, the only clocks that don't freeze are the ones that are needed to time the sync interval.

The sync interval is initiated when the Adder chip issues the SYN/REQ signal. The exact timing of this signal is programmed in the X SCAN COUNT registers. The clock generating logic should respond to SYN/REQ by starting a sync interval at the next major cycle boundary.

If multiple Dragon display systems are to be synchronized to each other, there should be one clock generator for all of the systems. The clock generator responds to SYN/REQ from one of the systems, and then stops the clocks to all systems and issues SYNC to all systems. This will synchronize the memory and display activities of all the systems, assuming that their screen formats are programmed identically.

Processor bus activity can proceed as normal during the sync interval.

5.4 I/D Bus

The Instruction/Data bus is intended to be tied directly to the Adder and Viper chips and to any external devices that might be connected to it. Because the bus is bi-directional and there are no direction controls, it is not possible to put buffers on this bus to increase its drive capability.

The maximum capacitive load allowed on the I/D bus is 400 pf for operation at full clock rate. This should be sufficient for 24 to 32 Viper chips per Adder plus a limited number of TTL devices, if the wire routing is compact. If TTL devices (or other DC loads) are connected to the I/D bus, it may be necessary to guarantee a minimum capacitance on the I/D bus; this could be provided by discrete capacitors, if needed.

The I/D bus transmits two bytes of command from the Adder chip and transfers two bytes of data from the Adder or Viper 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 and PHI1A. The command/data sequence is:

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.

5.4.1 External I/D Bus Devices

In addition to the Adder and Viper chips, external devices can be attached to the I/D bus. These devices must be write only because the Adder chip will always transmit data during an I/D bus command. External I/D commands are distinguished from Viper 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. See section 4.2.2.1 for a further discussion of external I/D instructions.

In order for an external device to respond to the I/D bus, it must decode at least the instruction byte which is present only during PHI2A cycles. An external device can synchronize to the PHI1/PHI2 A/B cycles by using SYNC, which always occurs during a PHI2B cycle (see section 5.3.2).

5.5 Chip Selects

Each Viper 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. The usual implementation of chip selects is to provide an unencoded pin from a chip select

register for every Viper chip; this allows any combination of Viper chips to be enabled. Chip selects are sensed by the Viper chip only during the instruction cycle (PHI2A).

Because the scroll process needs to access the Viper chips without disturbing the update process, it requires a separate set of chip selects. The Adder chip provides the IDCTL pin to select the scroll chip selects when a scroll I/D instruction is in progress. The scroll chip selects are normally provided by a register; but, they may simply all be enabled during scroll I/D cycles, if all planes will always be scrolled or erased together, and Z axis register loads are used to load the scroll fill register. A Z register load cannot be used for the scroll fill register if the update process might change the plane addresses in the Viper chips.

The chip select register can be connected either to the local processor bus directly, or as an external device to the I/D bus. The latter connection is required if a DMA controller feeds data to the Adder chip, because chip select changes are generally mixed with other Adder chip commands.

5.6 Memory Address Bus

The bitmap address bus is common to all bitmap planes. Buffering may be required to fanout the bus in systems with many planes. There are two groups of pins from the Adder chip that 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 only meaningful during write cycles; the SCROLL pin is only meaningful for 128 bit read cycles.

   ADDCLK is required to have a high interval that includes the first 30 nsec of the PHI2B (see section 5.4) 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 Adder chip. The previous pulse fetches the row address; and the following pulses (for a major cycle) fetch additional column addresses. There must be nine cycles of ADDCLK in a major cycle and two cycles of ADDCLK in a minor cycle. ADDCLK is normally the same for reads as for writes.

The series of column addresses in a major cycle form a sequential up count. If the memory controller needs to generate its own column addresses (for example, implicit addressing is used in nibble mode RAMs), it may use the first column address and assume the rest are sequential.

The MEMADs need to be latched externally to the Adder chip to deskew the propagation delay of the Adder chip 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 Adder chip; similarly, the row address is applied to the memory while the first column address is fetched from the chip. There is enough setup time available to the latch so that multiple banks of memory can be fed from separate latches, while a buffer chip fans out from the Adder to the latches.

The SCROLL pin indicates memory words that are contained in the scroll region. It comes out of the Adder chip with the address of the associated word and should be latched along with the MEMADs; but, it is accepted by the Viper chip as if it were data coming from the memory, so it must be delayed further so that it matches the data timing.

NOTE

The scroll pin from the Adder chip does NOT connect directly to the scroll pin of the Viper chip.

The WE pins are discussed in section 5.7.1.

2. 128/-16, R/-W and FORCE are provided by the Adder 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 is next, which will contain both read and write portions.

NOTE

The 128/-16, R/-W pins from the Adder chip do NOT connect directly to the 128/-16, R/-W pins of the Viper chip.

The force pin indicates major cycles in a frame which is down scrolling. It is further discussed in

section 5.7.1.

5.6.1 Memory Address Bit Assignments

The sequence of address outputs at the address pins is a function of the bus width and memory configuration bits in the X Scan Configuration register of the Adder 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 (for memories requiring either 128 or 256 refresh addresses) by keeping the fastest moving X and Y addresses in the low 8 row addresses. The Address bits are numbered according to their significance in address calculations. The lowest 2, 3 or 4 X address bits (not shown in the tables below) access pixels within a memory word in 4, 8 and 16 bit bus width modes, respectively.

"Greater than 512" mode:

Address Bit	10	9	8	7	6	5	4	3	2	1	0
16 Bit Bus Width											
Row time	X12	Y11	X10	Y5	Y4	Y3	Y2	Y1	Y0	X8	X7
Column Time	Y12	Y10	X11	X9	Y9	Y8	Y7	Y6	X6	X5	X4
8 Bit Bus Width											
Row time	Y12	Y10	X10	Y4	Y3	X8	Y2	Y1	Y0	X7	X6
Column Time	Y11	Y9	X11	X9	Y5	Y8	Y7	Y6	X5	X4	X3
4 Bit Bus Width											
Row time	Y11	Y9	X10	Y3	Y2	X7	X8	Y1	Y0	X6	X5
Column Time	Y10	Y8	Y12	X9	Y5	Y4	Y7	Y6	X4	X3	X2

"Greater than 1024" mode:

Address Bit	10	9	8	7	6	5	4	3	2	1	0
16 Bit Bus Width											
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
8 Bit Bus Width											
Row time	Y12	Y10	X10	X9	Y3	X8	Y2	Y1	Y0	X7	X6
Column Time	Y11	Y9	X11	Y4	Y5	Y8	Y7	Y6	X5	X4	X3
4 Bit Bus Width											
Row time	Y11	Y9	X10	X9	Y2	X7	X8	Y1	Y0	X6	X5
Column Time	Y10	Y8	Y12	Y3	Y5	Y4	Y7	Y6	X4	X3	X2

These assignments are suitable for standard Dragon systems. They result in memory spaces that are powers of two in both height and width. If necessary, the address bits can be demultiplexed and recombined to form a new arrangement of row and column addresses; there is even sufficient time in most memory arrangements to get the column address bits out of the Adder and include some of them in the row address going to the memory. Such remultiplexing can allow the Dragon system to be used when the requirements of other processors have already determined the necessary row/column combinations. Remultiplexing can also be used to alter the power of two restriction on the shape of the memory; for example, a 1024 x 1024 memory could be altered to 960 x 1088.

5.6.2 Memory Refresh

For the standard memory address sequence given in section 5.6.1, memory refresh is accomplished by the screen refresh process if MEMAD<7:0> are connected to the memories (it is assumed that memories at least 64K deep are used). The following table gives the number of unique row addresses generated per horizontal scan and the number of scans required to get a complete sequence of 128 or 256 row addresses, for each combination of the memory configuration bit and the bus width.

"Greater than 512" mode:

Bus Width	16 Bit	8 Bit	4 Bit
Addresses per scan	4	8	16
Scans for 128 rows	32	16	8
Scans for 256 rows	64	32	16

"Greater than 1024" mode:

Bus Width	16 Bit	8 Bit	4 Bit
Addresses per scan	8	16	32
Scans for 128 rows	16	8	4
Scans for 256 rows	32	16	8

For typical scan times, a complete 128 address sequence is completed in 0.5 to 0.6 msec; this is far less than the 2 msec normally required. However, no refresh is accomplished during vertical retrace and the last sequence of refresh addresses on the screen may not be complete (according to the number of displayed scans). Therefore, the worst case time to complete a refresh is the refresh period computed from the above table, plus

the vertical retrace time, and plus the duration of any incomplete refresh cycle that occurs at the bottom of the displayed screen.

Only the 128 bit read cycles perform the required screen and memory refresh. Therefore, if more than one bank of memory is used, it is only necessary to RAS all banks of memories for 128 bit, read cycles; only the selected bank need be RASed for all other cycles. Restricting the RAS cycles on memories can save considerable power in large memory systems.

5.6.3 Memory Selection And Configuration

Although the Dragon system was designed to work with 64Kx1 and 64Kx4 static column RAMs (RAMs having a very fast page mode cycle), Dragon can be used with a variety of other RAMs. Unfortunately, static column RAMs have not become as available as planned. The data rate during major cycles requires approximately 90 nsec page mode cycles. Speed requirements for minor cycles do not stress even ordinary RAMs.

A direct alternative to static column RAMs are nibble mode RAMs. These allow fast cycling of four column bits at a time. This requires two RAS cycles for each major cycle, but this should be possible. Unfortunately, the most readily available nibble mode RAMs are 256Kx1, which dictates that four screens of memory be used per plane; of course, this is ideal if large amounts of off screen memory are required, anyway. Other configurations of nibble mode RAMs should be available.

If two screens of memory per plane are required or acceptable, ordinary speed 64Kx1 or 64Kx4 RAMs can be arranged in even and odd banks with successive major cycle words accessing alternate banks. The same arrangement can be used for one screen per plane if 16Kx4 RAMs are used.

Video RAMs (having a built in shift register output for a row of data) are not useful for Dragon because the normal screen refresh reads are still required to pick up data that is being scrolled and to perform memory refresh.

When a memory configuration is used that meets the major cycle requirements for the visible screen but still more off screen memory is required, ordinary memory can be used for the additional memory. In this case only the RAS part of a major cycle read is applied to the additional RAM (for memory refresh) and the fast CAS cycles will never be applied.

5.7 Memory Data Bus

The memory data lines for each plane are connected to the Viper data lines, usually with a latch appropriate to the particular memory timing required. The latch is generally 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 to the requirements of the Viper; a transparent latch is normally used, which can be clocked by LTCLK. When 8 or 4 bit bus width mode is used, a proportionately reduced number of memories are attached only to the low order bits of the Viper data bus. The maximum capacitive load on the data bus that the Viper chip can drive is 40 pf, for operation at full clock rate; this is the primary limitation on the number of memory chips that can be directly connected.

The type of memory cycle in the Viper is controlled by the 128/-16 and R/-W pins. These pins are normally changed at the time of the PHI2B clock (see section 5.4) that starts each major or minor cycle. If the minor cycle is a read-modify-write (signaled from the Adder 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 Adder chip do NOT connect directly to the 128/-16, R/-W pins of the Viper chip.

The memory data bus is controlled differently for major and minor cycles.

1. Major cycles - The clocking of data is controlled by LTCLK. During 128 bit reads, data is latched in the Viper on the falling edge of LTCLK. During 128 bit writes, data propagates from the Viper on the rising edge of LTCLK. The timing of LTCLK is normally different for 128 reads and writes. During 16 bit cycles, LTCLK must continue to run; the read timing is normally used.

   The scroll signal is applied to the Viper along with each corresponding memory word during 128 bit reads. The scroll signal is further described in section 5.6.

2. Minor cycles - During a 16 bit read, data is latched in the Viper chip on the falling edge of the PHI2A in the middle of the cycle. During a read-modify-write, data is latched by the earlier of R/-W falling or PHI2A falling; the falling edge of R/-W will cause the Viper

to reverse its direction and enable write data.

5.7.1 Write Enable Circuits

The write enables of the bitmap memories are used by the clipping and scrolling logic to selectively write memory words and parts of words. In a standard Dragon system, 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 Adder chip provide the write enables for each nibble of each 16 bit word associated with each column address. The -WE signals should be latched externally on ADDCLK, just like the MEMAD signals. The write enables may be further delayed to match required memory timing. These signals provide the four bit resolution in the clipping and scrolling boundaries and are used by the write enable logic for each plane. When 8 or four bit bus width mode is used, only the least significant two or one -WE pin is used.

2. The PE pin from each Viper 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 that is present during each I/D instruction of the rasterop; that is, 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 Viper chip SCROLL CONSTANT register.

3. The FORCE pin from the Adder chip must be delayed until the rising edge of PHI2B that starts a major (or minor) cycle and 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 should be disabled in any plane where PE is not asserted, UNLESS both FORCE is asserted AND PE is not asserted.

4. Any high resolution edges in the final write enable waveform must be supplied by the high speed timing logic. At a minimum, an edge must be supplied for the read-modify-write cycle; 128 bit writes may also require

write edges, depending on the requirements of the memories. No write strobes should be generated during read cycles.

This logic can be simplified if system requirements allow. If all planes will always scroll together (no plane ever disabled for scrolling), the FORCE pin can be ignored. If all planes will always act together for both scrolling AND rasterops (no scroll disable and always chip selected during rasterops), the FORCE and PE pins can be ignored.

5.7.1.1 Single Pixel Clipping And Scrolling Boundaries

It is possible to increase the resolution of clipping and scrolling boundaries to single pixels with the addition of considerable logic external to the Adder and Viper chips. This capability is limited to systems that operate in 16 and 8 bit bus width modes, and that use single bit wide memories so that there is one write enable for every bit. The write enable logic described above would be increased to one set of gates for every bit (not just each group of four bits). In place of the write enable signals from the Adder, the write enable logic would be enabled by high resolution write enables generated as follows (the description assumes full Dragon functionality for a 16 bit system, some logic optimization should be possible):

1. Four 16 bit registers store write enable masks for both the left and right edges of both the clipping and scrolling region. These registers are programmed the same way as the Left and Right Scroll Boundary registers in the Viper chip; except that all six boundary registers can be programmed to any pixel.

2. 16 eight-to-one multiplexers select the correct write enable mask to supply to the 16 sets of write enable logic for each plane. The eight choices are: the four boundaries (left and right, clip and scroll), and two inputs each of a constant "write all" mask and a constant "write none" mask.

3. The three control signals for the multiplexer are the 128/-16 signal (as timed for the Viper chip input) to select the clipping or scrolling boundaries, and the -WE<1:0> signals from the Adder to determine if the word being written is:

    1. completely in the region (write all),
    2. completely out of the region (write none),
    3. contains the left edge of the region (left boundary),

4. contains the right edge of the region (right boundary), the right edge of the region is the first pixel outside the region on the right.

4. The programming of the left and right edges of the clipping and scrolling boundaries in the Adder chip must be "fudged" to cause -WE<1:0> to have the above meaning of: in, out, left and right. The registers are programmed to the exact address of the left and right boundaries (as the right boundary is defined for these registers) except that the low bits (4 or 3 according to the bus width) are replaced with the address of pixel <4> within the word. If the two edges fall in the same word (high bits the same), the programming of the right boundary must be increased by one bus width and the external boundary registers programmed as are the Viper boundary registers when both edges fall in the same word. The above programming will insure that -WE<0> is asserted if the word being written is in the region or contains the right edge, and -WE<1> is asserted if the word is in the region or contains the left edge. If the word is outside the region, neither pin will be asserted.

The clipping status bits will not be accurate if this technique for increasing the boundary resolution is used because the left and right edges of the clipping boundaries in the Adder chip are not programmed to the actual boundaries. Four bit bus width mode cannot use this technique because only one -WE pin is defined.

5.8 Video Bus And Video Output Circuits

The video bus from the Viper chip is clocked by ALPHA and provides data on every clock, every second clock or every fourth clock, depending on the selected bus width. The data should be deskewed by a short setup time latch or shift register, presumably on the next 60 nsec edge. Ultimately, the four bit video data is shifted in a serial stream through any required color map, digital to analog converter or other processing, and sent to a video monitor. The maximum capacitive load on the video bus is 40 pf for operation at full clock rate.

The BLANK and CMPSYN signals from the Adder chip are precision outputs that provide the composite video blanking and composite sync of the monitor (if programmed). These signals are generated with 60 nsec resolution and are normally deskewed with the video data. Because the BLANK output is only programmable in 60 nsec increments, care must be exercised to insure that the pipelining of the blank signal matches the higher resolution

video signal so that the correct pixels are blanked and unblanked. A latch with a synchronous clear or an equivalent logic structure must be used to correctly mix the blank with the video. Be sure that the pipelining used for blank and sync will result in legal programming of the X SCAN COUNT registers in the Adder chip (see section 4.1.1.5); additional delay may be needed to move the blank and sync events in the Adder to legal times, while maintaining the required relationships in the output timing.

5.9 Processor Bus

The local processor bus is designed to interface to any generic 16 bit processor, or to word operations on a 32 bit processor. The MC68000 and the uVAX were used as models. The INIT pin resets the operational aspects of the chip; it: 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. Details of the processor interface timing are covered in the Adder chip specification.

If an eight bit processor is required, the Adder chip only requires a 16 bit data path for writing registers. The only readable register is the STATUS register and this can be an 8 bit read if it is permissible to ignore the vertical blank and clipping status bits.

A DRAGON CHIP BUGS AND CHANGES

The second pass of both the Adder and Viper chips are now available for use in product design. No product or prototype should try to use the first pass of either of these chips.

A.1 Pass 2 Chips

A.1.1 Adder Chip - Pass 2

The pass 2 Adder is designated DC323B.

A.1.1.1 Bugs In The Pass 2 Adder

1. If the slow vector of a rasterop has a length greater than one and the rasterop is canceled with the cancel command, a following rasterop may not be executed due to a rasterop initialization problem. The temporary fix is to issue a dummy rasterop (pen up set in the mode register) after any cancellation of a rasterop with a slow vector length greater than one.

A.1.1.2 Features Not In The Pass 2 Adder

1. None.

A.1.2 Viper Chip - Pass 2

The pass 2 Viper is designated DC322B.

A.1.2.1 Bugs In The Pass 2 Viper

1. Z read commands on the I/D bus do not work correctly if executed during a scroll memory cycle (128 bit write cycle); the scrolling data is corrupted. This bug only affects operation in 8 and 4 bit bus width modes; 16 bit mode works. This bug will be fixed in the pass 3 Viper.

A.1.2.2 Features Not In The Pass 2 Viper

1. None.

A.2 Pass 1 Chips

A.2.1 Adder Chip - Pass 1

The pass 1 Adder is designated DC323. No design should use this part.

A.2.1.1 Bugs In The Pass 1 Adder

1. Scrolling does not work if the right scroll boundary is set to the right edge of the last major cycle read of the scan (the right edge of the screen in 16 bit bus mode with a displayed screen width of 1024 pixels). Full screen can still be erased by the scroll mechanism if the chip is initialized properly.
2. Indexes are not pipelined properly during scrolling. Causes some data to be written to the wrong place during scrolling.
3. Command synchronizer has been improved, but the old one should not cause any problems in normal use.
4. A circuit problem can cause a command through the ICS register to clear data that is being transferred by simultaneous BTP operation. This is not a problem except for scrolling during BTPs.
5. Y fill operations may cause data to be written in the left most pixel of each scan line.
6. Source 2 double read flag is not computed properly. Causes tiles with a width not equal to the bus width to be written improperly. Tiles the width of the bus width mode (4, 8, 16) are OK.
7. Double read flags for source 1 and source 2 are not pipelined properly. Causes various symptoms during scrolling, and with "italic" destinations.
8. The vertical sync pin is connected to the wrong signal internally.

9. Scaling basically works, but there are various anomalies in the operation.

10. Linear patterns do not scroll properly, and the scale engines are not initialized as described in the specification. There are no problems for ordinary operations, though.

11. Z mode BTPs don't work (they calculate the wrong pixel address within each memory word).

12. The wrong number of words may be transferred during PTBs, sometimes.

13. PTB+BTP while Scrolling - Numerous bugs, do not bother to try this.

14. Several bugs with fill mode while scrolling. Causes lines to extend from filled area.

15. Words may not be written with some source/destination operations while scrolling.

16. Filled areas with crossing vectors do not write all pixels at the cross.

17. Some rasterops do not init properly while scrolling. Rarely causes trouble.

18. Error in IDD FIFO controller causes words to be left in IDD FIFO during PTBs; more words taken out than put in. Only a problem when fed by a DMA controller.

A.2.1.2 Features Not In The Pass 1 Adder

1. Double Buffer Pause Register - The operation of the pause register is changed. Loading the register only causes a new pause value, it does not clear the status bit. The status bit is cleared explicitly, just like the scroll service bit. The pause value is double buffered, just like the scroll boundary registers. The status bit can queue two pause events (thus it may have to be cleared twice to make it stay cleared.

2. New Status Bit from Vertical Blank - There is a new status bit set by the leading edge of vertical blank. This bit is in bit position "D". It is cleared just like the scroll service bit.

3. Don't care scale factor when source 1 disabled - The scale factors will be ignored unless both the source 1 and the destination are enabled for a rasterop.

4. Detect Zero Width Rasterops - Zero height or width rasterops will cause no pixels to be drawn. The first pass chip will hang for a zero slow vector, and will draw a line across the screen for a zero fast vector.

5. Pipelining of indexes improved to prevent problems when changing update regions while scrolling.

6. Error Term Initialization - Two new error registers can be loaded with numbers added to the Bresenham error terms during rasterop initialization. Used to adjust the initial error term. Normally programmed to zero. Registers [2F] and [30].

A.2.2 Viper Chip - Pass 1

The pass 1 Viper is designated DC322. No design should use this part.

A.2.2.1 Bugs In The Pass 1 Viper

1. No plane address can be assigned higher than the bus width that the Viper chip is set to.

A.2.2.2 Features Not In The Pass 1 Viper

1. There are no foreground/background registers; the output of the logic unit goes directly to the mask mux. The source register is used for PTBZ operations.

   To use a second pass part as a first pass part: program the foreground register to all ones and the background register to all zeros. To use a second pass part as a first pass part for PTBZs: set bits <2:3> to "00"; however, the source register will be addressed for PTBZ operations, but the path from this register may not be fast enough to allow correct operation.

2. There is only one delay register; it is available to use with any CSR. Because all source operations require a

delay register, two source operations are not possible with first pass parts. Thus the delay register can be enabled in CSRs 0, 1, 4 and 5, and disabled in CSRs 2 and 6; this will insure compatibility with the second pass parts.

3. There are no complementer and resolution mode logic following the source register.

   To use a second pass part as a first pass part: bits <7:6> should be set to "11".

4. Although the first pass parts allow three bits of Z Block address, only two bits should be used for compatibility with second pass. Bit <3> selects the source or fill register.

   To use a second pass part as a first pass part: set bit <2> to "0".

5. There is no vertical scroll direction bit. To use a first pass part with the Adder chip, this input must be inverted during down scrolling. The scroll pin and write back enables must be forced for all planes not participating in down scrolling.

   To use a second pass part as a first pass part: bit <6> is set to a one, permanently.

6. Because there is no vertical scroll direction bit, the boundaries must be modified for down scrolling. If down scrolling:

   All bits of the LEFT boundary are set to the complement of the values specified for the RIGHT scroll boundary when NOT down scrolling.

   All bits of the RIGHT boundary are set to the complement of the values specified for the LEFT scroll boundary when NOT down scrolling.

   To use a second pass part as a first pass part: set the scroll direction bit for up scrolling and program the boundaries as described above for first pass.

This page intentionally left blank.

INDEX

-S-