

ESV Series Workstations

ESV Self Maintenance

Evans & Sutherland Proprietary

EVANS & SUTHERLAND COMPUTER CORPORATION
DESIGN SYSTEMS DIVISION
Salt Lake City, Utah

Evans & Sutherland Proprietary

The distribution of this document is limited to Evans & Sutherland employees and authorized individuals outside Evans & Sutherland. Distribution outside Evans & Sutherland requires a signed non-disclosure agreement with Evans & Sutherland and approval by an officer of Evans & Sutherland.

ESV is a registered trademark of Evans & Sutherland Computer Corporation.

E&S, ESV Workstation, ESV Series, ESV Series Workstations, ES/os, ES/Dnet, ES/PEX, ES/PHIGS, ES/PSX, Clean-Line, Fiber Link, Local Server, CDRS, and Shadowfax are trademarks of Evans & Sutherland Computer Corporation.

X Window System is a trademark of the Massachusetts Institute of Technology.

UNIX is a registered trademark of AT&T.

RISC/os and RISCompiler are trademarks of MIPS Computer Systems, Inc.

Motif is a trademark of the Open Software Foundation, Inc.

LAT Host Services, DEPICT, and PCONFIG are trademarks of Ki Research.

AVS is a trademark of Stardent Computer, Inc.

Looking Glass is a registered trademark of Visix Software Inc.

VAX, VMS, and DECnet are trademarks of Digital Equipment Corporation.

Ethernet is a registered trademark of Xerox Corporation.

SunPHIGS is a registered trademark of Sun Microsystems, Inc.

Spaceball is a trademark of Spatial Systems Pty Limited.

Kodak is a trademark of Eastman Kodak Company.

Part Number: 517910-900 Final Review

March, 1992

Protected under the Utah Trade Secret Act.

Table of Contents

Chapter 1 Product Overview	1
Hardware Summary	2
Graphics Subsystem	4
Interactive Devices	5
Monitor	5
Disk and Tapes	5
Communications	5
Packaging	6
Diagnostics	6
Software Summary	6
System Software	6
Graphics Interface	6
Window Interface	6
ESV Workstation Graphics System Specifications	6
Graphics Features	6
Shading and Lighting Features	7
Performance	7
Architecture	7
Graphics	7
Chapter 2 Hardware Overview	1
Introduction	1
Design Goals and System Features	1
3D Vector Capabilities	1
Polygon Capabilities	1
CPU Features	2
Software Interfaces	2
System Expansion	2
Hardware Architecture	2
Workstation Architecture	2
CPU Architecture	4
Geometric Processor Architecture	4
Frame Buffer Architecture	6
Video Pipeline Architecture	6
Hardware Assemblies	8
General	8
Backplane	8
CPU Card	8
GSE Card	8
DSP Card	8
Frame Buffer Card	10
Video Card	10

Table of Contents

FONI Card	11
Monitor	11
Chapter 3 Getting Started.....	1
Preparation for Booting	1
Debug Terminal	1
General Behavior	3
Test Descriptions	4
Non-visible Confidence Tests	4
Visible Confidence Tests	5
Booting UNIX	7
Logging On	9
Chapter 4 Central Processing Unit (CPU) Card	1
Objectives	1
Introduction	1
CPU Core	1
R3000 RISC Processor	4
R3010 Floating Point Coprocessor	4
Data and Instruction Caches	4
Write Buffer	4
Read Buffer	4
Mbus Interface Controller	4
E&S Graphics Coprocessor	5
Graphics Coprocessor Condition Flags	5
Mbus	6
Error Detection and Correction Logic	6
Memory Modules	6
SCSI Controller	6
Ethernet Controller	8
VMEbus Interface	8
VME/Gbus Gateway	8
SLOWbus Devices	10
Serial Ports and Timers	10
Boot EPROM	10
Time-of-day Clock, Non-volatile RAM	10
Diagnostic LEDs	10
VBIC	11
YAM	11
Memory Maintenance Register	12
Interrupts	12
Introduction	12

Table of Contents

Interrupts in the R3000	12
.Memory Maps	13
Chapter 5 Graphics System Executive (GSE) Card	1
DSP Subsystem	1
GBUS Subsystem	1
GBUS Arbiter	3
SYNC Memory	3
VRINT Control and Counter	4
Laundry Chute Subsystem	5
Output FIFO Arbiter	5
OBUS	7
Output FIFO	7
FBUS Controller	7
PPA Loader Subsystem	7
PPA FIFO	8
PPA PACKET CONTROLLER	8
READ Pixel Controller	10
Laundry Chute Diagnostic Mode	10
Chapter 6 Frame Buffer (FB) Card	1
Overview	1
Subsystem Descriptions	1
The Pixel Processors	1
Frame Buffer Memory	2
Chapter 7 Video Card	1
Overview	1
Subsystem Descriptions	1
Video Timing Control	1
Pixel Output Pipeline	1
Pixel Pipeline Data Flow	6
The Window Lookup Table	6
Frame Buffer Interface	6
GBUS Interface	6
Registers	6
Color Lookup Tables	8
Color Lookup Table 0 (CLUT0)	8
Color Lookup Table 1 (CLUT1)	8
Color Lookup Table 2 (CLUT2)	8
Color Lookup Table 3 (CLUT3)	8

Table of Contents

Window Lookup Tables	8
Window Lookup Table 0 (WLUT0)	9
Window Lookup Table 1 (WLUT1)	9
RAMDACs	9
Cursor Registers	9
Video Timing	9
VTGA internal registers	9
Maintenance	9
Maintenance register	9
Memory Maps	10
I/O Panel Connections	14
Video Connector	14
PC Connector (2x4)	14
Latches	14
Pinout	14
Signal Definition	14
Video Option Connector	14
PC Connector (2x10)	16
Pinout	16
Program Execution	17
Chapter 8 Display Adjustment	1
Display Adjustment	1
Image Data Formats	1
Screen Image Adjustments	2
Channel Selection	3
Selecting a channel for adjustment:	3
H Phase Adjustment	3
H Size Adjustment	3
H Position Adjustment	3
V Position Adjustment	3
V Size Adjustment	4
PCC - AMP Adjustment	4
PCC Phase Adjustment	4
Chapter 9 ESV Workstation Diagnostics	1
Introduction	1
TCL - Test Command Language	1
Introduction	1
Test Initiation	1
Help Availability	1
Diagnostic UUT and Test Phases	2

Table of Contents

Error Handling	2
Error Reporting and Handling	2
Batch Processing	3
Special Debugging	3
Tcl Command Line Arguments	3
Tcl Commands	4
Command Summary	9
Set Commands	9
Show Commands	10
Control Commands	10
Help Commands	10
Debug Commands	10
ESV System Diagnostic	11
Name	11
Release	11
Description	11
Execution	12
esvsysdiag	13
Name	13
Synopsis	13
Description	13
Execution	14
Options	14
Chapter 10 UNIX Networking	1
Network Protocols	1
Low Level	1
Physical Layer	1
Higher level	1
Transport Layer	1
Session Layer	1
Presentation Layer	1
Application Layer	1
Networking Tools	3
ping	3
spray	3
netstat	3
who & whodo	3
arp	3
rpcinfo	4
ifconfig	4
Network Addressing	8
Ethernet Address	8

Table of Contents

Internet Address	8
Examples:	8
Network File System	11
Server/Client	11
Daemons	11
NFS Client	11
Manually Mounting a File System	11
NFS Servers	12
ESDnet Overview	13
Using dap	13
File Transfer and Management Commands	14
General Utility Commands	14
Logging In to a Remote System	14
Login to a Remote System	14
Logging In From a Remote System	15
Using dnamail	15
Chapter 11 Disk Replacement and Restoration	1
Hard Disks and Tape Drive	1
Objectives:	1
Overview	1
Hard Disk Drives	1
Drive ID Selection	1
Trouble Shooting Hard Drives	4
Power Up Sequence	5
Power Down Sequence:	5
Tape Drive	6
Formatting and Partitioning	8
Formatting	9
Partitioning	9
Adding a Hard Disk	12
Backup and Restore	15
Appendix 1.....	Adding Swap Space to Your ESV
Appendix 2.....	Determining Disk Model Number
Appendix 3.....	Booting from Tape
Appendix 4.....	System Disk Upgrade
Appendix 5.....	ESV Default Disk Partition Table
Appendix 6.....	Temporary Restoral of the Fstest Account

1. Product Overview

The ESV Workstation is a UNIX-based, high-performance, 3D graphics workstation that comes with a 19" raster color monitor, a keyboard, and a mouse in the base configuration. The ESV Workstation is engineered to support applications in molecular modeling, industrial/automotive design, and design engineering and analysis. It supports the image quality and performance requirements of these applications. This chapter contains summaries of the hardware, the software, and specifications of the product.

ESV Workstations can function in different physical environments:

- As stand-alone workstations,
- In a network with other workstations,
- As front-end systems for a supercomputer.

The workstations have the following general features:

- Highest graphics performance available in a specified price range,
- Good internal CPU and memory for standalone procedures,
- Optional wide-bandwidth support for attachment to a central computing resource, disk access for diskless nodes, and networked hardware,
- UNIX operating system,
- X window system,
- PHIGS and PHIGS PLUS,
- PEX,
- Motif,
- PSX emulator.

The ESV Workstation is available in different configurations. Each configuration, defined by a model number, represents both a physical configuration and a level of performance outlined below:

<u>Model</u>	<u>Vectors 4-Pixel/sec</u>	<u>Vectors 10-Pixel/sec</u>	<u>Polygons/sec</u>
ESV 5	360K	360K	19K
ESV 10	525K	525K	28K
ESV 20	860K	680K	45K
ESV 30	1060K	680K	62K
ESV 40	1100K	680K	78K
ESV 50	1100K	680K	100K

Figure 1-1 shows the base system configuration and optional peripherals.

Hardware Summary

The ESV Workstation CPU is implemented with a MIPS R3000 RISC microprocessor. The CPU is central to the system, and it ties together the system's two global busses: the VMEbus and the Gbus, a proprietary bus used by the graphics subsystem. The CPU is capable of accessing devices on either the Gbus or the VMEbus.

The color raster monitor has a viewing format of 1280 x 1024 pixels. The custom keyboard maintains communication via a synchronous serial link. The ESV Workstation mouse is a three-button optical mouse.

The standard hardware configuration is listed below:

- 20 MIPS processor (25Mhz),
- Ethernet controller with TCP/IP,
- Two RS-232 ports,
- One keyboard port,
- One mouse port,
- SCSI support for external devices,
- 19" raster monitor,
- Graphics card set,
- Optical mouse,
- Alphanumeric keyboard,
- 380/760/1200 Mbyte hard disk,
- 150 Mbyte tape drive,
- VMEbus.

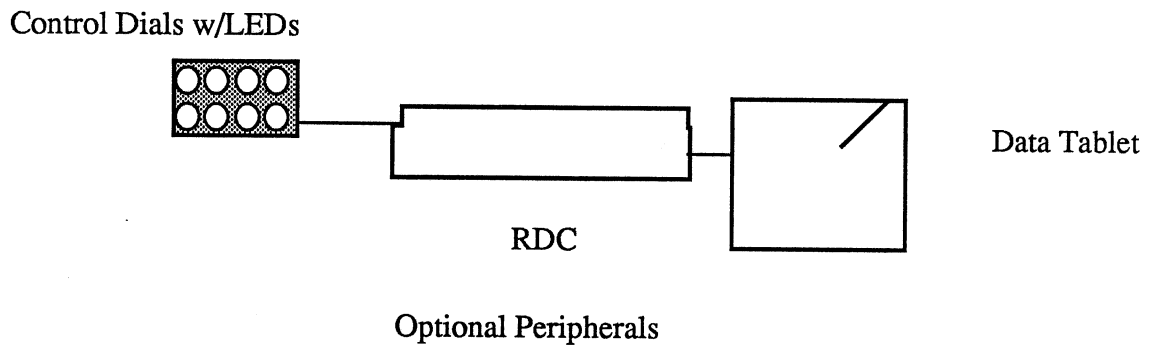
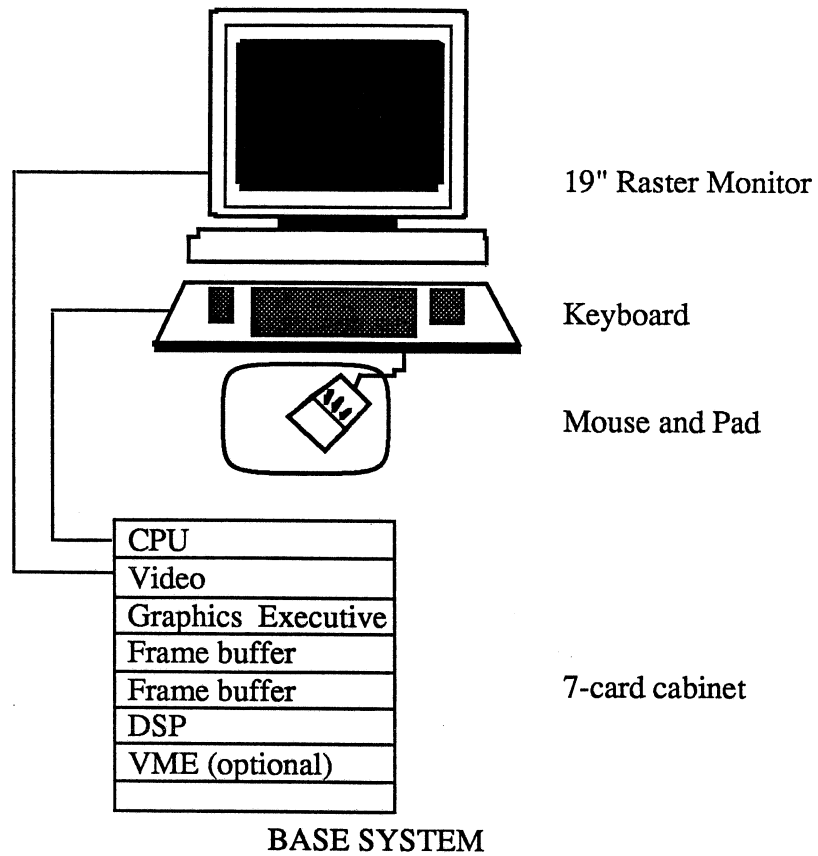


Figure 1-1. Base system configuration and optional peripherals

Graphics Subsystem

The initial release of the base graphics subsystem contains the following cards:

- One CPU card,
- One video card,
- Two frame buffer (FB) cards containing a 24-bit, double-buffered RGB,
- Graphics executive (GSE) card,
- Digital signal processor (DSP) card.

The graphics hardware consists of three main elements: the pixel processor, the DSP, and the frame buffer. The system is configured with 4 pixel processors.

The DSPs are designed to provide the following minimal and expandable configurations to match pixel processor performance:

- 1280 x 1024 pixels available on screen,
- 48 color planes, giving full 24-bit color, double buffered, plus 24-bit z-buffer,
- 8 window/overlay planes,
- Hardware 3-color cursor or 1-color cross hair,
- 4 pixel processors,
- Supports asynchronous window updates,
- Programmable video timing formats, including stereo.

The ESV Workstation optional features include the following:

- Add-on memory, 128 Mbyte maximum,
- Add-on DSP cards,
- 380 Mbyte, 760 Mbyte, and 1.2 Gbyte formatted hard disks,
- Fiber-optic network interface (FONI) card,
- Reprogrammable data concentrator (RDC) for additional peripheral support,
- Peripherals, including control dials with LEDs, and data tablets,
- Stereo viewing option.
- Thermal printer, providing photographic quality hard copy.

The video card is capable of supporting multiple timing formats, including those listed here:

- 1280 x 1024 60 Hz, non-interlaced (default),
- 640 x 1024, 56 Hz, interlaced, and
- 1280 x 1024, 56 Hz, interlaced.

Interactive Devices

Standard interactive devices include those listed below:

- Alphanumeric keyboard,
- Mouse/pad.

Optional peripherals include

- Reprogrammable data concentrator (RDC),
- Control dials with LEDs,
- Data tablets: 6 x 9, 12 x 12, 15 x 15, and 18 x 25.

Interactive devices attach either to the serial ports on the CPU card or to the RDC. The keyboard and mouse can always be passed through the RDC.

Monitor

The standard display is 19-inch color raster monitor. Resolution is 1280 x 1024 for the ESV Workstation. Standard is 60 Hz. Supports the stereo option at 111 Hz.

Disk and Tapes

The ESV Workstation disk options include 380 Mbyte, 760 Mbyte, and 1.2 Gbyte formatted hard disks. Either the 7-card or the 14-card cabinet houses two SCSI 5.25" drives and one 1/4" 150 Mbyte SCSI tape drive.

The CPU SCSI controller supports a maximum of seven peripheral devices (disk and tape) on a system. The cabinet supports three internal scsi devices with an external port for up to four external devices.

Communications

- Ethernet TCP/IP,
- VME slot,
- Two RS-232 ports,
- One keyboard port,
- DECnet,
- Fiber-optic network interface (FONI) card

Packaging

The ESV Workstation uses a 7-card cabinet or a 14-card cabinet. Both cabinets are cost effective, as small as possible, and able to operate in an office environment with a 20 amp circuit.

Diagnostics

The ESV Workstation diagnostics test to the chip and card-level for card bring-up and testing. There will be power on confidence tests for field service and customer use, system level tests and FRU tests for installation, field service, and self maintenance trouble-shooting.

Software Summary

System Software

The items listed here are part of the initial software development environment:

- UNIX V with BSD extensions,
- Sun RPC, XDR, and NFS,
- C, FORTRAN, Pascal languages and compilers,
- Source code debugger,
- Additional utility programs,
- FONI card driver and interactive device drivers for control dials, function buttons, and data tablets.

Graphics Interface

An application can use the following graphics libraries:

- PHIGS and PHIGS PLUS
- Xlib
- Xt (X toolkit)
- Motif (OSF's X toolkit)
- PSX - PS 390 emulator

Workstation Graphics System Specifications

Graphics Features

Depth-cueing,
Firmware picking, and
Primitives and attributes.

Shading and Lighting Features

Constant shading,
Gouraud shading,
Ambient lighting,
Diffuse lighting,
Specular lighting,
Twelve light sources,
Colored sources,
Ambient sources,
Directional sources,
Point sources, and
Spot sources.

Performance

MIPS	20
Dhrystones	35.5 K (normal optimization) 41 K (high optimization)
Standard Floating-point	

Architecture

Processor: RISC
Clock Speed: 25 MHz,
Floating point (std.) MIPS,
Memory (min/max): 8/128 Mbyte,
Hard Disk (min/max): 180 Mbyte/ 1.2 Gbyte unformatted,
Tape Drive: 1/4" 150 Mb,

Floating point accelerator,
Configuration: 7-card or 14-card cabinet.

Graphics

Display size: 19",
Resolution: 1280 x 1024,
Colors: 16.7 M,
Bit Planes (std): 48,
z-buffer: 24,
H/W shading support.

2. Hardware Overview

Introduction

This chapter is an overview of the ESV Workstation hardware design. Detailed information for each major subsystem in the workstation is found in subsequent chapters of this manual.

Design Goals and System Features

A primary goal of the ESV Workstation is to combine the following features into a modern workstation package:

- Shadowfax-quality 3D graphics at increased performance levels,
- Realtime high performance z-buffered polygons,
- RISC CPU running UNIX with 20 mips performance,
- PEX and PS 390 software compatibility, and
- Expandable performance on both graphics and CPU.

3D Vector Capabilities

The ESV Workstation provides a superset of the Shadowfax (*i.e.*, PS 390) 3D vector capabilities. As with Shadowfax, antialiasing of vectors is performed with no speed degradation. The ESV Workstation offers a choice of either 2-pixel-wide or 4-pixel-wide vectors, with both types antialiased. The 2-pixel-wide vectors are good for general applications and provide maximum performance, whereas the 4-wide vectors provide identical quality to Shadowfax and are suitable for stereo viewing.

Other vector features which are new for the ESV Workstation include color-blending, z-buffering, and a new blending mode which blends a vector to what is actually stored in the pixel, as opposed to a fixed background color register.

A key to these features is a new VLSI pixel processor chip. Each workstation uses a total of four of these chips. Although the ESV Workstation uses fewer pixel processor chips than Shadowfax (16), the ESV Workstation is able to produce a much greater overall system performance. The workstation is designed to reach a peak rate of between 1 and 2 million short vectors per second.

Polygon Capabilities

The ESV Workstation improves on the PS 390 by adding a hardware z-buffering capability to provide realtime dynamic polygons. The pixel processor supports the rendering of jaggy Gouraud shaded polygons by providing inter-

polygon and z-checking of scanline segments of polygons. The maximum polygon performance is 100,000 4-sided, arbitrarily oriented, Gouraud shaded, 10x10 pixel polygons with one white light source.

CPU Features

The ESV Workstation provides a general purpose UNIX CPU based on a MIPS R3000 RISC microprocessor running at 25, 33, or 40 MHz. The CPU is equipped with a floating-point accelerator, Ethernet controller, SCSI, serial ports, and a VME interface. It has a virtual memory system which, in addition to serving UNIX and general applications, acts as the graphics structure memory.

Software Interfaces

PEX is the native graphics programming library to the ESV Workstation. The workstation is unique in that it has been designed specifically to efficiently run the PEX graphics standard, as opposed to a proprietary interface.

In order to provide compatibility with existing E&S software, a PS 390 interface, PSX, is also provided. It performs equal to or better than the original PS 390.

System Expansion

In order to provide a range of products at first introduction, the ESV Workstation hardware allows for modular expansion of certain key performance-related subsystems. Initially, CPU performance can be increased by adding system memory in the allowed range of from 8 Mbyte up to 128 Mbyte. Also, a range of hard disk sizes is available, from 380 Mbyte up to 1.2 Gbyte. In the future, faster CPUs will be available. Graphics performance can be increased simply by adding cards to the graphics subsystem. Hardware Architecture

Workstation Architecture

Figure 2-1 shows that the system architecture is based around two global buses; the VME bus and the Gbus. The VME bus is intended primarily for I/O and other third-party devices. An example of such a device is the E&S Fiber Link card, a high-bandwidth fiber-optic communications card. The Gbus is a proprietary bus which is used exclusively by the ESV Workstation graphics subsystem.

The figure also shows that the CPU, as it should be, is central to the system and ties these two busses together. The CPU is capable of accessing devices

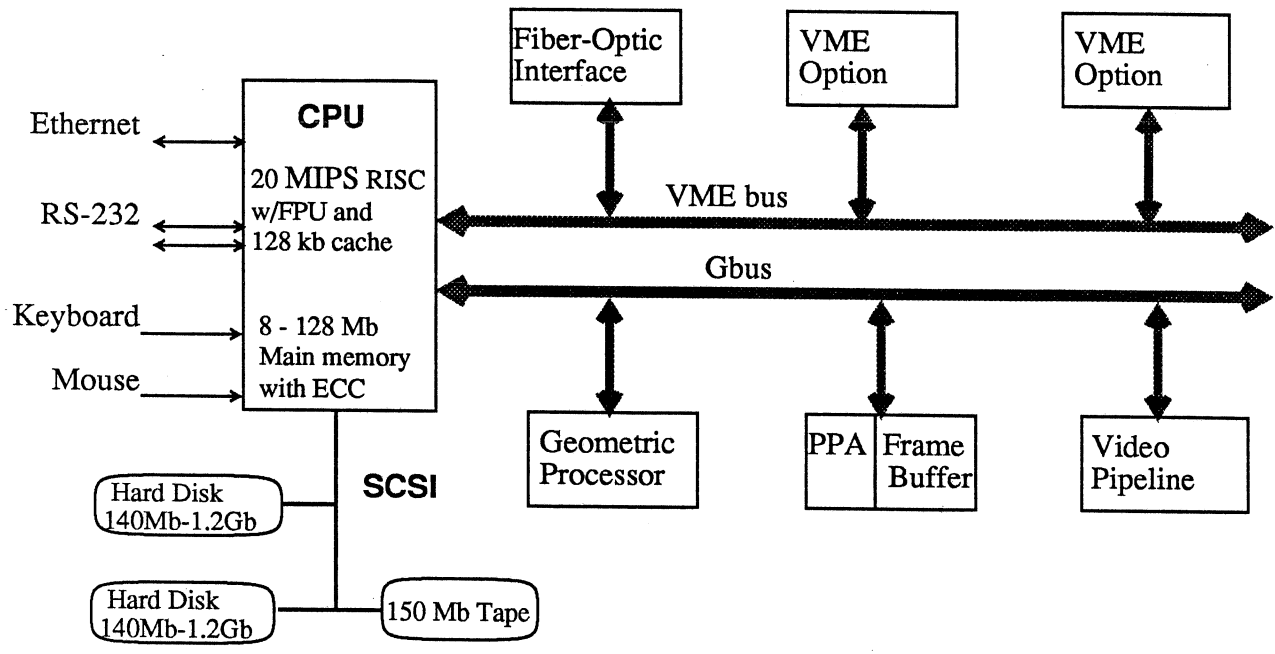


Figure 2-1. ESV Workstation architecture

on either the Gbus or the VME bus. It allows any VME bus master to access the Gbus through an address map. VME bus masters are also capable of accessing all of the physical memory on the CPU.

The three major subsystems in the ESV Workstation (CPU, graphics, VME I/O) are all clock-independent in the sense that one subsystem can be sped up without necessarily having to redesign the other two. It is planned that future products in the ESV Workstation family will come about by upgrading only one or two of these subsystems at a time

CPU Architecture

As mentioned above, the CPU core is implemented with a MIPS R3000 microprocessor. There is a total of 128 Kbyte of high-speed cache memory. System memory ranges from 8 Mbyte up to 128 Mbyte. Memory is increased by adding small surface-mount RAM modules to the CPU card. The memory system provides single-bit error detection and correction. The CPU card includes all of the I/O controllers which are standard with the ESV Workstation; Ethernet, SCSI (disk & tape), RS-232, and keyboard. The CPU also includes a special graphics FIFO which allows the structure walker process to feed graphics primitives to the Gbus.

Geometric Processor Architecture

The geometric processor constitutes the topmost portion of the overall graphics processing pipeline. Figure 2-2 shows that this is implemented as a parallel array of DSP (Digital Signal Processor) units. The Graphics FIFO shown in figure 2-2 is the one described as part of the CPU card. The DSP units process the graphic commands as they become available through this FIFO. The Gbus is used as the transfer medium for data going from the FIFO into the individual DSPs.

The individual DSP unit is based on the AT&T DSP32C chip, which is fully programmable and has a peak performance of 25 Mflops. Each DSP unit has 64 Kbyte of off-chip static RAM, as well as interfaces to both the Gbus and an output FIFO. A newer, higher performance, DSP card is available with extended off-chip RAM of 256Kbyte.

Each DSP processes its graphics commands to the point where it has produced the appropriate command packets for the pixel processor array in the frame buffer. The DSPs output these command packets through a separate output FIFO which buffers commands for the pixel processors. The output FIFO structure is often commonly referred to as the *laundry chute*.

Because all the DSPs are identical, and contain identical programs, the overall graphics performance can be increased by adding more DSPs. The

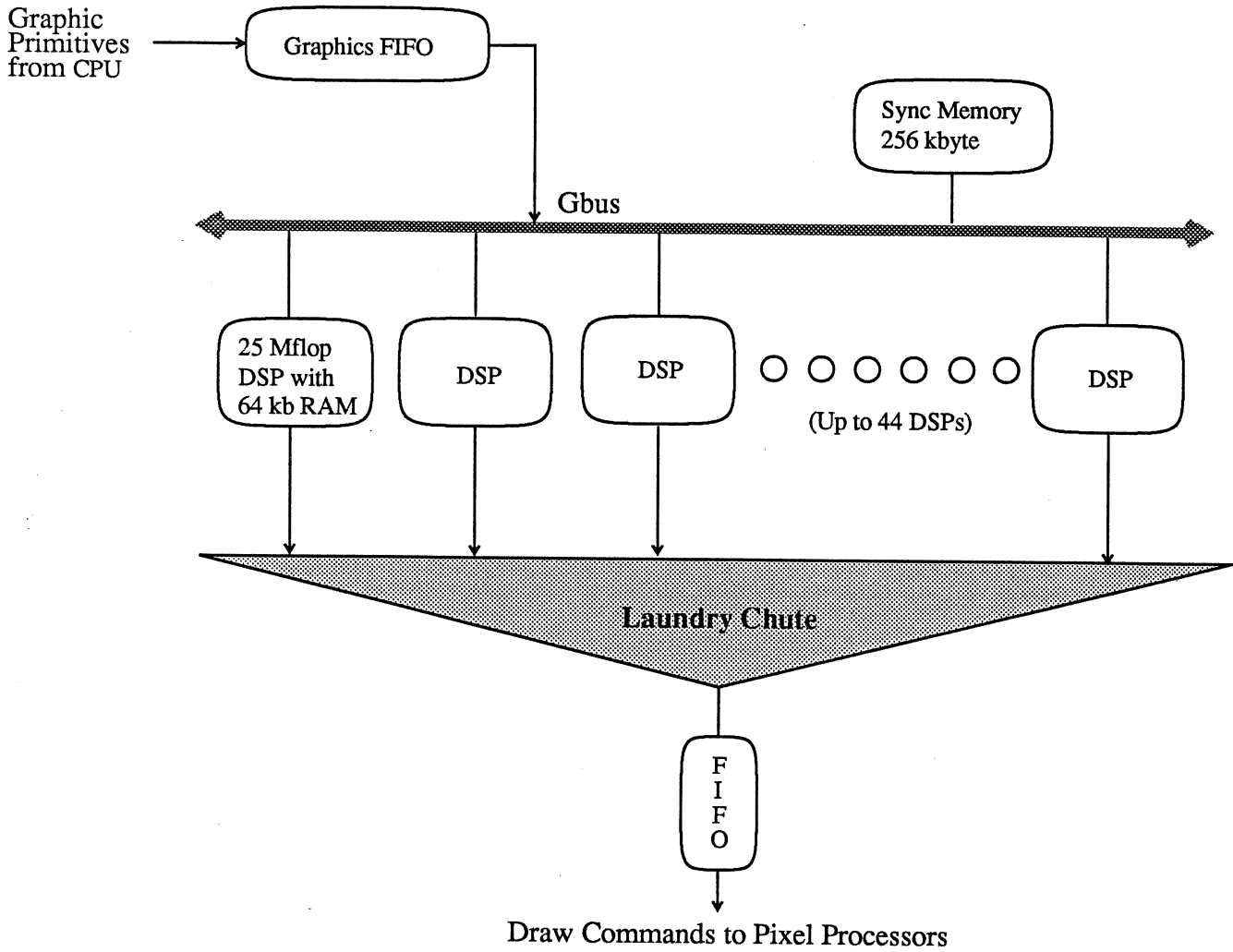


Figure 2-2. Geometric processor architecture

ESV Workstation provides a wide performance range in this regard; the number of DSPs can vary from a minimum of four to a maximum of forty-four.

There is also a small, shared memory which is accessible to all DSPs over the Gbus known as sync memory. It has a capacity of 256 Kbyte. It is used chiefly for maintaining and synchronizing the graphics processing state of the individual DSPs.

Although the Gbus connects to all components of the graphics subsystem, the primary use of its bandwidth during normal runtime is consumed by transfer of command blocks between the CPU's graphics FIFO and the DSPs. Most of the other devices and registers on the Gbus are provided for initialization, maintenance, and diagnostics.

Frame Buffer Architecture

The workstation's frame buffer, as shown in figure 2-3, consists mainly of four primary banks of video RAM memory, each controlled by one pixel processor chip. The four pixel processors are connected via the PPbus, which is normally used to transfer commands from the output FIFO of the DSPs. The PP bus also allows the pixel processors to interchange pixel data during pixel block transfers (BLTs), and provides a readback path should a DSP wish to read pixel data via the Gbus.

All frame buffer configurations provide 1536 x 1024 pixels of total memory, of which 1280 x 1024 can be visible on the screen. The frame buffer is equipped with 88 bits of video ram for each of the pixels that are visible on the monitor.. Figure 2-3 shows the purpose of these bits and their roll in creating the image.. This includes 24 total RGB color planes, 8 window/overlay planes. An additional 8 control planes, known as *valid planes* allow for an extremely quick clearing of an arbitrary window. A second set of 24 RGB color planes provide the full 24-bit double buffered graphics capability. A final bit plane is the addition of 24 z-planes. These are necessary for doing real-time polygon operations.

Video Pipeline Architecture

The video pipeline actually starts in the frame buffer's video RAM chips, which are continually scanning out pixels to refresh the screen (see figure 2-4). The ESV Workstation provides the ability for on-screen windows to individually select which buffer (A or B) is to provide video pixels at any given time. This allows windows to update and swap their double-buffers completely independent of each other. With Shadowfax, all windows had to update and swap together, always drawing video from the same buffer.

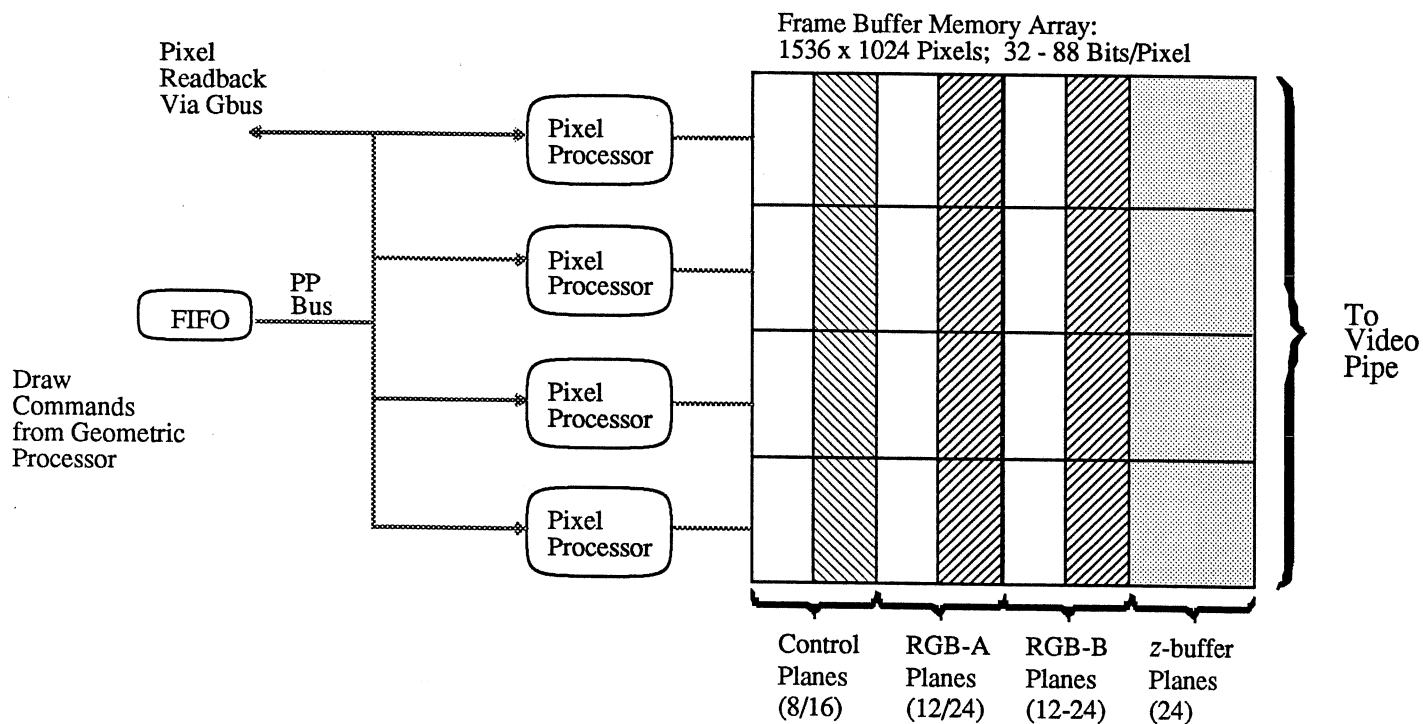


Figure 2-3. Frame buffer architecture

The window lookup table, a small high-speed RAM, provides information that is variable on a per-window basis. This information includes active video buffer select (as above), background color, color-lookup table select, etc. The window controller uses this table to multiplex pixels from the two buffers. A hardware cursor inserts its data into the pixel stream when appropriate. It allows for a 64 x 64 pixel cursor, which can be moved simply by loading a position register. Multiple color lookup tables allow selection on a per-window basis.

Hardware Assemblies

General

All ESV Workstation card assemblies are implemented on 9U high, 400 mm deep Eurocard format PC cards. All backplane connectors are the standard VME 96-pin variety.

Backplane

Currently, there are four varieties of backplane.

- The 7-slot version allows one VME card, one CPU, one Graphics System Executive (GSE), one DSP, two frame buffer, and one video card.
- The 14-slot version allows four VME cards and five DSP cards.
- The 15-slot version that provides for the Local Server option and also supports SUN VME compatibility.
- The test backplane is similar to the normal 14-slot version, except that it allows cards to be inserted from either side, as is done in the ESV Workstation test racks.

Because the (non-vme) slots are dedicated to a card type, a keying scheme is provided which allows only the correct card type to be plugged into any given slot.

CPU Card

The CPU card contains all the components as previously described. It occupies a much wider space than the other card types because the memory card modules which are added to this card protrude vertically. However, the CPU has the ability to accept up to 128 Mbyte of memory in this fashion, and the amount of memory installed does not affect the total usage of card slots.

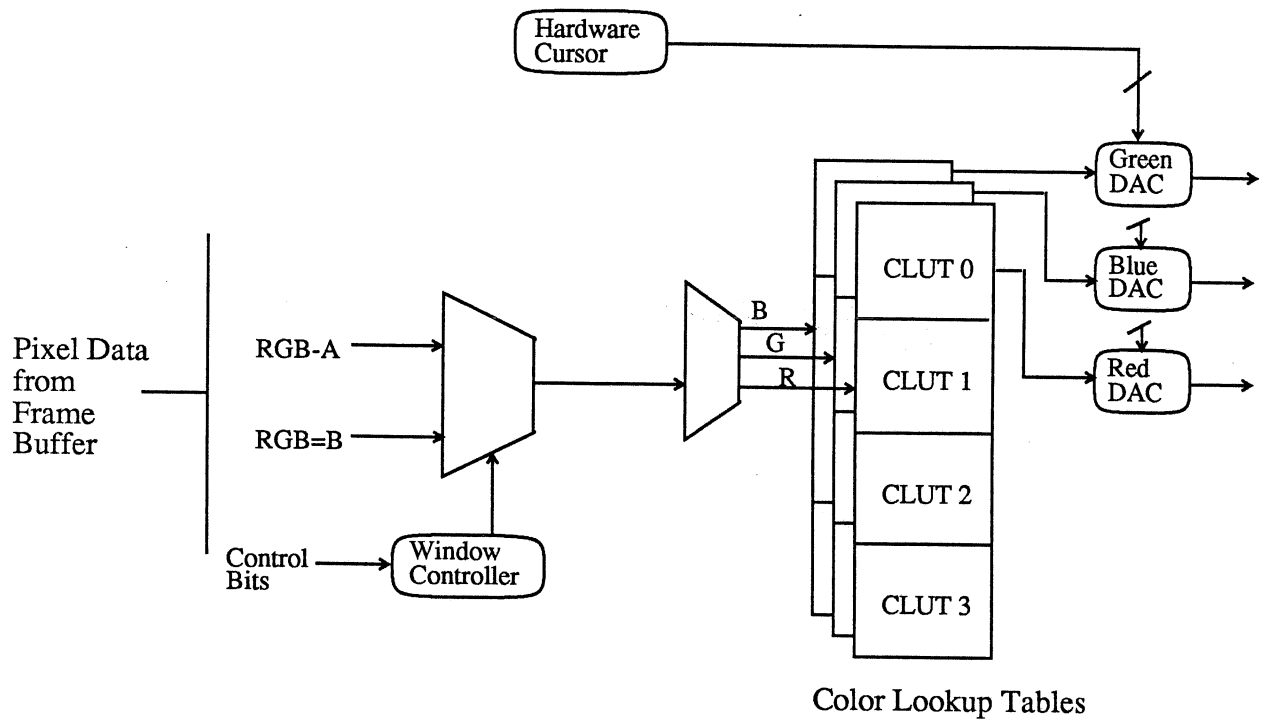


Figure 2-4. Video pipeline architecture

GSE Card

The GSE card is standard with every ESV Workstation system. It contains four of the DSP units and some other miscellaneous that go with the graphics subsystem. The miscellaneous includes sync memory, Gbus arbiters and output FIFO controllers.

DSP Card

DSP cards can be optionally added to the ESV Workstation. Each DSP card contains 8 DSP units. The 14-slot backplane holds up to five DSP cards for a total of 44 DSP units including the four that the GSE contains.

Frame Buffer Card

Every ESV Workstation has two identical frame buffer cards. Each card has two pixel processor chips and their associated banks of video RAM memory. The frame buffer card provides the upper level of the video pipeline, including the section that multiplexes pixels from the A & B buffers. Each frame buffer card outputs a video pixel to the backplane every 18 ns, for a total maximum video speed of 9 ns/pixel.

Video Card

The video card is standard for every ESV Workstation. It contains part of the window lookup table, the hardware cursor, and the RAMDACs, which include the color lookup tables.

Fiber Link Card

The Fiber Link card is an E&S implementation of an FDDI standard fiber-optic token-ring communications controller. It can be added as an option to the ESV Workstation, providing network communications at a bandwidth of 100 Mbits/sec. It occupies one of the VME I/O card slots. The Fiber Link card Users Guide and Installation Manual should be consulted for more information.

Monitor

The ESV Workstation monitor is a Mitsubishi 64 KHz multisync device which allows the display of 1280 x 1024 pixels in a noninterlaced format. It can automatically synchronize with an interlaced format which is used for the ESV Workstation stereo implementations. The monitor has a bonded screen that reduces glare and reflections. Adjustment and Care of the monitor will be discussed in Chapter 8.

Keyboard

The keyboard is a modified version of the IBM PC 101-key standard layout. The modifications include relocation of the CTRL key, and the addition of an audible beep/click circuit. The keyboard connects with the standard PC connector and synchronous protocol.

Mouse

The mouse is a three-button optical variety. The communication protocol is RS-232 at 1200 baud.

Dials

The ESV Workstation dials are the same unit as provided with CDRS, the LED-label variety.

Tablets

The WACOM cordless tablets in various sizes are used on the ESV Workstation. Also, Summagraphics 12 x 12 and 6 x 9 tablets are supported.

Reprogrammable Data Concentrator (RDC)

The RDC consists of a circuit card with six interactive device ports, one debug port, and one host computer port. The RDC is optional because the mouse, keyboard, and two other RS-232 devices can be connected directly to the ESV Workstation cabinet without it. If the RDC is used, all peripheral devices may be connected to it.

System Performance Data**CPU**

20 MIPS, 4 Mflops, 42,000 Dhrystones/sec (25 Mhz)

I/O System

VMEbus @ 8 Mbyte/sec

Graphics

Over 1 million short vectors/sec (8 pixel vectors), 100,000 polygons/sec (10x10 polygons)

Overall Product Parts Structure**Standard**

- CPU card w/ 0 Mbyte DRAM
- GSE card
- Video card
- Keyboard
- Mouse
- Monitor w/cables
- 150 Mbyte cartridge tape drive

Mandatory Additional

- CPU memory module set (choice of 8 Mbyte, 16 Mbyte, 24 Mbyte, 32 Mbyte, 64 Mbyte, 96 Mbyte, 128 Mbyte)(memory choices are limited to 16 Mbyte, 32 Mbyte, 64 Mbyte, or 128 Mbyte with the 33/40 Mhz CPUs)
- Cabinet (choice of 7-slot, 14-slot, or 15-slot)
- Hard disk (choice of 380 Mbyte, 760 Mbyte, or 1.2 Gbyte)
- Frame buffer 2-card set.

Options

- DSP card(s)
- Second hard disk drive (180 Mbyte, 380 Mbyte, 760 Mbyte or 1.2 Gbyte)
- Fiber Link card
- RDC
- Stereo
- Dials
- Tablet

3. Getting Started

Preparation for Booting

Booting the workstation is a simple procedure. Before starting the booting process, please check the following:

- 1) Make sure that the control unit and display are turned off.
- 2) Check that the control unit and display power cords are plugged into active wall outlets of adequate capacity. The large cabinet ESV requires a dedicated 20 amp source.
- 3) Mouse should be connected to the control unit CPU I/O control panel port 1 or to the RDC port B. If the system includes an RDC, the RDC connects to the CPU I/O control panel on port 1. This port is also used for the debug terminal. (see figure 3-1 on the next page)
- 4) The keyboard and all other peripherals should be connected to the correct ports. Exactly where the system expects the device is part of the configuration information in the special device files. All peripherals can be configured to work through the CPU's I/O port or through the RDC. The default configuration includes keyboard, monitor, and mouse, but does not include an RDC. (see the chapter on peripheral device configuration)

Figure 3-1 shows the default configuration for plugging in peripherals in a system without an RDC. Figure 3-2 shows which RDC ports the peripherals use.

Note: In systems with RDCs, the mouse must connect to the port on the RDC; it cannot plug into the port on the cabinet.

- 5) Display is connected to coax connectors marked RGB on the cabinet. Composite sync is carried on the green signal, making it unnecessary for a separate sync connection under most circumstances.

Debug Terminal

The debug terminal is an optional port to get diagnostic and bootup messages. The debug terminal should be handy to use if there is some problem that prevents the graphics subsystem from being used. The debug terminal should be some ASCII terminal with an RS232 interface that will plug into the mouse/RDC port on the I/O panel. The Boot PROM defaults to 1200 baud for the debug terminal port due to the ports normal function of being the mouse port. The baud rate of this port is reconfigured during the boot to 9600 baud. This can be troublesome, because the baud rate being different at the boot prom than it is at a UNIX run level..

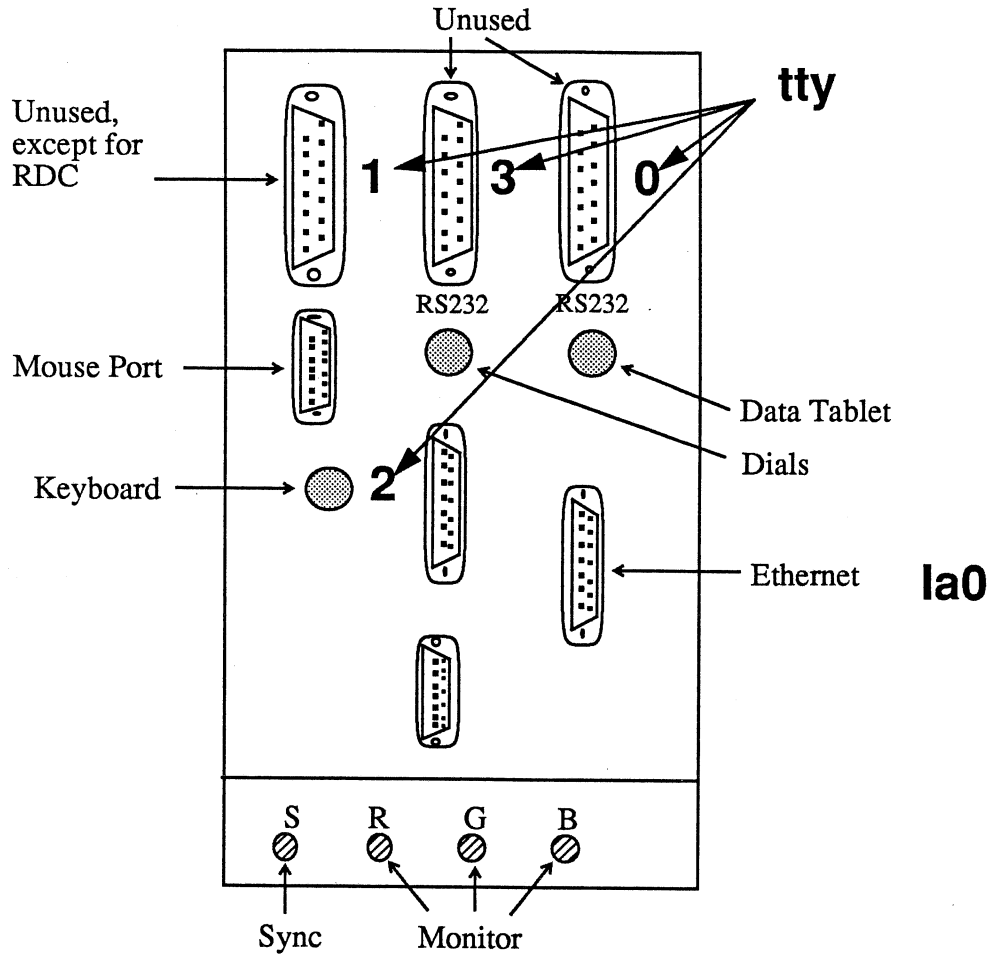


Figure 3-1. Default Cabinet Port Configuration (without RDC)

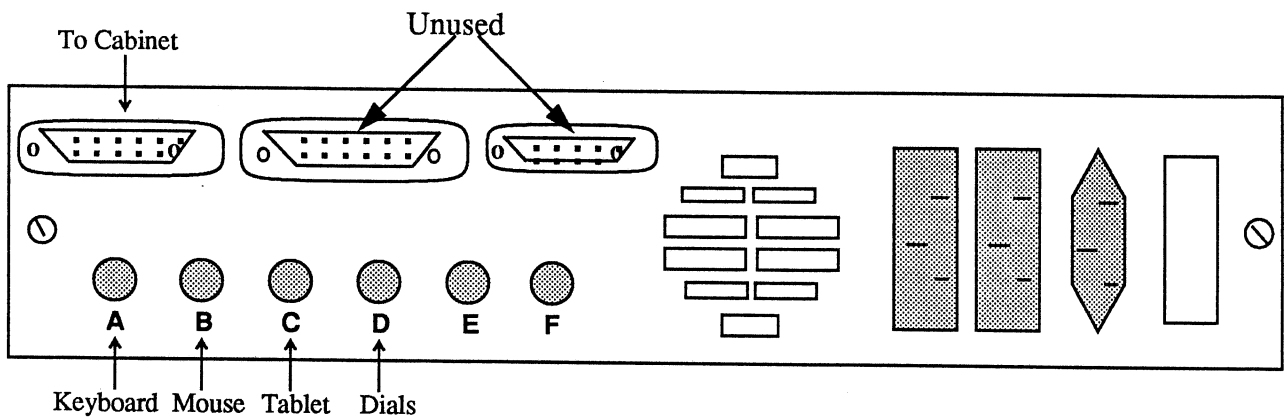


Figure 3-2. RDC Port Configuration

Normally the system does not print anything to the debug port. It can be enabled by generating a "break" signal any time after power up as long as there are no graphics applications running. Also, the debug terminal is automatically enabled if any of the graphic subsystem tests fail.

Once enabled, everything that is printed on the graphics display is also sent to the debug terminal. Also, the debug terminal can be used for input as well as the system keyboard. To disable the debug terminal, you must change the environment variable **console** to **l**.(lower case L) It should be **l** for a machine with graphics and an **s** for a machine without graphics (*i.e.*, server), **r** to force the enable of the debug port..

It is not possible to use the debug terminal while running a graphics application. The application needs the mouse or RDC to be hooked up instead of the debug terminal. The primary use of the debug terminal is for running diagnostics on the graphics subsystem.

General Behavior

The start-up process from the console consists of four steps, and the entire process is complete in approximately two minutes. A slight variance in the time is due to the number and size of the disks that the program must check. The steps in the process are:

- 1) Running non-visible confidence tests,
- 2) Running visible confidence tests,
- 3) Booting UNIX, and
- 4) Starting a graphics application.

Turn on the display, then the control unit. Turning on the workstation's power switch begins the non-visible confidence tests. These tests occur before the graphics subsystem is initialized to print messages. They perform quick checks for major problems in the graphics subsystem. If these confidence tests pass, the graphics subsystem is initialized to print out further messages, and visible confidence tests begin running automatically.

When visible confidence tests begin, a system banner is printed on the graphics display indicating that the power on confidence tests are being run. Each test prints out a title on the console terminal and a PASSED or FAILED status upon completion. After all the tests are run, memory is cleared and the boot PROM prompt is displayed as shown:

>>

Test Descriptions

Each confidence test writes a specific LED pattern when the test is entered. If an error occurs, a specific LED failing pattern is flashed for about six seconds and a failing message is written to the console (if it's a visible test). The failing code is also written to the PON-MASK location in the NVRAM. The tests proceed regardless of a failure (except an early exception). If another error occurs, the LEDs flash but the NVRAM PON-MASK location is only written over if the new error is of a higher priority than the previous error.

Non-visible Confidence Tests

The purpose of power-on testing is to provide you with reasonable confidence that your system functions correctly. An outline of the process is as follows:

- 1) When you turn the system on, the CPU begins executing at the start of PROM.
- 2) It initializes status and CPU registers.
- 3) It calls the memory configuration routine, **Imem_config()**.

The routine **Imem_config()** performs the "soft" configuration of the memory boards. As it runs, it calls other routines and writes various LED patterns to indicate its progress. Successful completion of this routine indicates the following is true:

- The CPU can execute instructions.
- The write buffers are functional.
- Some CPU I/O decode logic is functional.
- The VME bus is communicating.

The following is a list of the non-visible confidence tests that the system runs:

- Enable Hardware
- Test UART
- Initialize UART
- Register Test of Video Card
- Initialize Video Card
- Register Test of GSE Card
- Initialize GSE Card
- Initialize Pixel Processors

- Initialize Frame Buffer
- LED Test

Visible Confidence Tests

This section contains information about the visible confidence tests which are the next diagnostic tests run on the system. These are still low-level tests. If an exception occurs at this level, the system is unusable. Should an exception occur, the LED value indicates which test was being run at the time of the failure.

Each test is followed by the PASSED or FAILED indication. The listing below shows what is displayed on the screen, beginning with the diagnostic banner, as these tests are run and passed.

```
Running Power-On Diagnostics. . .
Cache Test #1. . .PASSED
Cache Test #2. . .PASSED
Data Cache MATS+ Test. . .PASSED
Instruction Cache MATS+ Test. . .PASSED
Data Cache Block Refill Test. . .PASSED
Instruction Cache Block Refill Test. . .PASSED
Write Buffer Test. . .PASSED
Memory Test . . .PASSED
TBL Test. . .PASSED
Exception Test. . .PASSED
Instruction Streaming Test. . .PASSED
EDC Test. . .PASSED
Battery Check Test. . .PASSED
NVRAM Test. . .PASSED
Timer Test. . .PASSED
Time-of-Day Clock Test. . .PASSED
Duart Port Tests. . .PASSED
FP Test #1. . .PASSED
FP Test #2 . . .PASSED
Lance Slave Register Test. . .PASSED
Lance Master Test. . .PASSED
```

You can use the following command to see the value of all the PROM environment variables:

```
>>printenv
```

Part of the installation process should include setting the environmental variables at the PROM prompt that suit the user and the location. Variables to consider are `netaddr`, `console` and `bootmode`.

At the boot PROM prompt, you can start UNIX booting by typing `auto`. If the PROM environment variable `bootmode` is set to `c`, typing `auto` is unnecessary. With `bootmode` set to `c` the PROM automatically tries to boot UNIX after powerup. The failure of any confidence test, however, sets `bootmode` to `e`, and the environment variable requires reset to allow automatic booting again. The variable is reset by entering the following command:

```
>>setenv bootmode c
```

When the UNIX initialization is complete, a login prompt is displayed. At this point you can login and use the system as an ANSI terminal or start the X window display manager, `xdm`. You can also start a graphics application that controls the display. The terminal emulator remains dormant as long as the graphics application has control of the display. As soon as the application exits, the display is cleared and the terminal emulator starts.

The boot time is also the opportunity for the system administrator to load release tapes and change system configuration information. For detailed information on system administration, consult the following documents:

- *RISC/os System Administration Reference Manual*
(Part No. 400403-100)
- *RISC/os System Administrator's Guide*
(Part No. 400404-100)

Booting UNIX

Booting is the process of reading the UNIX system kernel, usually stored in `unix` on System V into the system memory and starting it running.

When all the power on confidence tests have executed, the system clears memory, initializes the system keyboard, and displays the boot prompt. At the boot prompt, `>>`, either UNIX begins booting automatically, or you type `auto` to start UNIX booting. The listing below is a sample boot prompt.

```
Autoboot: Waiting to load dkis(0,0,8)sash (CTRL-C  
to abort, RETURN to expedite)loading
```

```
118400+21680+170416 entry: 0xa0300000
MIPS Standalone Shell Version 4.10 MIPS OPT Sat Jan
20 13:22:32 MST 1990
```

Optional action that you can take at the boot prompt includes loading and running programs from disk, tape or Ethernet. In the example above, the program, **sash**, is run. This is the program that boots UNIX. Usually, commands are issued at the boot prompt only when installing a new system.

All input comes from the system keyboard (or debug terminal) and is displayed on the graphics display. The display is treated like a simple ANSI terminal with 80 columns and 40 lines.

The system keyboard can be hooked up in two ways. The first is to plug it into the keyboard port on the I/O panel of the system, and the other is to plug it into the RDC. When the system initializes the keyboard, it looks for it first on the I/O panel and, if it doesn't respond there, the system checks the RDC. If no keyboard is found, a message is printed to that effect and the boot PROM will sit and do nothing. Otherwise the keyboard will be initialized and used as the console input device.

As UNIX is booting, it prints messages, then prints a login prompt at the completion of booting.

The listing below is an example of the messages that UNIX prints on the screen as it boots. Several items in these messages vary from system to system: addresses, memory sizes, number of files, IDs, and system names.

```
Loading dkis(0,0,0)/UNIX
718576+85328+497648 entry: 0x80021000
CPU: MIPS R3000 Processor Chip Revision: 2.0
FPU: MIPS R3010 VLSI Floating Point Chip Revision: 2.0
```

```
ES/OS Release 1_0 ESV Version R_100
```

```
Total real memory = 25165824
```

```
Available memory = 23425024
```

```
root on dev 0x1000 (fstyp is ffs)
```

```
Available memory = 22564864
```

```
Checking root file system () automatically.
```

```
The system is coming up. Please wait.
```

```
*****Normally all file systems are fscked.
```

```
*****To fsck only dirty ones, type 'yes' within 5 seconds:
*****All file systems will be fscked.
mountall: fscking /dev/usr (/usr).
** /dev/usr
** Last Mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference counts
** Phase 5 - Check Cyl groups
5632 files, 58523 used, 23249 free (201 frags, 2881 blocks,
0.2% fragmentation )

*****FILE SYSTEM WAS MODIFIED *****
/dev/usr/mounted on /usr
Saving system core dump
Internet daemons:  routed portmap inetd rwhod.
NFS daemons:  nfsd biod.
systemname: /dev/dsk/isc0d1s4 mounted on /usr1
systemname: /dev/dsk/isc0d1s6 mounted on /usr2
The system is ready
Console login:
```

Logging On

At this point, may enter your login name and password or start the X display manager by typing xdm at the login prompt. The console, prior to starting X, behaves like an ANSI terminal and can be used with screen based editors.

4. CPU Card

(217110-100 25Mhz, 217165-100 33Mhz, 217165-101 40Mhz)

Objectives

At the end of this chapter you should understand the following:

- The CPU card subsections and their functions.
- How the CPU interacts with the rest of the workstation.
- What the CPU's purpose is in relation to the graphics portion of the workstation.

Introduction

The ESV Workstation CPU card is the central controller in the ESV workstation family. Its main purpose is to run the UNIX operating system and application software. It is also an integral part of the graphics subsystem of ESV; display structures are maintained and traversed out of virtual memory by this CPU.

The major features of the ESV Workstation CPU card are:

- 25 MHz R3000 RISC CPU yielding 20 VAX MIPS, 8 MFLOPS/
33/40 Mhz R3400A yielding 27/33 VAX MIPS, 11/13 Mflops
- 128 Kbyte cache memory (64K instruction, 64K data)
- 8 Mbyte main memory, expandable to 128 Mbytes
- SCSI controller for hard disk and tape
- Ethernet controller
- 4 serial ports and 2 timers
- Time-of-day clock and 2K-byte non-volatile RAM with battery back-up
- VME bus master/slave interface
- Coprocessor interface to ESV Workstation graphics processors
- Dual Port Ram interface for processor to processor communication is equipped on the 33/40 Mhz version.

A block diagram of the CPU card is shown in Figure 4-1.

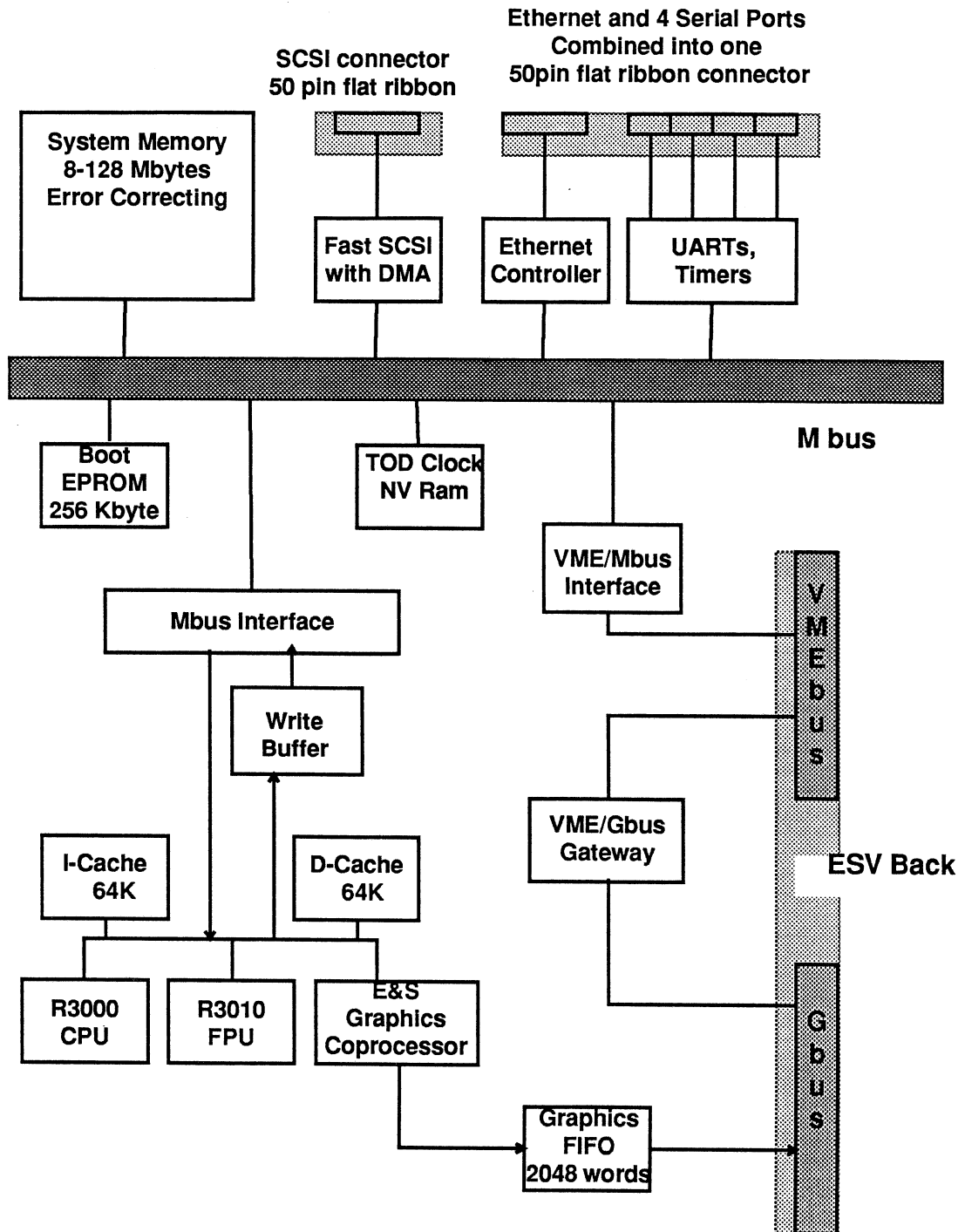


Figure 4 - 1 CPU Card Block Diagram

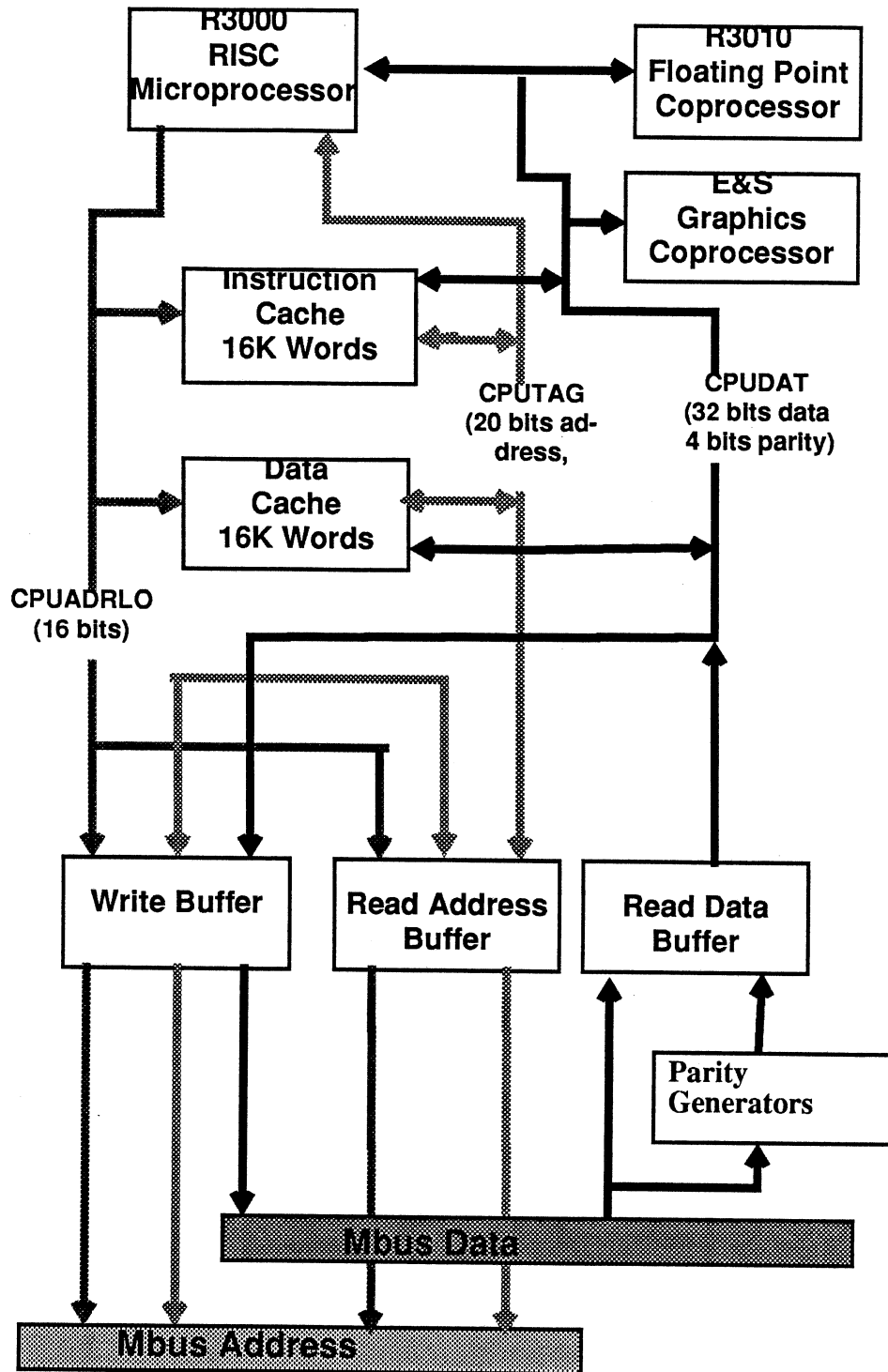


Figure 4 - 2 CPU Core Block Diagram

CPU Core

The CPU core subsystem consists of the following pieces:

- R3000/R3400A RISC processor,
- R3010 floating point coprocessor,
- 128K bytes of high speed cache memory (64K bytes instruction, 64K bytes data),
- E&S graphics coprocessor,
- Write buffer,
- Read buffer,
- Mbus interface controller.

These sections are interconnected via a high-speed bus with 200/320 Mbyte/sec peak bandwidth. The interconnection of the CPU core section is shown in Figure 4-2.

R3000 RISC Processor

The MIPS R3000 processor is a 144-pin PGA that contains the integer processor and coprocessor 0 of the MIPS RISC architecture.

The R3000 does all the instruction and operand fetching for itself as well as for the floating point coprocessor and the graphics coprocessor.

R3010 Floating Point Coprocessor

The MIPS R3010 floating point accelerator is an 84-pin QuadCerPack that handles floating point operations for the CPU. The floating point accelerator operates as coprocessor 1 to the CPU, monitoring the instruction stream for a floating point accelerator instruction, and reporting results synchronously to the CPU.

Data and Instruction Caches

The R3000 supports separate instruction and data caches. Each cache is 64K bytes.

Write Buffer

In order to minimize the impact of writes on processor operation, a write buffer allows the R3000 to retire writes to the Mbus at cache speeds. This buffer is implemented using four R3020 write buffer chips. Each chip handles eight bits of the CPU data bus, and eight bits of the CPU address bus.

Note: With the 33/40Mhz CPU the write buffer function is performed by the LSI 3230 (Read/Write Buffer).

Read Buffer

The R3000 receives instructions and data from the Mbus via the read buffer which consists of a 34-bit address register and a 36-bit data register. This provides the critical output-enable speed required by the R3000.

Mbus Interface Controller

This interface controls all communication between the R3000 and the various devices resident on the Mbus.

E&S Graphics Coprocessor

An important feature of the ESV Workstation and one which distinguishes it from previous E&S interactive displays is that the ESV Workstation graphics display structure resides in the system's virtual memory. This is accomplished by having the CPU do the traversal of this structure, and using the R3000's coprocessor interface to load the graphics input FIFO. Although it is generally true that the structure will not fit into the cache, the ESV Workstation design takes advantage of the fast block-refill interface which exists between the data cache and the main memory system. By using the coprocessor load instruction, we can send one 32-bit word down the graphics FIFO each 40ns cycle when the data is in cache. When the data is not in cache, the processor halts and the cache block-refills at 40ns/word, plus latency (400ns). Because the block size is 16 words, the ESV Workstation runs at 105 ns/word overall if the structure does not fit into the cache.

The graphics FIFO queues up graphic command primitives from the E&S coprocessor and buffers them for output onto the graphics bus. The FIFO is 2048 words deep. The large size enables the structure-walking process to proceed without frequent stops and starts. The FIFO can be configured to interrupt the R3000 on going full, on going almost full (32 words left), or going half empty. This allows UNIX to deschedule the structure walking process when the FIFO has gone full, and to re-schedule it when the FIFO has gone down to half-full.

The coprocessor hardware consists of an instruction decoder and FIFO loader, a 2048-word FIFO, and an output register and output control PAL.

Graphics Coprocessor Condition Flags

The E&S graphics coprocessor subsystem provides four condition flags to tell the CPU how full or empty the FIFO is. The four signals provided are listed below:

- *COPFIFOEF

This signal is asserted low when the FIFO is empty. It is low after *COP3RESET is asserted, and after the last word is unloaded from the FIFO.

- *COPFIFO F

This signal goes low when the FIFO is full.

- *COPFIFOHF

This signal is low whenever the FIFO is at least half full.

- *ALMOST_FULL

This signal goes low whenever there are fewer than 32 available

In addition to the graphics coprocessor condition flags, a two-way interrupt system exists between the CPU and the DSP farm. The CPU can post an interrupt to the DSP farm by setting a bit in the YAM register.

Mbus

The Mbus is the private system bus used by the R3000 processor to communicate with its main memory, EPROM memory, time-of-day memory, serial ports, SCSI interface, Ethernet interface, and the VMEbus interface. See Figure 4-3 for a block diagram of the Mbus.

The Mbus has 32 bits of address and 32 bits of data along with several bus control signals. It is a parallel, 25-MHz synchronous bus.

Due to excessive loading, a buffered portion of the Mbus is implemented for all of the slower slaves that reside on the bus. This buffered bus is the SLOWbus. Slaves on the SLOWbus include EPROM and NVRAM, DUARTS, and maintenance registers.

Error Detection and Correction Logic

Because this memory is implemented with 1-Mbit or 4-Mbit dynamic RAMs, and because high reliability is required, the memory system includes an error-correcting circuit. This circuit detects and corrects single-bit errors on entire 32-bit words. An additional seven syndrome bits (or checkbits) are stored with each 32-bit word to provide the needed error-correction information.

A double bit error is not correctable, and should be considered a fatal error by the operating system.

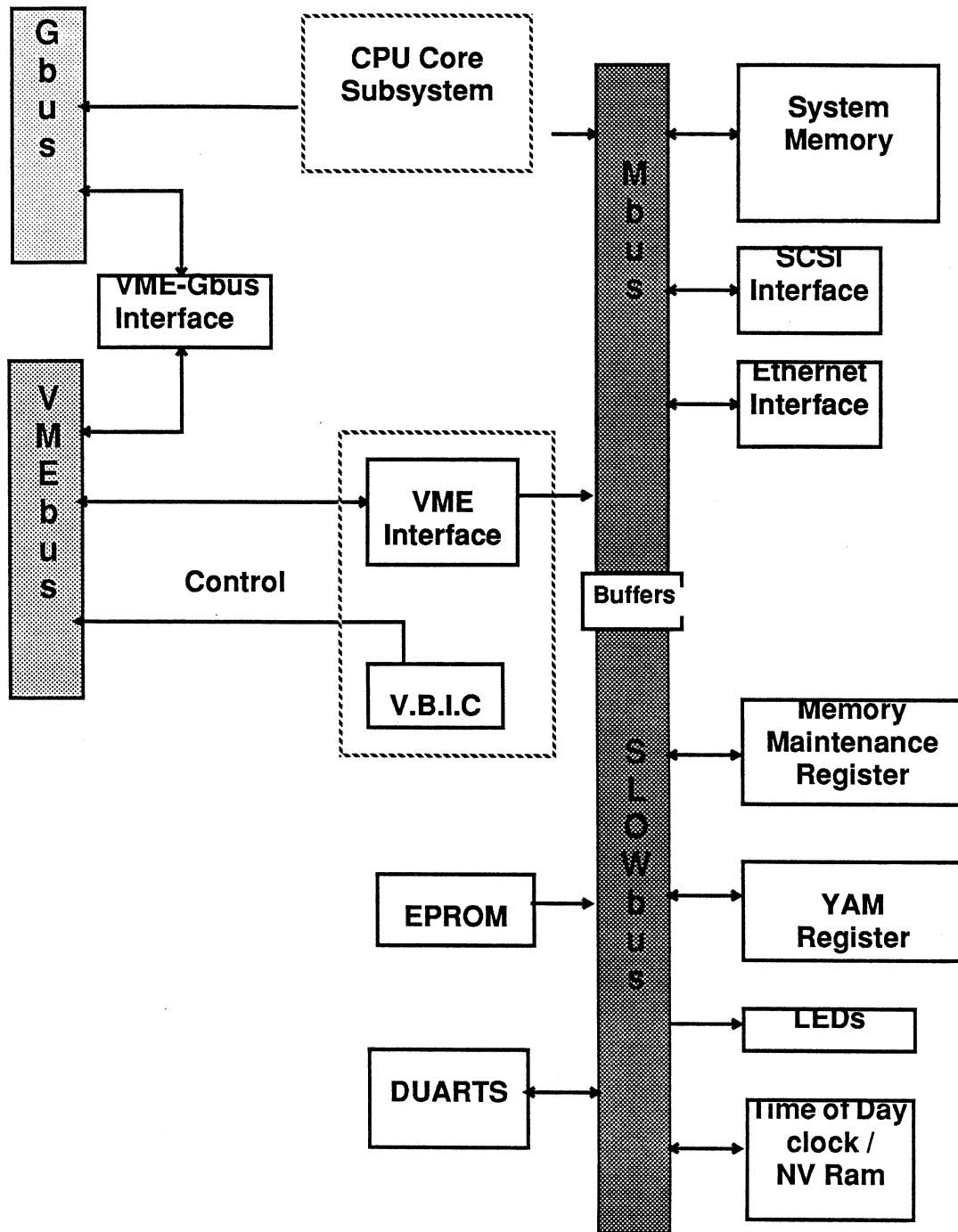


Figure 4 - 3 Mbus Block Diagram

Memory Modules

(217111-100: 4Mbyte 25Mhz)
(217112-100: 16Mbyte 25Mhz)
(217162-100: 4Mbyte 25/33/40Mhz)
(217163-100: 16Mbyte 25/33/40Mhz)

The actual memory portion of the memory subsystem is implemented with vertical plug-in surface mount memory modules. There are two different sizes of module: 4 Mbytes and 16 Mbytes. Each module contains 4 or 16 Mbytes of DRAM plus 1 or 4M x 7 bits of checkbit memory. The system must contain all 4 Mb modules or 16 Mb modules. A mixture of each will not work. Each memory module serves as one leaf of a bank of memory.

The faster 33/40Mhz CPU requires faster system memory. The system memory on the 33/40 is a performance-enhanced version of the 25Mhz CPU's system memory. For reads, the memory acts as a single, 4-way interleaved, static column memory with a peak bandwidth of 160 Mbytes/s. On writes, the memory acts as two independent, 1-way interleaved memories with a combined peak bandwidth of 40 Mbytes/s. This is a significant improvement over the write performance of the 25Mhz CPU.

Figure 4-4 shows a block diagram of system memory.

Troubleshooting System Memory (25Mhz CPU)

The ESV has the option of 8-128 megabytes of system memory. There are 8 memory module connectors on the CPU card. Memory modules for the ESV are one of two types. Module 217111-100 contains 4 Mbytes of memory per module. The 217112-100 contains 16 Mbytes per module. The amount of system memory the ESV has is based on the number of module pairs installed on the CPU card. Modules must be added or removed in pairs and the two different types of modules can not be mixed. There are two jumpers that must be moved if the type of module is changed.

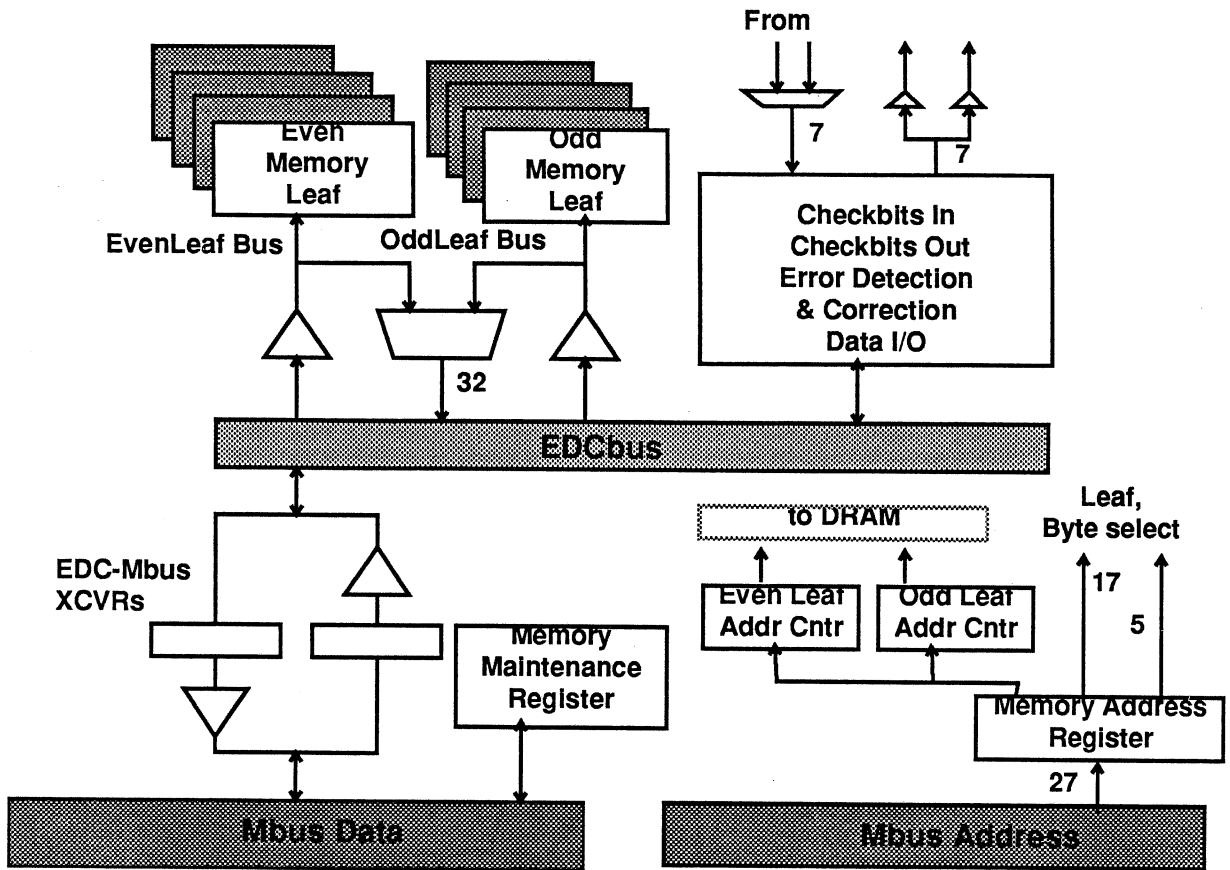
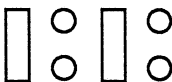
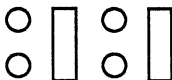


Figure 4 - 4 ESV Workstation System Memory Block Diagram

Memory Module Type Selection Jumpers

U900  Jumper placement when using
4 mb modules (217111-100)

U900  Jumper placement when using
16 mb modules (217112-100)

The system memory is diagnosed as part of the powerup sequence. Should one of the modules fail the diagnostic an address of the failure, the expected and received value of memory will be displayed. The address of the failure can be used in most cases to point to the faulty memory module. Words in memory are interleaved between the modules. Modules installed at connectors P11, P12, P13, and P14 store the odd words in memory. Modules installed at P21, P22, P23, and P24 store even words in memory.

Remember, the address displayed is a byte address. So, ignore the first two bits of the address when determining if the address is an even or odd word. The least significant hexadecimal number of the address will be 0, 1, 2, 3, 8, 9, A, or B if the address is an even word and 4, 5, 6, 7, C, D, E, or F if it is an odd word.

ODD WORDS ARE ON P1() MODULES.
EVEN WORDS ARE ON P2() MODULES.

The following table shows the area of uncached memory covered by the module connectors both 4mb and 16mb:

4mb:	16mb:
P11&21 A0000000-A07FFFFFFF	P11&21 A0000000-A1FFFFFFF
P12&22 A0800000-A0FFFFFFF	P12&22 A2000000-A3FFFFFFF
P13&23 A1000000-A17FFFFFFF	P13&23 A4000000-A5FFFFFFF
P14&24 A1800000-A1FFFFFFF	P14&24 A6000000-A7FFFFFFF

Troubleshooting System Memory (33/40Mhz CPU)

The double sided surface mount modules are nearly identical to the for the 25Mhz. The main differences are the addition of a two-bit code which ones used provides installation and module size information to the memory controller. This eliminates the need for the "module type" jumpers on the 33/40 Mhz CPU. Also, the speed of the DRAMs is faster, 80ns as opposed to 100ns on the 25Mhz. The new modules are backward compatible with the 25Mhz CPU. However, the older 25Mhz modules are not forward compatible with the 33/40Mhz CPU.

Valid Configurations:

16Mbytes...4Mbyte modules in P21,P22,P11,P12

32Mbytes...4Mbyte modules in all slots.

64Mbytes..16Mbyte modules in P21,P22,P11,P12

128Mbytes..16Mbyte modules in all slots.

Determining a module failure during the power-on diagnostics is of course different from the 25Mhz modules due to the 4-way interleaving...

If you should have a failure during the power-on diagnostics for system memory, the error should give you the hex address of the failure. Using the failed address you should be able to determine the failing memory module. The least significant digit of the 8 digit address and the address range, based on the type of module, 4Mb or 16Mb, will tell you which of the modules contains the failing address.

1) Determine the least significant digit of the failing address: If the failing address equaled: 0x12345678, 8 would be the value of the least significant digit.

2) Use the following tables to determine the failing module based on the address of the failure and the type of modules installed.

4 Mbyte Memory Module (217162-100) Table

If the least significant digit equals:	failing address is between:	suspect module is in slot #
0,2,4,6	0xA0000000 & 0xA0FFFFFF	P21
	0xA1000000 & 0xA1FFFFFF	P23
1,3,5,7	0xA0000000 & 0xA0FFFFFF	P11
	0xA1000000 & 0xA1FFFFFF	P13
8,A,C,E	0xA0000000 & 0xA0FFFFFF	P22
	0xA1000000 & 0xA1FFFFFF	P24
9,B,D,F	0xA0000000 & 0xA0FFFFFF	P12
	0xA1000000 & 0xA1FFFFFF	P14

16 Mbyte Memory Module (217163-100) Table

If the least significant digit equals:	failing address is between:	suspect module is in slot #
0,2,4,6	0xA0000000 & 0xA3FFFFFF	P21
	0xA4000000 & 0xA7FFFFFF	P23
1,3,5,7	0xA0000000 & 0xA3FFFFFF	P11
	0xA4000000 & 0xA7FFFFFF	P13
8,A,C,E	0xA0000000 & 0xA3FFFFFF	P22
	0xA4000000 & 0xA7FFFFFF	P24
9,B,D,F	0xA0000000 & 0xA3FFFFFF	P12
	0xA4000000 & 0xA7FFFFFF	P14

SCSI Controller

In order for the CPU to have access to local disk and tape drives, there is an on-card SCSI controller. The controller used is the NCR 53C90 SCSI chip. This device handles all transactions with the SCSI bus. The data rate is about 1-2 Mbytes/sec.

The SCSI controller has the ability to DMA into main system memory over the Mbus, and to interrupt the R3000. The SCSI cable connector is on the card's I/O panel. It is a 50-pin flat ribbon connector, labeled P5.

Ethernet Controller

The CPU card has a standard Ethernet controller for network communications and possible diskless node operation. This is implemented with the AMD LANCE chip and the companion SIA serial chip. The LANCE operates by reading command lists out of CPU system memory, and transferring data packets to/from the same memory. The LANCE requires a memory bandwidth of 10 Mbits/sec to avoid data overruns on ethernet. This translates to 625,000 memory cycles per second. The Mbus arbiter therefore gives highest priority to the Ethernet interface.

The Ethernet transceiver cable connector is located on the cabinet's connector bulkhead. The bulkhead is connected to the CPU card through a ribbon cable which attaches to the connector labeled P4 on the CPU card.

VMEbus Interface

The ESV Workstation system is based on a 32-bit VME bus. The CPU card has a VME interface to allow it to control graphics and other I/O devices in the system. The VME interface allows the main system memory to appear as a slave device to any other VME master. In addition, the CPU card acts as the VME interrupt handler and Slot 1 "System Controller", providing the functions of bus arbiter and IACK daisy-chain driver. The VME interface is designed around the VBIC chip (VME Bus Interface Controller) from SBE, Inc. The VBIC handles much of the VME bus signal timing, the Slot 1 Controller functions, and the interrupt handler interface to the CPU.

VME/Gbus Gateway

The CPU card contains an interface between the VME bus and the Graphics bus. This circuitry maps the 8 megabyte address space of the Gbus into a region in 32-bit VME address space.

The gateway acts only as a slave device on the VME bus, and allows only 32-bit data access (no individual byte cycles). The CPU uses this path to load the DSPs with microcode, and to access sync memory while the DSPs are running.

SLOWbus Devices

Serial Ports and Timers

The CPU card has four serial RS-232 ports and two system timers. Two of the ports are reserved for the keyboard and mouse. The other two ports are for general use. A pair of Motorola 68681 DUART chips, which are also used on the JCP and the RDC, are used for this application. These chips are able to interrupt the R3000.

The serial port connectors are on the ESV Workstation system's bulkhead connector panel. The CPU card connects to the bulkhead I/O panel through a 50-pin ribbon connector, P4, on the CPU card edge. If an RDC is used with the system, because either there are too many peripherals or the head is far from the controller, the RDC cable connects to the upper left of the DB-25 connectors on the bulkhead and the special keyboard and mouse connectors are unused.

Boot EPROM

The boot EPROM is a 32-bit wide bank of read-only memory which contains the system boot code, a low-level debugger, and some power-on diagnostics. Much of this source code comes from the MIPS SPP (System Programmers Package), and is altered to work with this particular card.

Time-of-day Clock, Non-volatile RAM

There is a time-of-day/date chip, the Thomson/Mostek MK48T12-25 ZEROPOWER TIMEKEEPER RAM, with battery backup to provide the correct time and date after every system boot. Along with this TOD clock, there is a 2Kbyte nonvolatile RAM that store installation-specific parameters.

Diagnostic LEDs

An 8-bit register is provided which drives a row of LEDs lights on the I/O edge of the CPU card.

Starting at Bit 0

The LEDs display vital confidence-test information during the reset sequence. During runtime they display the state or loading of the processor.

The following is a list of test phases and their corresponding LED values. This should be helpful in trying to figure out what is wrong with a system that hangs on powerup. By knowing what the LED pattern was when the system hung or a pattern that flashes repeatedly you can find the test that failed.

Non-visible Tests

	Hex value	Ponmask
• Duart Test	03	2
• Video Test	0A	8
• Gse Test	41	9
• Pixel Processor Test	44	18

Visible Tests

• Cache Test #1	05	1
• Cache Test #2	06	1
• Data Cache MATS+ Test	07	1
• Instruction Cache MATS+ Test	09	1
• Data Cache Block Refill Test	19	N/A
• Instruction Cache Block Refill Test	1A	N/A
• Write Buffer Test	0B	19
• Memory Test	0C	11
• TLB Test	0D	16
• Exception Test	0E	5
• Instruction Streaming Test	4D	N/A
• EDC Test	42	6
• Battery Check Test	0F	12
• NVRAM Test	0F	12
• Timer Test	11	15
• Time-of-Day Clock Test	12	17
• Duart Port Tests	13,14,15	3
• FP Test #1	16	7
• FP Test #2	17	7

-
- | | | |
|-----------------------------|----|----|
| • Lance Slave Register Test | 4A | 10 |
| • Lance Master Test | 4B | 10 |

VBIC

The VBIC is a VME Bus Interface Controller chip from SBE, Inc. It is a 144-Pin gate array that provides VME Master functionality, along with VME Bus Arbiter function, VME Bus Interrupt Handler, VME Bus Time-out Counter, and other functions.

YAM

The YAM (Yet Another Maintenance Register) controls individual resets for the different CPU card subsystems. It also contains the interrupt from the CPU to the DSP farm, and the bit which allows the CPU to clear the interrupt from the DSP farm. On power-up or push-button reset, the register is cleared to all zeros. The YAM is accessed by the CPU over the SLOWbus at byte 1 (see CPU Memory Map).

Memory Maintenance Register

The main memory maintenance register controls the overall operation of the ESV Workstation CPU memory subsystem. It provides both configuration control and diagnostic utility control. The register is 16 bits wide and resides on bytes 0 and 1 (bits 31-16) of the SLOWbus.

Interrupts

Introduction

Interrupts are one of the several types of exceptions that are handled by the R3000. The R3000's exception handling system is designed to efficiently handle various machine exceptions, including translation lookaside buffer misses, arithmetic overflows, I/O interrupts, system calls, breakpoints, reset, and coprocessor unusable conditions.

Interrupts in the R3000

The R3000 supports six general purpose hardware interrupts and two software-generated interrupts.

Since the R3000 only has six hardware interrupt lines, some of the interrupts must share a common line. In such a case, the R3000 must use some sort of polling to distinguish which device is requesting service. The interrupts are assigned to the six hardware interrupt lines as follows:

-
- *SYNCINT[0]/*FPUINT - Interrupt from the R3010 floating point coprocessor.
 - *SYNCINT[1]/*DBERRINT - Fatal double bit error interrupt from the memory controller.
 - *SYNCINT[2] - *ETHERINT
 - *SYNCINT[3] /*SCSIINT - Interrupt from the NCR53C90 SCSI controller chip
 - *SYNCINT[4] - *VMEINT - This includes all of the VME interrupts as well as the *COPFIFOEF, *COPFIFOHF, *COPFIOFF, *ALMOST_FULL, and *DSPINT interrupts which are routed through the VBIC chip. Polling is necessary to identify the actual interrupter.
 - *SYNCINT[5] - *DUARTINT or *SBERRINT - This includes all interrupts associated with the four serial ports and the two timers. The write buffer bus error interrupt is also routed through the DUARTs. The R3000 must first poll the DUART chips to see if they require service. If they have not caused the interrupt, the R3000 then assumes that a single bit error has occurred.

Memory Maps

Figure 4-5 shows how the ESV Workstation CPU virtual address space correlates with the Mbus physical address space and VMEbus physical address space. Figure 4 - 6 shows the Mbus address space map. With the use of these two diagrams an address can be determined for all of the Mbus devices. With the address of that device, the contents of maintenance registers can be read or written, NVRAM read or written, etc.

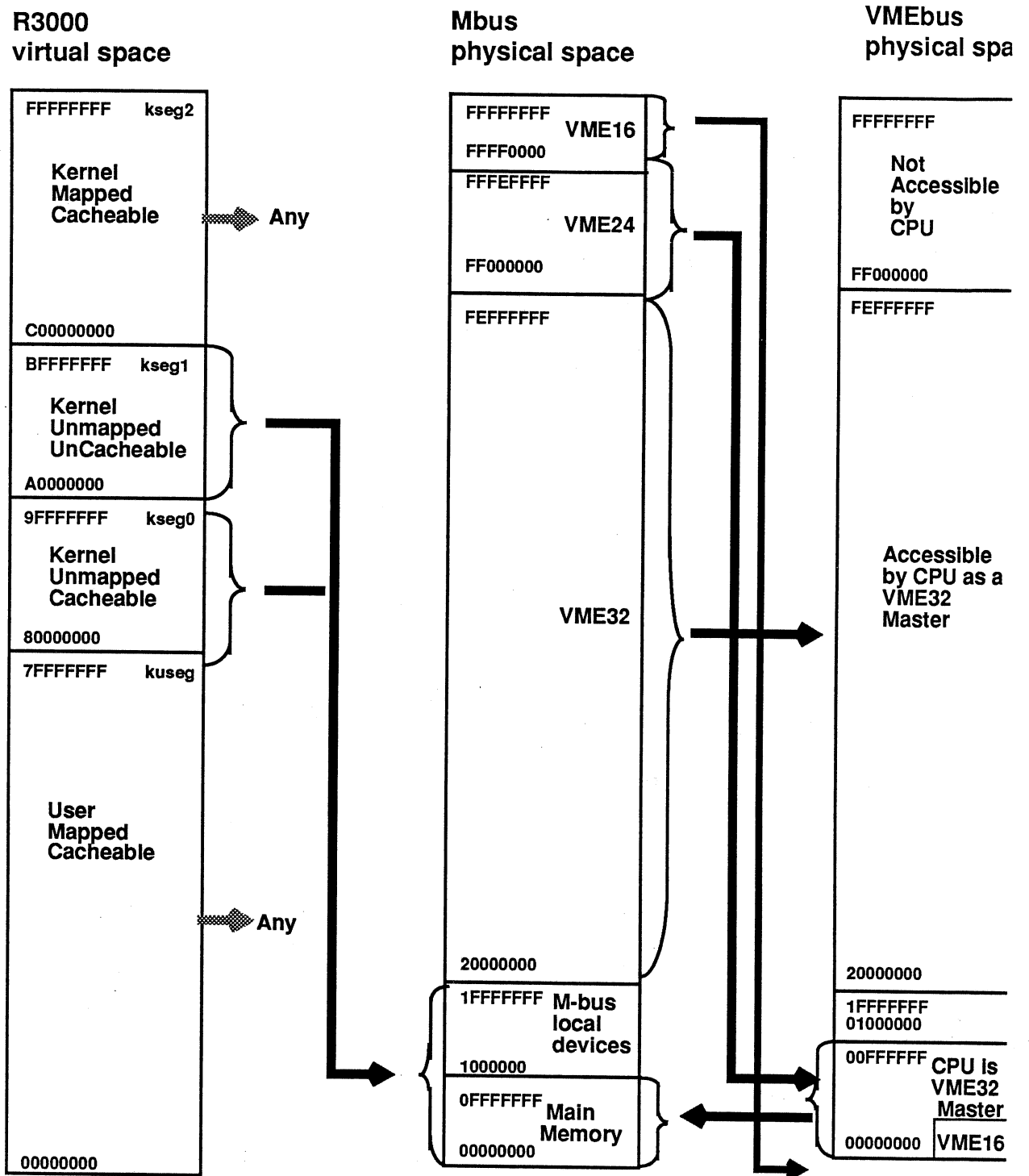


Figure 4 - 5 ESV Workstation CPU Address Space Map

		Bytes In Word Used (In white)			
		31	0		
1F000000	UNUSED EPROM duplicate				
1FC3FFFF	BOOT EPROM				
1FC00000	256K bytes				
1F000000	UNUSED EPROM duplicate				
1E000000	UNUSED				
1E000000	SCSI duplicate				
1E000004	Ethernet LANCE Chip				
1D000000	UNUSED				
1D000040	SCSI duplicate				
1D000044	SCSI DMA Address Register				
1D000040	SCSI PAL Control				
1D00003F	SCSI Controller Chip				
1D000000					
1C000000	UNUSED NVRAM duplicate				
1C001FFF	NVRAM, TOD Clock				
1C000000					
1B000000	UNUSED VBC duplicate				
1B00003F	DUART 1 Registers				
1B000000					
1A000000	UNUSED VBC duplicate				
1A00003F	DUART 0 Registers				
1A000000					
19000000	UNUSED VBC duplicate				
19000200	VME Interrupt Level/Ack				
190001FF	VIBC Registers				
19000000					
18000000	UNUSED Memory Maint duplicate				
18000000	Memory Maint Register				
17000000	UNUSED YAM Duplicate				
17000000	YAM Register				
17000000	UNUSED LED Duplicate				
16000000	LED Register				
15000000	UNUSED				
14xxxxxx	SetWBMaintMode				
13xxxxxx	EnWBdat				
12xxxxxx	EnWBAdr				
11xxxxxx	ClrWBErrInt				
10xxxxxx	UNUSED				

0	1	2	3	(R)
0	1	2	3	(R/W)
0	1	2	3	(W)
0	1	2	3	(W)
0	1	2	3	(R/W)
0	1	2	3	(R/W)
0	1	2	3	(R/W)
0	1	2	3	(R)
0	1	2	3	(R)
0	1	2	3	(R/W)
0	1	2	3	(R/W)
0	1	2	3	(R/W)
0	1	2	3	(R)

DATALESS
DATALESS
DATALESS
DATALESS

Figure 4 - 6 M-Bus Local Devices Address Map

5. Graphics System Executive (GSE) Card

The ESV Workstation GSE card is the hub of the graphics subsystems and functional units distributed throughout the ESV Workstation card set. Graphics data and commands defined in the CPU card and transformed in the several DSP cards are funneled through the GSE card before being handed off to the pixel processor array. Several functional units which could not be located elsewhere, or which might have been located elsewhere but could not for want of space, have been assigned to the GSE card.

For documentation purposes, the GSE card has been divided into four subsystems. These are the following:

- The DSP "farm"
- The Graphics Bus (GBUS)
- The Laundry Chute
- The Pixel Processor Array Loader (PPA Loader)

Figure 5-1 shows a block diagram of the GSE card.

All I/O connector pins on the GSE card are grouped under three different bus systems (G, F, and P buses) which service the card.

DSP Subsystem

The GSE card includes an array of four DSP units. Three other functional units support the DSP array:

- Parallel I/O (PIO) port,
- Direct Memory Access (DMA) controller,
- Board BUS (BBUS).

These units are fundamentally DSP card topics as they are duplicated on each DSP card. The four DSP units on the GSE card are supported in such a way that an ESV Workstation can carry on all graphics functions, albeit at a reduced performance, without any DSP cards present.

GBUS Subsystem

The graphics bus is an asynchronous bus used for communication among the graphics input FIFO, the VME Bus, sync memory, the DSP farm and the FBUS. Signals on the GBUS include 32 bits of data bus, 22 bits of address and a number of control signals. Masters on the GBUS include the DSPs and the VME bus. The graphics input fifo, sync memory and the FBUS interface are slave devices on the GBUS.

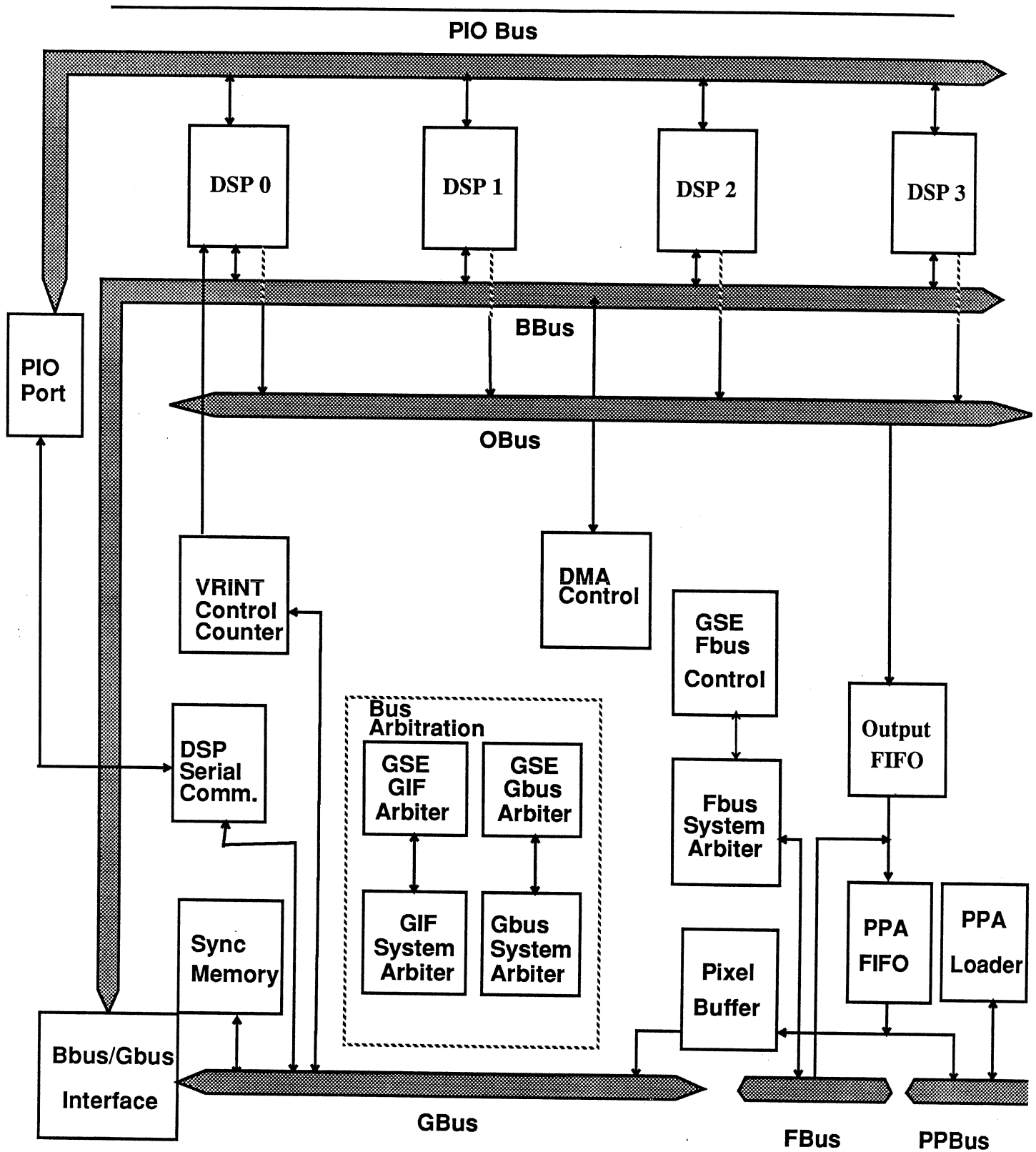


Figure 5 - 1 GSE Block Diagram

GBUS Arbiter

The GBUS arbiter controls the arbitration on the GBUS. Masters on the GBUS request the bus by asserting *GBUSREQ. In some cases, the master may hold on to the bus for multiple bus cycles. A DSP may hold onto the bus to get exclusive access to sync memory, or to do a block transfer to sync memory, or when a DMA transfer from the graphics input FIFO has begun. To facilitate read/modify/write in sync memory, a master may hold on to the GBUS. This allows communication between DSPs through software semaphores. Also, to accommodate a rapid transfer of state information, a DSP may hold on to the GBUS and initiate a fast block transfer between sync memory and the DSP's local memory. Graphics information from the graphics input FIFO is usually transferred by DMA. The master initiating the DMA transfer from the GIF to a DSP's local memory does not release the bus until the DMA goes to completion. In the event that the GIF goes empty during the DMA transfer, the master releases the bus and requests the bus again after the *GIFEMPTY signal is negated.

SYNC Memory

Sync Memory stores graphics state information for DSP intercommunication and for CPU to DSP communication, and it functions as overflow program and data memory for the DSPs.

The sync memory is located on the GSE board, together with the Pixel Processor Loader, the GBUS arbiter and the hardware semaphore logic. A maximum of 2 Mbytes sync memory can be supported in the ESV Workstation product. The current amount of sync memory is 256 Kbytes. There is a single port into sync memory from the GBUS. DSPs access sync memory across the GBUS, and the CPU accesses sync memory via the VME bus to GBUS interface.

Read and write cycles from a DSP occur at the rate of approximately 700ns per 32 bit word. Read/modify/write cycles occur at the rate of 1.5us per 32 bit word. A master on the GBUS gets exclusive access to sync memory by maintaining control of the GBUS. The DSPs use this feature to implement fast block access to sync memory. In this fast block transfer mode, after the initial overhead, data is read at the rate of 300ns per 32 bit word. The fast block access is entirely controlled by the DSPs.

The fast block access overhead includes obtaining and locking the GBUS, modifying and restoring the DSPs PCW register, if necessary, and release of the GBUS. The address space on the GBUS is divided into a "sticky" region and a "non-sticky" region. All the hardware resources on the bus can be accessed in the sticky and non-sticky address space. The difference is that when a GBUS access is done in the "sticky" region, the bus master does not negate its GBUS request signal which results in the master having exclusive use of the bus until he accesses in the "non-sticky" address region of the GBUS.

VRINT Control and Counter

The task of the vertical retrace interrupt controller is to interrupt DSP 0 when a vertical retrace is occurring.

Vertical retrace is the time it takes for the monitor to move from the bottom horizontal scan line to the top line and begin displaying the next frame. Since the video is blanked during vertical retrace, this time is used to update registers and tables within the video pipeline that could have an effect on the display if updated during an active video scan.

The controller also keeps a count that indicates how much time is left in the current vertical retrace cycle.

When the video system asserts the vertical retrace interrupt signal (VRINT), the controller first obtains control of the graphics input FIFO semaphore to reduce GBUS traffic. After getting control of the GIF semaphore, the controller sends an interrupt to the DSP. The DSP starts the necessary processing for a vertical retrace cycle. After the first read of the vertical retrace counter, the controller negates the interrupt. The DSP can check the vertical retrace counter to find out how much time is left for additional vertical retrace processing. After completing vertical retrace processing, the DSP must write to the vertical retrace counter to reset the counter.

Laundry Chute Subsystem

The laundry chute is a "collector system" which channels PPA packets from up to 44 DSPs into the PPA FIFO. It includes many arbiters, busses, FIFOs, and finite state machines which are located on up to six different cards. Figure 5-2 shows a block diagram of the laundry chute subsystem (coupled with the PPA Loader subsystem).

Each DSP unit obtains graphic command primitives from the graphics input FIFO as described in the CPU card chapter. From these primitives the DSP units generate PPA packets which are queued in DSP local memory. All DSP units in an ESV Workstation may work together in parallel in the generation of packets. The ESV Workstation pixel processor receives all instructions for drawing, BLTing, frame buffer reading, etc., in the form of PPA packets.

A packet is a group of five to thirteen 32-bit words. The first word of a PPA packet is the command word which is interpreted by both the PPA loader and the PPA. After a DSP has queued a block of PPA packets, it asks for permission to write the block into the output FIFO over the OBUS. A block is some whole number of packets where the sum of all 32-bit words in all the packets totals near to, but not exceeding, 256 words.

The GSE card has an OBUS, an output FIFO and an output FIFO arbiter which serve the four DSP units on the card. Similarly, on each DSP card there is an OBUS, output FIFO and output FIFO arbiter which serve eight DSP units. The output FIFO arbiter resolves DSP contentions for the OBUS. After a DSP unit has transferred its block into the output FIFO, it relinquishes the OBUS and begins processing another primitive.

Packets written into the several GSE and DSP card output FIFOs must be collected and deposited into the PPA FIFO located on the GSE card. This is the job of the FBUS and FBUS controllers. There is one FBUS. It is located on the backplane, and it connects one or more DSP cards with the GSE card. The GSE card and each DSP card have separate FBUS controllers which monitor their respective output FIFOs. The controllers contend for the FBUS and permission to write packets into the PPA FIFO. The FBUS arbiter, located on the GSE card, resolves contentions for the FBUS.

Output FIFO Arbiter

The output FIFO arbiter is a PAL state machine which runs on its own 40 ns clock asynchronous to the DSP units. It arbitrates access to the output FIFO over the OBUS. Each requestor has equal priority and arbitration is round-robin type. The processes of being granted the OBUS has been termed obtaining the laundry chute semaphore.

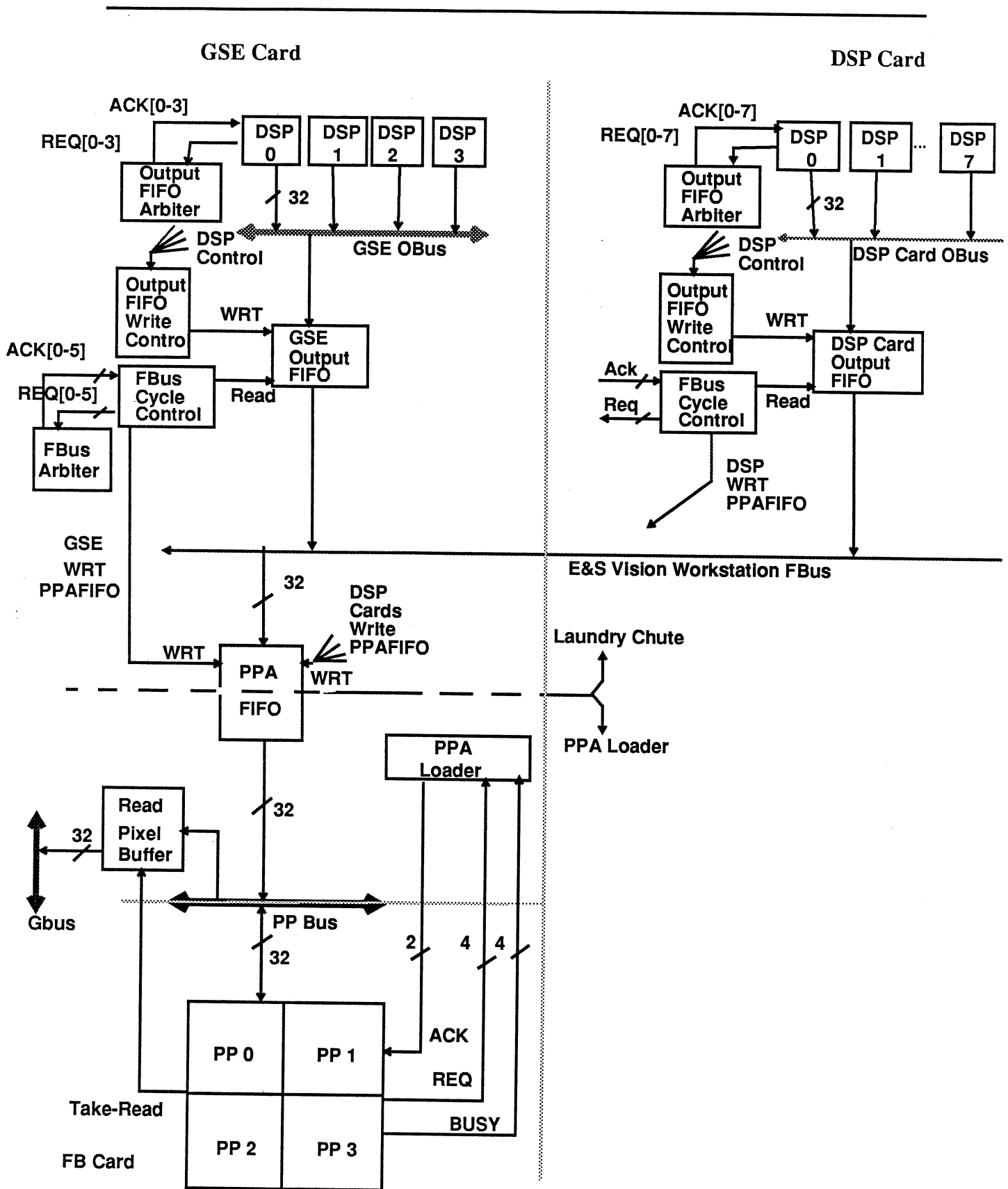


Figure 5 - 2 Laundry Chute and PPA Loader Subsystems

OBUS

The OBUS is a 32-bit bus which is the path between a DSP unit's output FIFO data latch and the output FIFO. All of the DSP units on a card share an OBUS which is entirely local to that card. Therefore, there is a GSE OBUS, a DSP card 0 OBUS, a DSP card 1 OBUS, etc., with six different OBUSses in a fully loaded ESV Workstation.

Output FIFO

The output FIFO is 32-bits wide and 512 words deep. It is written under DSP control and read under FBUS control. Reading and writing are asynchronous events. The pointers of the output FIFO are initialized with a GBUS reset.

FBUS Controller

An FBUS controller is located on the GSE card and on each DSP card. The jobs of an FBUS controller are:

- to watch for output FIFO data,
- to request the system FBUS when data is present, and
- to write the data into the PPA FIFO when granted the FBUS.

PPA Loader Subsystem

The PPA Loader Subsystem, located entirely on the GSE card, performs the following duties:

- Monitors PPA FIFO status and PPA requests to transfer packets from the laundry chute to the pixel processors.
- Buffers 32-bit pixel data read from the frame buffer by the pixel processors.
- Transfers read-pixel data to the GBUS.
- In laundry chute diagnostic mode, it transfers 32-bit data from the PPA FIFO output to the GBUS.

Figure 5 -3 is a block diagram of the PPA loader. The four ESV Workstation pixel processors are located on two frame buffer cards. Packet data to the PPA and pixel data from the PPA pass through a 32-bit, bi-directional port on the pixel processors which is situated on the PPBUS. The instruction loading path begins at the output of the PPA FIFO and ends at the PPBUS with two regulating registers situated in between. The read-pixel data path begins at the PPBUS and ends at the GBUS with one register and the read pixel buffer in between. In laundry chute diagnostic mode, these two paths are combined to route data from the laundry chute onto the GBUS while the pixel processors are idle.

PPA FIFO

The PPA FIFO is 32-bits wide and 512 words deep. The PPA FIFO is written under FBUS control and read under PPA loader control. Reading and writing are asynchronous events. The pointers of the PPA FIFO are initialized with a GBU

PPA PACKET CONTROLLER

The PPA packet controller has the job of transferring packets from the PPA FIFO to the PPA when they are in a state of mutual readiness. The packet controller knows that a packet is ready by monitoring the PACKETFLAG.

The packet controller reads and registers the first word of a packet when PACKETFLAG asserts. The first word is the command word. It contains information which the packet controller must interpret before proceeding. How the packet transfer proceeds from this point depends on the state of the two "loader action" bits of the packet command word and the state of the eight pixel processor signals NP-REQ[0-3] and BUSY[0-3].

Bits 23 and 22 of the packet command word are the loader action bits which have the following encoded meanings:

<u>23</u>	<u>22</u>	
0	0	Wait for NP-REQ[0-3] and wait for BUSY[0-3] all to be in LOW state then transfer to all PPs.
0	1	Wait for NP-REQ[0-3] all to be LOW then transfer to all PPs.
1	0	Wait for NP-REQ[0,1] to be LOW then transfer to even PPs only.
1	1	Wait for NP-REQ[2,3] to be LOW then transfer to odd PPs only.

Pixel processors 0 and 1 are the even processors which write the pixels of the even scan lines of the display. Pixel processors 2 and 3 are the odd processors which write the pixels of the odd scan lines of the display. NP-REQ is the signal from the pixel processor requesting a new packet from the PPA loader. BUSY is the signal from the pixel processor indicating that it is busy and therefore cannot change its internal state. Whether the packet goes to even, odd, or all pixel processors depends on how the loader acknowledges the PPA. The even pixel processors take the packet data if the loader asserts NP-ACK[0] (LOW) on the packet transfer. The odd pixel processors take the packet data if the loader asserts NP-ACK[1] (LOW) on the packet transfer.

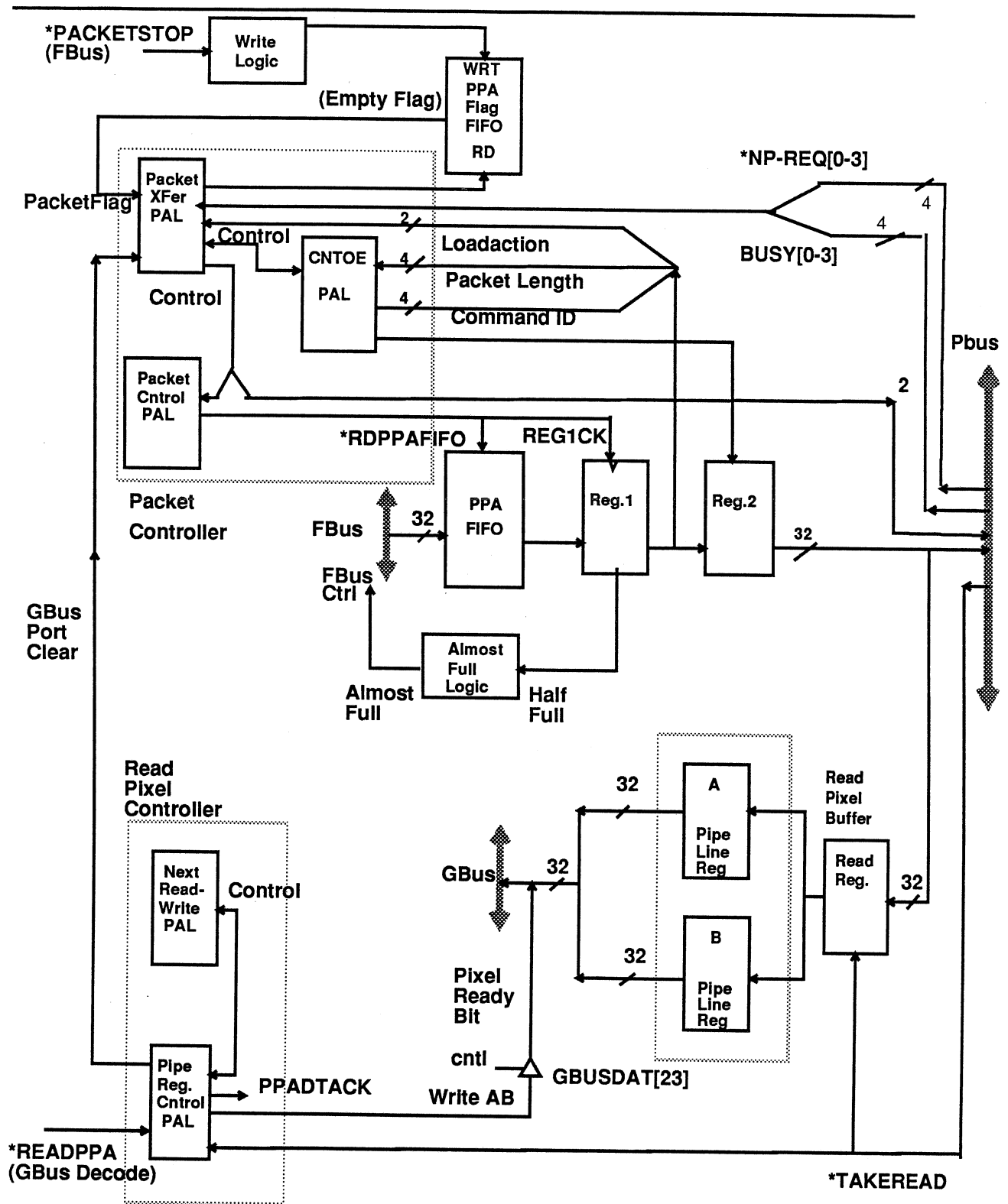


Figure 5 - 3 PPA Loader Block Diagram

READ Pixel Controller

In response to each read pixel packet given to the PPA, the PPA loader must be ready to receive four 32-bit pixels, one pixel each 70 ns. The PPA loader has the PPBUS unless the PPA must perform a read pixel cycle or a BITBLT cycle in which case the PPA has the PPBUS. If the packet being handed the PPA is read pixel, BITBLT, or TEXTBLT, the PPA loader releases the PPBUS one state after the PPA clocks in the last word of the packet.

The PPA signifies the presence of pixel data on the PPBUS by asserting *TAKE-READ. For each of the four pixels *TAKE-READ is asserted one state before the data. The PPA loader looks for *TAKE-READ before each pixel, but it expects that all four will arrive uninterrupted in 280 ns. BITBLTs between pixel processors are executed similarly while the PPA loader is tri-state on the PPBUS except that *TAKE-READ is not asserted.

The PPA loader is prepared to double-buffer sets of four pixels. This expedites the reading of the frame buffer by allowing the PPA to output pixels while a GBUS master reads from the previously loaded buffer of four pixels. Read pixel data is at GBUS address A60000.

Laundry Chute Diagnostic Mode

The laundry chute may be put in diagnostic mode by setting bit 13 of the DSP maintenance register. The purpose of diagnostic mode is to allow data sent down the laundry chute from any DSP to be read back via the GBUS rather than sent to the PPA. Any GBUS master can read diagnostic words from GBUS location A60000.

6. Frame Buffer (FB) Card

Overview

The back end of the ESV workstation includes two identical FB cards (FB0 and FB1), the Graphics Executive (GSE) card, and a video card. Each FB card contains the following:

- Two pixel processor chips
- One-half of the total frame buffer memory

Pixel processors render graphics primitives into frame buffer memory. The GSE (also known as the loader card) feeds the pixel processors via the pixel processor bus. Pixel processors also use this bus for intercommunication. Pixels in frame buffer memory are scanned out to the video card. The first FB card, FB0, drives even-address pixels across the V0 Bus. FB1, the second FB card, drives odd-address pixels across the V1 Bus.

Subsystem Descriptions

The Pixel Processors

The pixel processors are custom VLSI chips. The pixel processors provide the following functionality:

Note: All present ESV Workstations have a fully-populated 88-plane frame buffer.

- 1280 by 1K screen
- Anti-aliased dots and lines
- Depth-cueing and linear shading
- 24-bit fixed point generalized z-buffering
- Overlay/underlay and window clipping
- Write masking
- Patterned lines and tiles
- Frame buffer access from an external processor
- Block clear
- Valid clear
- BitBLT
- Line drawing pre-computing

-
- Frame buffer VRAM control
 - DRAM refresh
 - Display refresh

Frame Buffer Memory

Figure 6-1 shows the organization of the frame buffer memory.

Total frame buffer memory contains 1536 x 1024 pixels and is up to 88 planes deep. This includes two color buffers of 24 bits each, 8 window/overlay/underlay bits, 8 valid bits, and 24 z-buffer bits. The second color buffer, the valid bits, and the z-buffer bits are optional.

One-half of the memory is located on each FB. Each pixel processor accesses one-quarter of the total memory. The block of memory accessible by a pixel processor is divided into 3 banks. The pixels of a row are interleaved between two blocks, one on each FB and between the three banks in each block. Even rows are found in the block accessed by pixel processor 0 on each FB and odd rows in the block accessed by pixel processor 1. The location of the first 12 rows and the first 12 pixels of rows 0, 4, 8, 1020 are shown in the figure.

The basic memory read-modify-write time is 490 nsec. Each of the 4 pixel processors does two read-modify-writes in parallel. The effective read-modify-write time is $490 / (4 \times 2) = 61.25$ nsec/pixel. The basic write cycle time is 245 nsec. Each of the 4 pixel processors "blast writes" six banks in parallel, leading to an effective blast write time of $245 / (4 \times 6) = 10.2$ nsec. Single pixel access time is approximately 735 nsec. These times are based on the pixel processor clock period of 35 nsec.

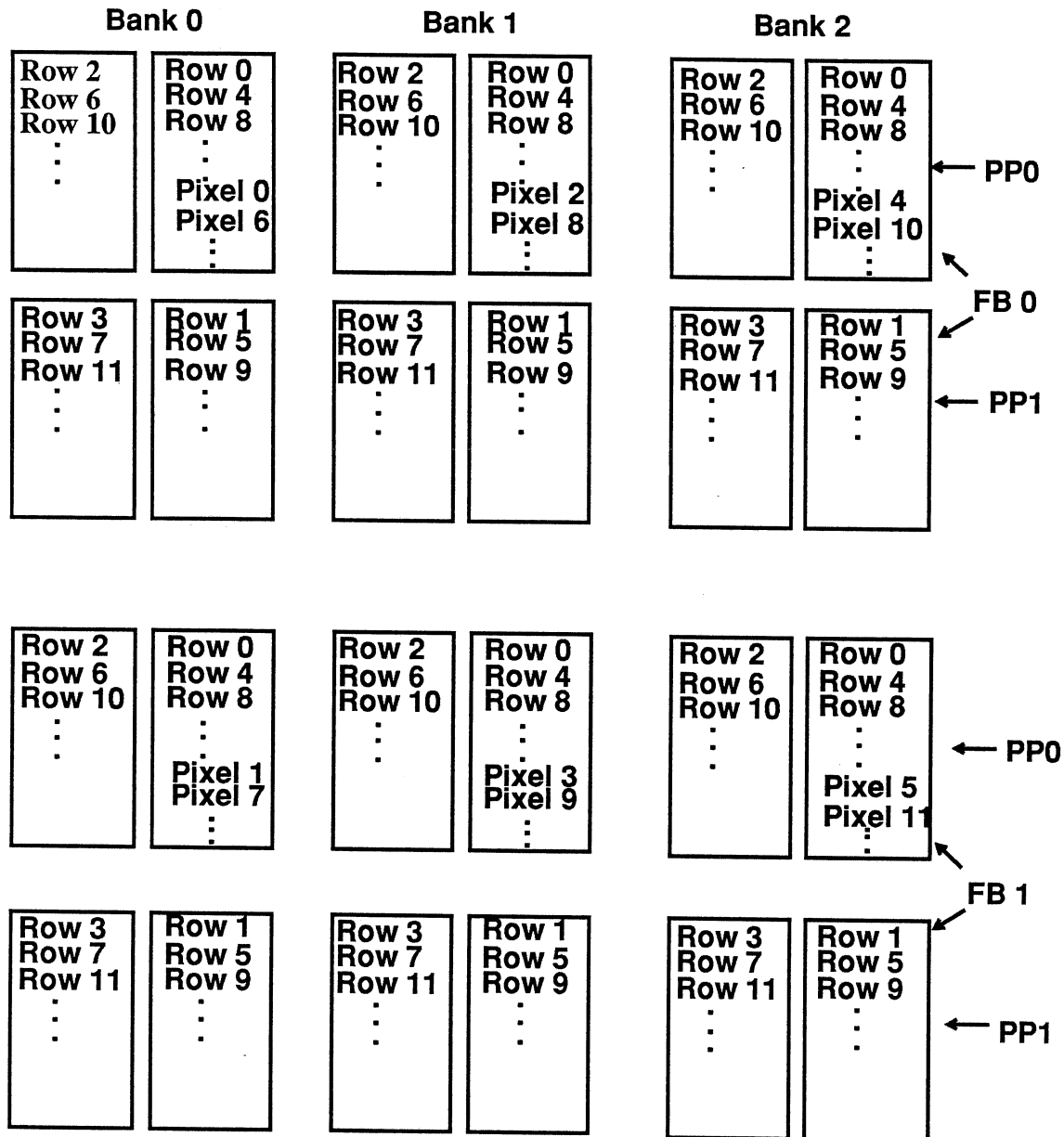


Figure 6 - 1 Frame Buffer Memory Organization

7. Video Card

Overview

The video card consists of the following subsystems:

- Video timing control
- Pixel output pipeline
- Gbus interface
- Frame buffer interface

The video card takes pixel, window, and valid data from the frame buffer, reformats it according to the contents of the window lookup table (WLUT), color lookup tables (CLUTs), window data, and valid data, then converts the reformatted data to three analog video signals which drive the monitor.

The video card has an interface to the frame buffer (FB) cards, the pixel processor array loader on the GSE card, and an interface to the Gbus. The first part of the entire pixel output pipeline is located on the frame buffer cards. Figure 7-1, "System Connect Block Diagram," shows how the video card interacts with the rest of the system. Figure 7-2a and 2b, "Video Card Block Diagram," is a block diagram of the various components of the card.

Subsystem Descriptions

Video Timing Control

The video timing control subsystem provides clock and timing signals for the pixel output pipeline, and timing signals for external video devices such as monitors and stereo shutters. The cursor generation hardware is considered part of the video timing control.

Pixel Output Pipeline

The pixel output pipeline reformats data from the frame buffer to provide the many different modes of displaying the frame buffer data and presents the reformatted data to the combination CLUTs and digital-to-analog converters (RAMDACs) which drive the monitor and other video peripherals. Figure 7-3, "Video Path with RAMDAC," is a conceptual picture of the pixel output pipeline for a single pixel.

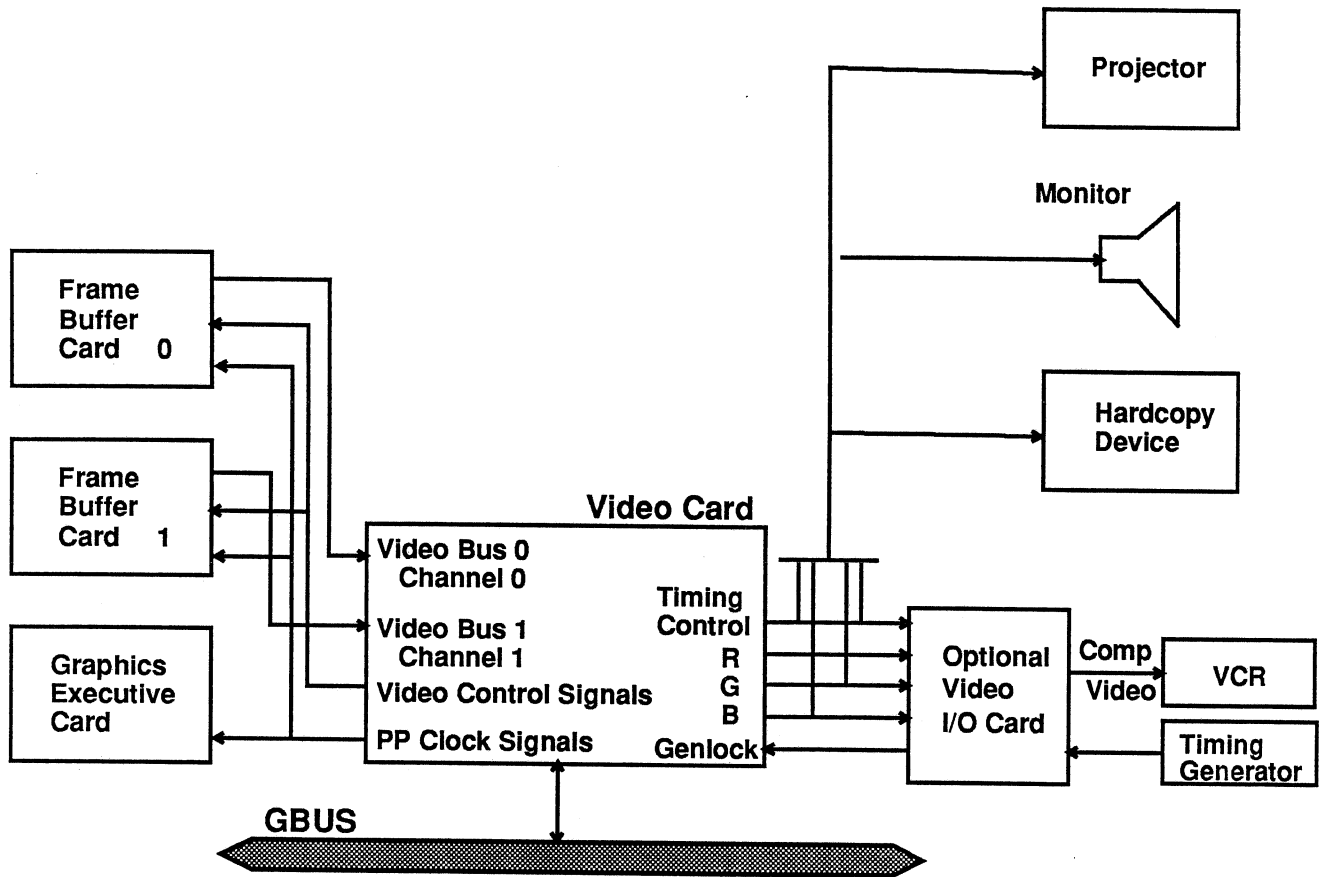


Figure 7 - 1 System Connect Block Diagram

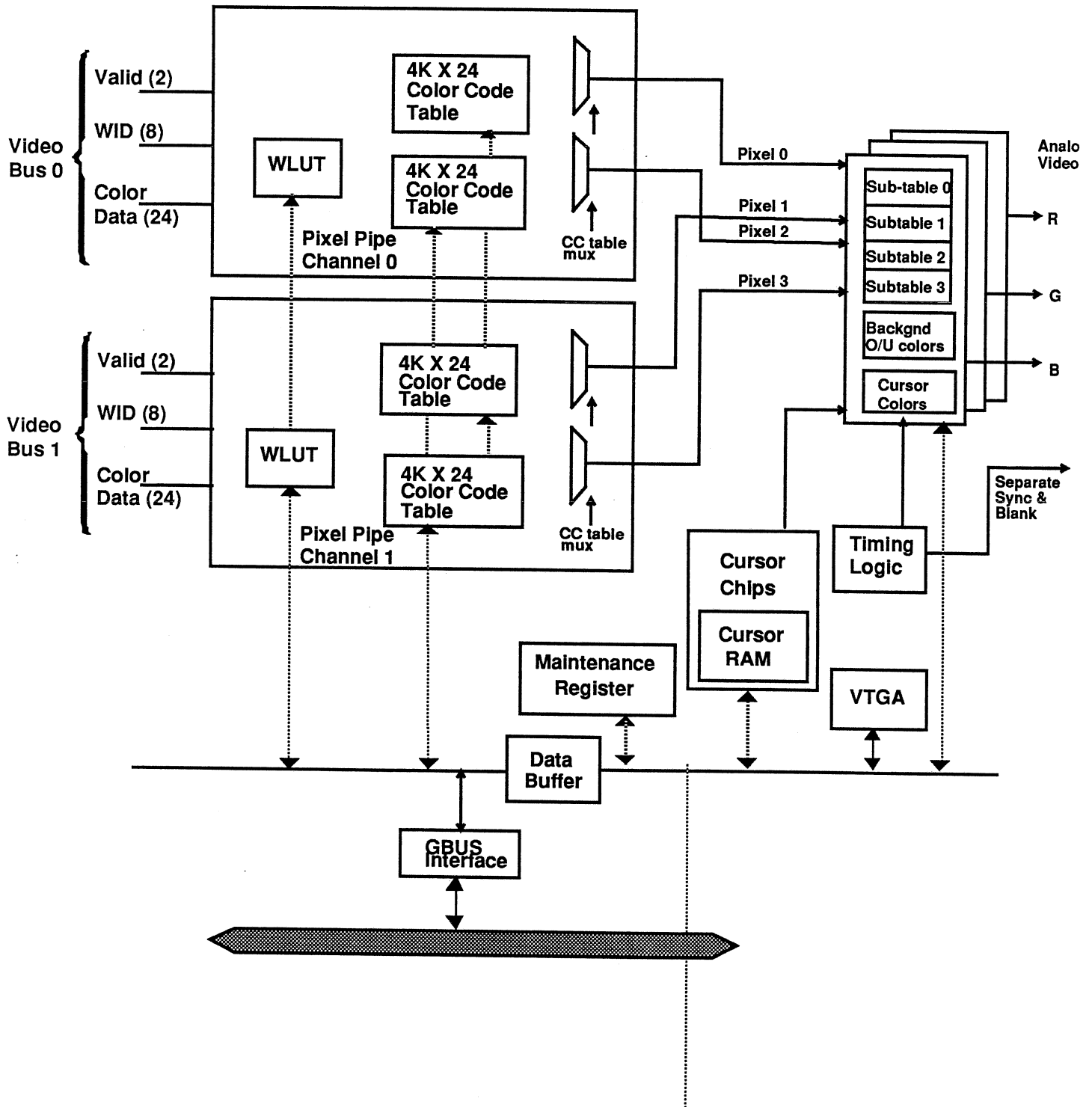


Figure 7 - 2a Video Card Block Diagram

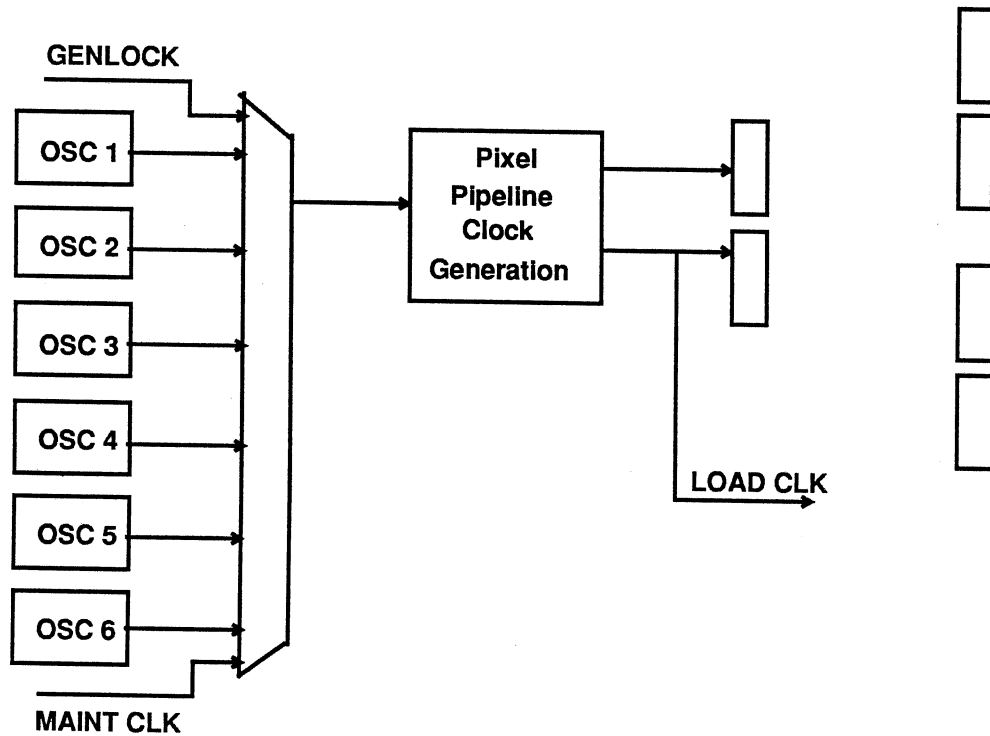
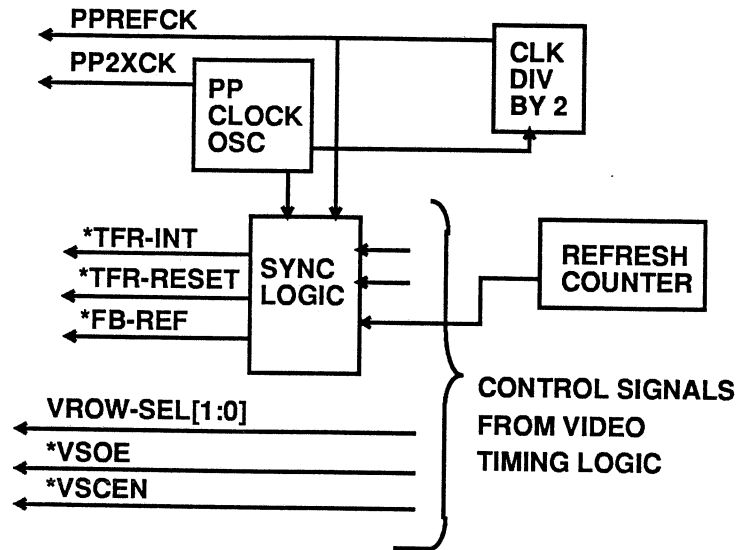


Figure 7 - 2b Video Card Block Diagram

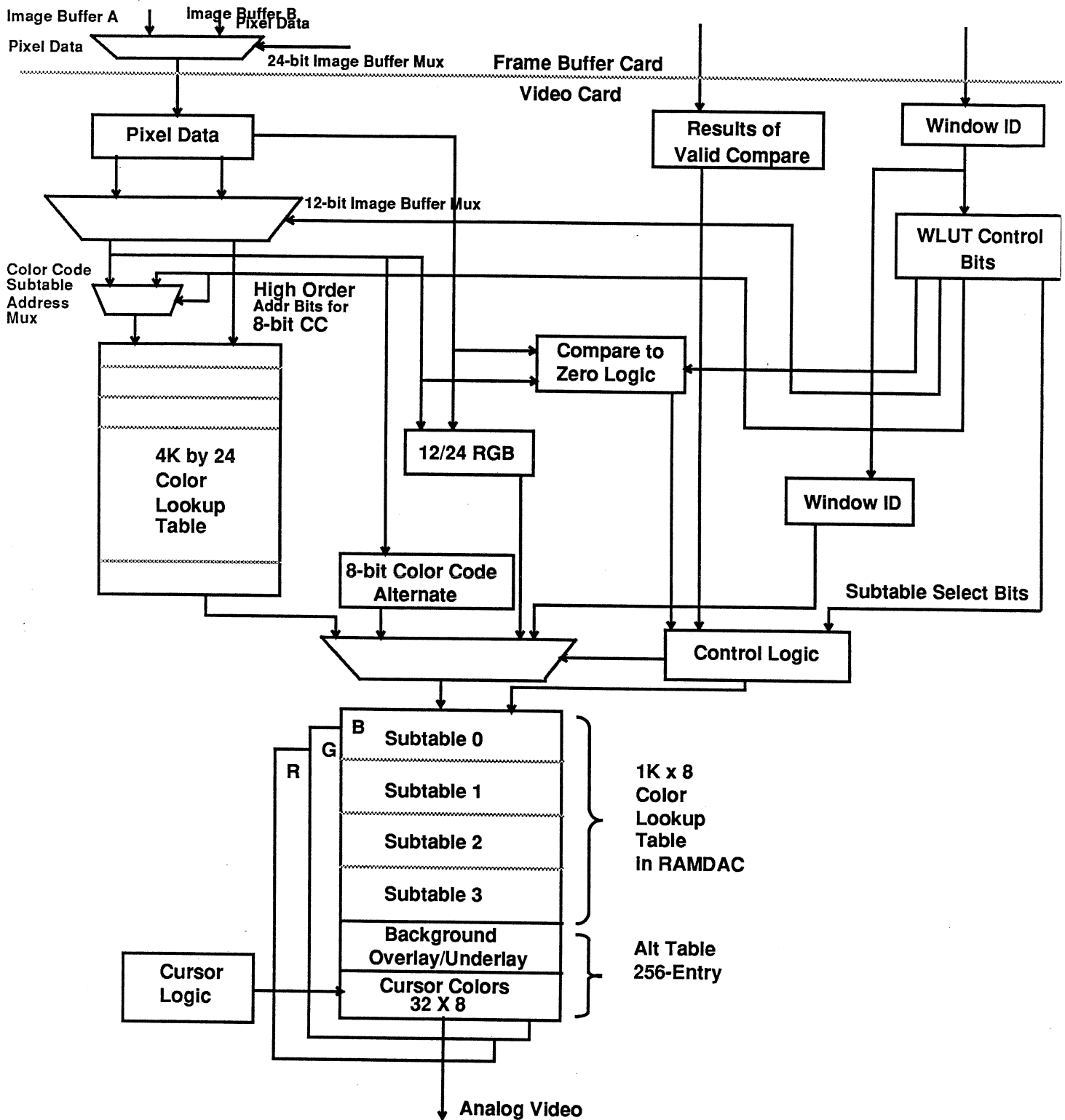
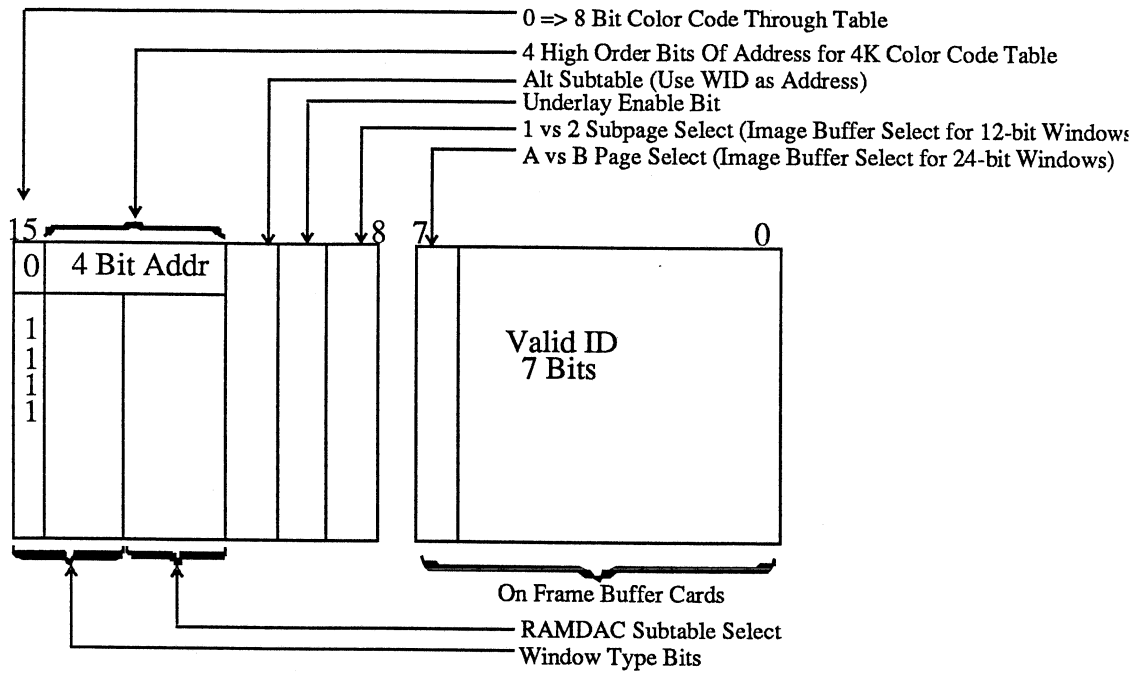


Figure 7 - 3 Video Path With RAMDAC



Selecting 24-Bit Image Buffers (Windows)

		WLUT Bit 8	
		WLUT Bit 7	
Page A	1	1	
Page B	1	0	

Window Type

0	x	x	8-Bit Color-coded in 4K CLUT
1	0	0	12-Bit Color-coded in 4K CLUT
1	0	1	8-Bit Alt Color-coded in RAMDAC
1	1	0	12-Bit RGB
1	1	1	24-Bit RGB

Selecting 12-Bit Image Buffers (Windows)

		WLUT Bit 8	
		WLUT Bit 7	
Subpage A1	1	1	
Subpage A2	0	1	
Subpage B1	1	0	
Subpage B2	0	0	

Subtable Select

0	0	Subtable 0
0	1	Subtable 1
1	0	Subtable 2
1	1	Subtable 3

Figure 7-4 Window lookup table

There are ten different display modes. These display modes are as follows:

- 8-bit color coded from LSBs using 4K color table
- 8-bit color coded from MSBs using 4K color table
- 8-bit color coded from LSBs using RAMDAC,
- 8-bit color coded from MSBs using RAMDAC
- 12-bit color coded from LSBs
- 12-bit color coded from MSBs
- 12-bit RGB from LSBs
- 12-bit RGB from MSBs
- 24-four-bit RGB
- Background from window ID

The organization of data storage for these window types is shown in Figure 7- 5, "Data Organization in Frame Buffer."

Pixel Pipeline Data Flow

The pixel pipeline consists of multiple paths which can be followed by the data on its way from the frame buffer to the RAMDACs. There is a different path for each of the display modes.

The Window Lookup Table

The function of the pixel output pipeline is controlled by the contents of the WLUT. Each location in the WLUT contains eight bits of information to control the lower part of the pixel output pipeline.

Frame Buffer Interface

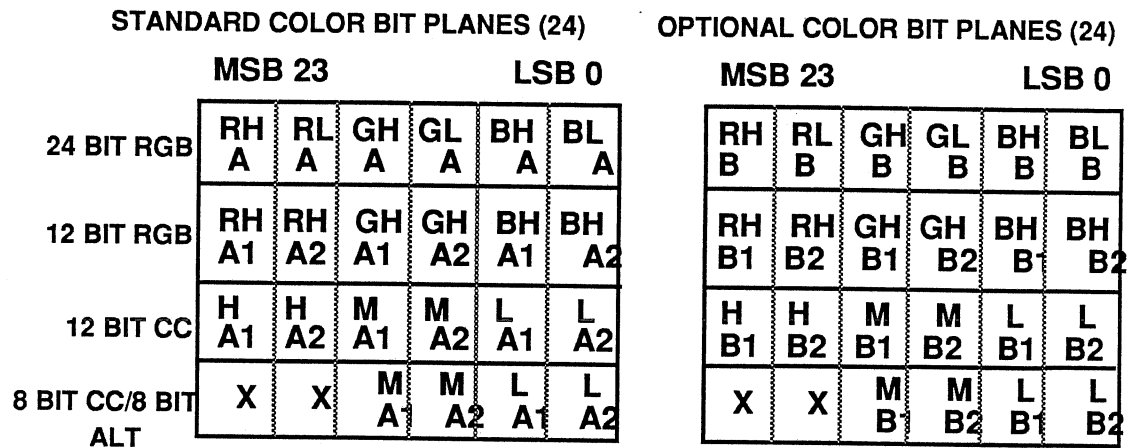
The FB interface subsystem provides clock and control signals to the FB cards and the pixel processor array loader. A 57.1429 MHz TTL clock and a reference clock are provided.

GBUS Interface

The GBUS interface subsystem allows the Gbus to read and write maintenance registers on the video card, CLUTs and WLUTs, and video timing control registers. This read and write capability is used for initialization, vertical retrace updates, and diagnostic purposes.

Registers

The four items listed below pertain to the video card's registers.



COLOR CODE VIEWED AS ADDRESS TO TABLE

11	7	0
H	M	L

- A => PAGE A (FULL 24 BITS)
- B => PAGE B (FULL 24 BITS)
- A1 => SUBPAGE 1 OF PAGE A (12 BITS)
- A2 => SUBPAGE 2 OF PAGE A (12 BITS)
- RH => RED COMPONENT OF RGB COLOR - HIGH ORDER NIBBLE
- RL => RED COMPONENT OF RGB COLOR - LOW ORDER NIBBLE
- X => DON'T CARE
- H => HIGH ORDER NIBBLE OF 12-BIT COLOR CODE
- M => MIDDLE ORDER NIBBLE OF 12-BIT COLOR CODE / HIGH ORDER NIBBLE FOR 8-BIT COLOR CODE
- L => LOW ORDER NIBBLE OF 12-BIT OR 8-BIT COLOR CODE

Figure 7 - 5 Data Organization in FB

- 1) Writes to all registers except for the maintenance register and the video timing gate array should only be done during vertical retrace. This restriction is best accomplished by waiting for vertical blank not asserted, then waiting for vertical blank asserted, and then doing the writes. Following this procedure will ensure that the entire vertical retrace period is available for doing the writes, and that no anomalies will be visible on the screen. Failure to follow this procedure may result in anomalies being visible on the screen.
- 2) Writes to and reads from the WLUTs and CLUTs will only work properly if the bit *PADROE (bit 6) in the maintenance register is high. If the bit is low, the address source for those reads and writes will be the data in the pixel output pipeline.
- 3) Writes to any of the CLUTs will result in writes of those data to all four of the CLUTs.
- 4) Writes to any of the WLUTs will result in writes of those data to both WLUTs.

Color Lookup Tables

The following mapping applies to all CLUTs:

- GBUSDAT[23-16] maps to red
- GBUSDAT[15-8] maps to green
- GBUSDAT[7-0] maps to blue

Color Lookup Table 0 (CLUT0)

Address X'A80000' - X'A83FFC' R/W memory

Color Lookup Table 1 (CLUT1)

Address X'A84000' - X'A87FFC' R/W memory

Color Lookup Table 2 (CLUT2)

Address X'A88000' - X'A8BFFC' R/W memory

Color Lookup Table 3 (CLUT3)

Address X'A8C000' - X'A8FFFC' R/W memory

Window Lookup Tables

The following mapping information applies to both WLUTs:

- GBUSDAT[15-8] maps to WLUTDAT[15-8].

<u>Name</u>	<u>Bit</u>	<u>Explanation</u>
STEREO	23	Stereo left/right signal. Status bit indicates which eye view of a stereo image is currently being displayed. A logic high indicates that the left eye view is being displayed. A logic low indicates that the right eye view is being displayed.
EVENFIELD	22	Even field signal. Status bit indicates which field of an interlaced video format is currently being displayed. A logic high indicates that the even numbered scan lines are currently being displayed on the screen. Television numbers the scan lines beginning with one, so the odd numbered rows of the frame buffer are displayed when the EVENFIELD signal is a logic high.
VBLANK	21	Vertical blank. Status bit indicates that the display is in the vertical blanking interval when all tables and registers can be updated. The pixel pipeline diagnostics use this signal to determine when to expect pixel data to be valid in the pixel output pipeline.
C1VALID[1]	20	Channel 1 valid bit 1. Status bit indicates value of valid bit 1 from channel 1. Used for diagnostic purposes.
C1VALID[0]	19	Channel 1 valid bit 0. Status bit indicates value of valid bit 0 from channel 1. Used for diagnostic purposes.
C0VALID[1]	18	Channel 0 valid bit 1. Status bit indicates value of valid bit 1 from channel 0. Used for diagnostic purposes.
C0VALID[0]	17	Channel 0 valid bit 0. Status bit indicates value of valid bit 0 from channel 0. Used for diagnostic purposes.
OSCSEL[2]	16	Pixel clock oscillator select bit 2.
OSCSEL[1]	15	Pixel clock oscillator select bit 1.
OSCSEL[0]	14	Pixel clock oscillator select bit 0. OSCSEL[2-0] are used to select the pixel clock source. They map to the sources as follows:

Select: Frequency: Name, use.

000	107.4528 MHz	HIRESNONINT. High resolution, non-interlaced.
001	55.2614 MHz	HIRESINT0. High resolution, interlaced 0.
010	53.7264 MHz	STEREO. Stereo at half resolution.
011	--.---- MHz	EXTPIXCLK. External pixel clock source.
100	49.3495 MHz	HIRESINT1. High resolution, interlaced 1.
101	14.7500 MHz	PAL. PAL/SECAM (European TV) timing.
110	12.2098 MHz	NTSC. RS-170-A (USA TV) timing.
111	--.---- MHz	MAINTCLK. Maintenance clock, for diagnostics.

Note: The RAMDACs must be reset after the OSCSEL bits are changed. See description of bit 0, *DACRESET in the maintenance register for details.

*V-DIV-CK-3	13	Video divide clock by three (active low). This bit was originally intended to control whether the video RAM shift clocks were generated by dividing VnCLK by three or by six, which would have been used to allow two pixel resolution in the video timing. It was decided that this feature was not necessary, so now this bit is spare.
DSPINTRL	12	Display is interlaced. This bit is used to inform the cursor control PAL that an interlaced format is being used. A logic high indicates that the display is interlaced. A logic low indicates that the display is not interlaced.
STORINTRL	11	Image is stored interlaced. This bit is used to inform the video control logic (Field/Row PAL) that the image is stored in an interlaced format. If the image is stored <i>interlaced</i> , then transfer cycles by the VRAMs are to be performed every two rows. If the image is stored <i>non-interlaced</i> , then transfer cycles are to be performed every four rows. This distinction is important if stereo images are to be stored in <i>split</i>

frame buffer style, where the left eye view is stored in lines 0 - 511 of the frame buffer, and the right eye view is stored in lines 512 - 1023 of the frame buffer. Split frame buffer storage is considered *non-interlaced*, but can be displayed in an interlaced fashion on the monitor. A logic high indicates that the image is stored interlaced, and a transfer cycle should be performed every two rows. A logic low indicates that the image is stored non-interlaced, and a transfer cycle should be performed every four rows.

VLDINSTLD	10	Valid planes option installed. This bit informs the pixel pipeline control logic that the valid planes option either is or is not installed in the frame buffer. A logic high indicates that the valid planes are installed in the frame buffer. A logic low indicates that the valid planes are not installed in the frame buffer.
MAINTCLK	9	Maintenance pixel clock. This bit can be selected as a source of the pixel clock for the pixel output pipeline. If this bit is selected as a source of the pixel clock, the bit can be written repeatedly with a logic low followed by a logic high, which will single-step the pixel output pipeline. This capability can be used to test the functionality of the pixel output pipeline. Note: The video RAMs used on the frame buffer cards have a static shift register, so this clock can be run arbitrarily slow without loss of data.
MREGSPARE1	8	Spare bit in maintenance register. Reserved for future use.
MREGSPARE0	7	Spare bit in maintenance register. Reserved for future use.
*PADROE	6	Pixel address output enable. This bit is used to select between the two possible sources for addressing into the WLUT and CLUT. A logic high select the Gbus address. A logic low selects the address from the pixel output pipeline. This bit should be high when reading and writing locations in the WLUT. This bit

should be low when video data are being displayed. This bit should be low when testing the pixel output pipeline using the maintenance pixel clock.

SUBTAB[1]	5	RAMDAC sub-table select bit 1.
SUBTAB[0]	4	RAMDAC sub-table select bit 0. SUBTAB[1-0] select which of four possible color lookup sub-tables in the RAMDAC are used by 8-bit color coded using 4K color table windows. All 8-bit color coded using 4K color table windows use the same RAMDAC sub-table.
		Note: Eight-bit color coded using 4K color table windows should not be confused with 8-bit color coded using RAMDAC windows.
*BSVCLR	3	Bit slice vertical clear. This bit is used to clear the signal *VRINT, the DSP vertical retrace interrupt. When the DSP chips which to clear the vertical retrace interrupt, they must write this bit to logic low to clear the interrupt, and then write it back to a one to enable future interrupts.
*MREGCBLNK	2	Maintenance register composite blank. This bit is used to disable video output. When this bit is low the output video will be forced to the blanking level. This bit should be high for normal video display.
*FBRESET	1	Frame buffer reset. This bit allows the pixel processors to be reset in a controlled way. A logic low in this bit will cause the pixel processors to be reset. This bit should be high for normal operation.
*DACRESET	0	Digital to analog converter pipeline reset. This bit allows the internal pipeline delay of the RAMDACs to be set so that all RAMDACs are aligned. This bit should be held low through one complete vertical blank interval whenever the oscillator select bits are changed. RAMDAC blinking will not work properly when this bit is low, so this bit should be high for normal operation.

Memory Maps

Figure 7-6, "Video Interface Address Space," is a memory map of the video card.

I/O Panel Connections

The following is the I/O panel pinout for the video card. The card has three connectors on the I/O side. These connectors are used for connections to the monitor, the video option board, the stereo shutter system, and any other external video devices. There is one 8-pin right angle header, one 20-pin right angle header, and one 10-pin right angle header.

Video Connector

This is an 8-pin right angle header that uses a recommended 75 Ω coax cable.

PC Connector (2x4)

E&S part number: 401264-004

Latches

E&S part number: 401262-001

Pinout

Pin 1: Blue

Pin 2: Ground

Pin 3: Green

Pin 4: Ground

Pin 5: Red

Pin 6: Ground

Pin 7: Comp Sync

Pin 8: Ground

Signal Definition

RED: Red video out. The signal conforms to RS-343-A standard voltage levels when terminated in 75 Ω to ground.

GREEN: Green video out with optional composite sync. The signal conforms to RS-343-A standard voltage levels when terminated in 75 Ω to ground.

BLUE: Blue video out. The signal conforms to RS-343-A standard voltage levels when terminated in 75 Ω to ground.

CSYNCOU: Composite sync output, TTL levels.

Video Option Connector

This is a 20-pin right angle header that uses a recommended 20 conductor twist/flat.

A80000	A80000	Color Lookup Table 0	
	A84000	Color Lookup Table 1	
	A88000	Color Lookup Table 2	
	A8C000	Color Lookup Table 3	
A8FFFF			
A90000	A90000	WindowLookup Table 0	
	A94000	WindowLookup Table 1	
	A98000	RAMDAC Address Register Low	
	A98800	RAMDAC Address Register High	
	A99000	RAMDAC Alt, Overlay Register	
	A99800	RAMDAC Primary Color Palette	
	A9A000	Cursor Address Register 0	
	A9A800	Cursor Address Register 1	
	A9B000	Cursor Ram	
	A9B800	Cursor Control Register	
	A9C000	Video Timing Gate Array	
	A9FFFF	A9E000	Maintenance Register

Figure 7 - 6 Video Interface Address Space

PC Connector (2x10)

E&S part number: 801290-120

Pinout

Pin 1: %EXTPIXCLK+. External pixel clock, positive polarity ECL.

Pin 2: %EXTPIXCLK-. External pixel clock, negative polarity ECL.

Pin 3: *CLRHHN. Clear horizontal in. Used for genlock.

Pin 4: Ground.

Pin 5: *CLRHHN. Clear vertical in. Used for genlock.

Pin 6: Ground.

Pin 7: *CSYNCOU. Composite sync output, TTL levels.

Pin 8: Ground.

Pin 9: *HSYNCOU. Horizontal sync output, TTL levels.

Pin 10: Ground.

Pin 11: *VSYNCOU. Vertical sync output, TTL levels.

Pin 12: Ground.

Pin 13: *CBLANKOU. Composite blank output, TTL levels.

Pin 14: Ground.

- Color lookup table, VTGA,
- RAMDACs,
- Cursor chips, and
- Pixel pipe.

Analytic diagnostics test all registers and memories through the Gbus interface.

Visual diagnostics use observable pixel color to test the following:

- Pixel data correctly accesses the window lookup table, the color lookup table, and the RAMDAC tables,
- Cursor functionality,
- Ability of window lookup table data to control the pipe,
- DAC performance,
- Compare to zero for underlay mode,
- Correct multiplexer functioning for the data paths corresponding to each window mode,

- Valid data comparison.

Visual diagnostics are also used to implement full speed pipe tests cycling through color data patterns, window data patterns, and window modes.

Program Execution

The diagnostic has two user modifiable parameters:

- 1) The user may select from a menu of video timing characteristics. The user's choice (or a default) is used by the diagnostic to select a set of initialization values for the video timing gate array.
- 2) Second, the user may indicate whether the frame buffer is installed or not. If it is not installed, the memory diagnostics can still be run with a few changes, but the visuals used to test the pixel pipe cannot be run.

8. Display Adjustment

Display Adjustment

This section covers field procedures for tuning and alignment of the *Mitsubishi Color Display Monitor*. Adjustment procedures are discussed in two sections, Screen Image or User Adjustments and Internal or Depot Adjustments.

The display monitor can memorize a total of 20 different screen image timings. The Auto Channel can memorize 11 of these timings. Seven function settings can be adjusted and stored for each of 22 different timing formats. The Auto Channel contains 11 formats. Enhanced channels CH1 to CH7 contain 1 storage location for each channel; 3 channels for VGA (CH8) and 1 channel for MacII (CH9).

Channel switch settings have no effect on the monitor's auto-sync ability. The monitor will lock on any timing format within its range of operation.

Image Data Formats

Image data can be saved in each of the following formats:

CH0 Auto Channel - 11 separate areas are used to store 30-64KHz of horizontal frequency and screen image data. Horizontal frequencies must be 3 KHz or more for the monitor to select the correct screen image data.

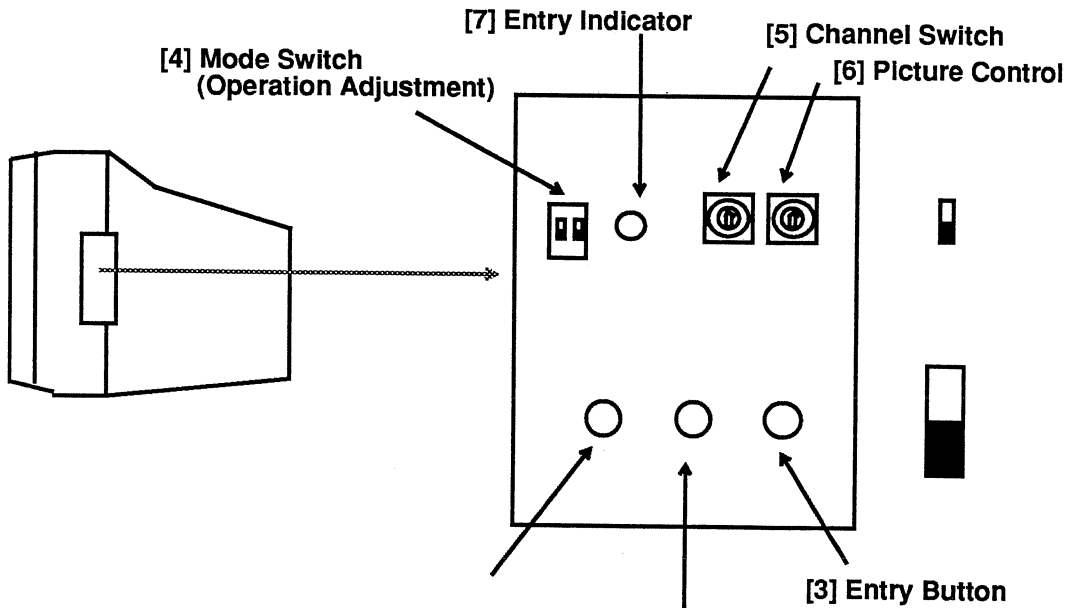
CH1 to CH7 Enhanced Channels - Each channel stores screen image data for one format.

CH8 VGA Enhanced Channel - Requires separate vertical and horizontal sync. Channel selection determined by sync polarity.

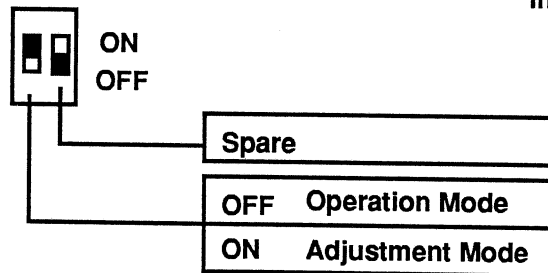
CH9 Enhanced Channel - MAC II

Screen Image Adjustments

All screen image adjustments are made with 2 rotary switches, 1 slide switch and 3 push buttons located under the right side panel of the monitor. To access these adjustments remove the 8 panel screws and slide the panel from the rear of the monitor.

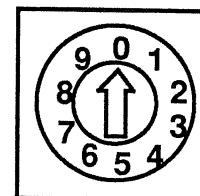


[1] Adjustment Button Decrease [2] Adjustment Button Increase



[5] Channel Switch

Channel		
CH0	Pre - settable by User 11 Timings	Auto Channel
CH1 to CH7	Pre - settable by User 1 Timing for CH	Enhanced Channel
CH8	VGA 3 modes	
CH9	MAC II	



Channel Selection

Selecting a channel for adjustment:

- 1) Set the channel switch to CH0 (Auto Channel) or CH1 - CH7 (Enhanced Channel).
- 2) Set the mode switch to the ON (ADJUSTMENT) position.
- 3) Adjust channel. After the adjustment push Entry SW [3] to store.

Note: If the Entry Indicator [7] is blinking, this area has been previously stored. New data can be written over the old data by pushing SW [3] again. It is not necessary to depress the entry button after each adjustment. Depressing the Entry Switch is required only once and all current parameters will be saved. Some adjustments interact and it will be necessary to repeat the adjustments.

H Phase Adjustment

The H Phase adjusts the relative position of the screen image with the raster.

- 1) Set the picture control rotary SW [6] to 1.
- 2) Push the Adjustment button [1] to move the screen image to the right or [2] to move to the left.
- 3) After adjustment is made push Entry SW [3] to store.

H Size Adjustment

- 1) Set the picture control rotary SW [6] to 2. 13³/₄
- 2) Push the Adjustment button [1] to widen screen image (raster size) or push [2] to narrow.
- 3) After adjustment is made push Entry SW [3] to save.

H Position Adjustment

The H Position adjustment centers the background raster on the display.

- 1) Set the picture control rotary SW [6] to 3.
- 2) Push the Adjustment button [1] to move raster position to right or push [2] to move to left.
- 3) Push Entry SW [3] to save.

V Position Adjustment

The V Position centers the image vertically on the display.

- 1) Set the picture control rotary SW [6] to 4
- 2) Push the Adjustment button [1] to move raster position upward or push [2] to move downward.
- 3) Push Entry SW [3] to save.

V Size Adjustment

The V Size sets image size to 11 height by 13 and 3/4 width.

- 1) Set the picture control rotary SW [6] to 5.
- 2) Push the Adjustment button [1] to widen screen image (raster size or push [2] to narrow.
- 3) Push Entry SW [3] to save.

PCC - AMP Adjustment

The Pin Cushion - AMP is for straight sides correction.

- 1) Set the picture control rotary SW [6] to A.
- 2) Push the Adjustment button [1] or [2] to acquire the optimum raster shape.

PCC Phase Adjustment

This sets the parallel adjustment.

- 1) Set the picture control rotary SW [6] to B.
- 2) Push the Adjustment Button [1] or [2] to acquire the optimum raster shape.

9. ESV Workstation Diagnostics

Introduction

This chapter includes a description of the Test Command Language (TCL) being developed by the Design Systems Division Diagnostic Development group. The chapter also contains diagnostic descriptions for the following:

- CPU card,
- DSP card,
- GSE card,
- Video card,

TCL - Test Command Language

Introduction

This section describes the Test Command Language - TCL - developed by Design Systems Division Diagnostic Development. The language is the interface used for diagnostics for both manufacturing test and field service. TCL is a command language and interface that allows a diagnostic technician to test Evans & Sutherland hardware from a host computer. The interface and error reporting are standardized across all diagnostic tests.

Test Initiation

TCL is embedded in each diagnostic. This means that the user need only execute the diagnostic (by typing the name at the UNIX shell prompt) in order to interact with TCL. Arguments given on the command line can modify the behavior of TCL.

It is possible to alter certain modes of the command interpreter (*e.g.* the verbosity level or the error action) by running the test with certain command line arguments. It is also possible to supply command line arguments which set the phase(s) to be run by TCL and cause the test to be immediately initiated on startup.

Help Availability

Help is available for the command interaction with TCL and also for specifics of the test. The help command gives information about TCL interaction. This includes a short summary about each of the TCL commands. The **show phases** command gives specific information about the test phases for the current card or unit under test (UUT). This consists of a phase number and a brief description of the phases.

Diagnostic UUT and Test Phases

There are diagnostic tests for each UUT in a system. Each diagnostic defines the phases which are run to test the hardware. The prompt indicates the selected UUT.

The TCL command interpreter allows the user to control which phases are to be run, how many times the phases should be run, and how errors are to be reported. Phases are referred to using their phase number. Phase numbering starts at 1 and continues sequentially up to the last phase of the selected test.

The UUT may be selected by typing the UUT acronym at the current prompt. The default on startup is all system cards.

The user may control the phase(s) to be run using the **set range** command. This command takes a phase list *plist* as the argument. The command **set range 6,6,3,2,1-8 8-10** runs phase 6 two times then phase 3,2, and 1, then phases 1 through 8, then phases 8, 9, and 10 in that order when the **go** command is executed. Some of the diagnostic phases may be optional. These phases are always the highest numbered phases.

Error Handling

The user can control the behavior of the test when an error is detected with the **set action** command. The action may be set to **abort**, which causes immediate termination of the test phases; **cont**, which continues the test after reporting the error; **skip**, which skips to the next phase of the current phase sequence; **retry**, which restarts the current phase (and will probably generate the same error again); or **user**, which prompts the user for the action to take after reporting the error.

Additional control over error handling is given through the **set limit** and **set plimit** commands. These allow the user to set a maximum number of errors which will be reported for the entire test or for an individual test, respectively. When the limit is exceeded, the test is aborted.

Error Reporting and Handling

Errors may be reported to the debug terminal, to an error log file, to neither, or to both. The user controls error reports to the terminal and to the log file independently.

The user may set an error log file with the **set errorlog** command. This command takes a file path name as an argument. The command attempts to open the file for writing. If successful, all error reports will be output to the file.

The output to the debug terminal is controlled with the **set verbosity** command. The level of verbosity may be set to the following

- **verbose**,
- **normal**,
- **brief**, or
- **quiet**.

Batch Processing

Command input to TCL can be redirected to come from a command file. This is accomplished with the **source** command. Input may also be redirected as usual on the UNIX shell command line using the < operator.

Special Debugging

The **read** and **write** commands allow the user to directly read and write physical devices. The commands take an offset which determines the actual address which is accessed. The control registers for most devices start at offset 0. Access to hardware may be by byte, word, or double word. If the **read** or **write** commands must access more than one device driver, the **device** command is used to switch the device driver that is accessed.

The **debug** command provides an interface to diagnostic-specific routines to access and stimulate the hardware.

Tcl Command Line Arguments

This section details the command line arguments which are allowed by TCL. The command line arguments are listed here with a description of their effects.

- g** Begin the test phases immediately with no user interaction.
- i** Run the test interactively. (Opposite of **-g**.) This is the default mode, unless phases are supplied on the command line. See **<plist>** below.
- v** Set verbosity level to **verbose**. This is the default verbosity level.
- b** Set verbosity level to **brief**.
- q** Set verbosity level to **quiet**.
- a** Set error action to **abort**.
- c** Set error action to **cont**.
- s** Set error action to **skip**.
- u** Set error action to **user**. This is the default error action.
- r <n>** Set the repetition count to **<n>**. **<n>** should be a number with a leading space. The default repetition count is 1.

-
- l <n>** Set the test error limit to **<n>**. The default error limit is 1000.
 - p <n>** Set the phase error limit to **<n>**. The default phase error limit is 0.
 - e <fname>** Set the error log file to **<fname>**. There is no default error log file.
 - <plist>** Specify the phase order list. Phase numbers should be separated by commas. Two numbers separated by a minus sign are interpreted as the range of phase numbers from the first to the second, inclusive.
 - ""** If **<plist>** is supplied and the test is run without the **-l** option, the test phases are immediately initiated with no user interaction.

In addition to the command line arguments listed above, there may be test specific arguments which are allowed. These arguments are always designated by capitalized letters.

Note: It is not possible to set the error action to **retry** from the shell command line. The utility of such a capability is questionable.

Tcl Commands

This section details the actions of TCL commands. Each command is listed with its proper syntax and a description of its effect. Command arguments must be separated by a space

For the set commands (for example, **set range**) the word **set** may be omitted, and the argument may be entered directly as the command.

set range <plist>

This command sets the range and sequence of phases which will be run. The test phases are executed when the user gives the **go** command. **<plist>** is a list of phase numbers separated by commas or a starting phase number and an ending phase number separated by a minus sign. The maximum phase number varies for each test, but can be found with the **show phases** command. If **<plist>** is not supplied, the phase sequence defaults to 1-n where n is the highest numbered non-optional phase.

set repeat <n>

This command sets the number of repetitions of the test phases. If **<n>** is 0, the test will loop forever. (Leave it to a programmer to equate zero with infinity.) If **<n>** is not supplied, the repetition count defaults to 1.

set limit <n>

This command sets the maximum number of errors which will be output to the debug terminal or to the error log file for the duration of the test. If more than **<n>** errors are found, the test is aborted. If **<n>** is 0, the test will not abort on an error limit. If **<n>** is not supplied, the error limit defaults to 1000.

set plimit <n>

This command sets the maximum number of errors which will be output to the debug terminal or to the error log file during any single phase. If more than **<n>** errors are found, the current phase is aborted, and test execution continues with the next phase. If **<n>** is 0, the test will not abort on a phase error limit. If **<n>** is not supplied, the phase error limit is set to 0.

set errorlog <fname>

This command sets the error log output to the file specified by **<fname>**. If an error log file previously existed, it is closed. **<fname>** is opened for writing and all subsequent error information is output to that file.

Informational messages and user prompts are not output to the error log file. If no errors occur during testing, the error log file will be empty.

If **<fname>** is not supplied, any previously open error log file will be closed and no further error information is output to an error log file.

set verbosity {quiet | brief | normal | verbose}

When the verbosity level is **verbose**, all informational and error messages are output to the debug terminal plus detailed instructional and informational messages in some test phases.

When the verbosity level is **normal**, the default verbosity, all informational messages and error messages are output to the debug terminal. This includes feedback messages detailing what circuitry is being currently tested, and dots (.) which are periodically output to the debug terminal to indicate that the test is proceeding normally.

When the verbosity level is set to **brief**, only error messages and short phase initiation messages are output to the debug terminal.

When the verbosity level is set to **quiet**, no messages are output to the debug terminal. However, prompts are still displayed to the debug terminal if it is the input source.

Set verbosity may be omitted.

set action {abort | cont | skip | retry | user}

This command sets the behavior of a test when an error is detected and reported. If the error action is set to **abort**, the test is aborted when an error occurs.

If the error action is set to **cont**, the test continues after reporting the error.

If the error action is set to **skip**, the current phase is aborted, and the next phase is initiated.

If the error action is set to **retry**, the current phase is restarted. Ordinarily, this causes the same error to occur.

If the error action is set to **user**, the default error action, the user is prompted for the action to take when an error occurs.

If the input source is not interactive (*e.g.*, a command file) and the error action is **user**, the test simply continues.

Set action may be omitted.

**show {phases | repeat | limit | plimit | action| verbosity | errorlog
| devices | status}**

The show command is used to display information about the state of the command interpreter on the debug terminal. **show range** displays the current phase range list setting.

show repeat displays the repetition count.

show limit displays the maximum error limit.

show plimit displays the maximum phase error limit.

show action displays the current error action.

show verbosity displays the current verbosity level.

show errorlog displays the name of the current error log file.

show devices displays the names of the active devices and the offsets which can be accessed via the **examine** and **deposit** commands.

show status displays all of the above. If the **show** command is run without an argument, **show status** is assumed.

show phases displays a list of all the phases for the current UUT. The list includes phase number and one-line description. The same one-line description is printed at the beginning of each phase if the verbosity level is set to **normal** or **verbose**.

version

This command displays the current version number of this diagnostic test.

go <plist>

This command initiates the test phases. If **<plist>** is provided, it determines the phases which will be run, but does not change the phase range set by the **set range** command. If **<plist>** is not provided, the phases specified by the **set range** command will be used.

quit

This command exits the TCL command interpreter.

help <topic>	This command displays information about TCL commands. If <topic> is not supplied, a command summary is displayed on the debug terminal.
select <menu>	This command allows the user to set any test specific options which may exist for the test. This is done by interacting with simple menus.
debug	The debug command provides an interface to diagnostic-specific routines to access and stimulate the hardware.
device <devname>	The device command allows the user to choose the device which the read and write commands access. The <devname> is usually a special file which is located in the /dev directory. If <devname> is omitted, the next device shown by the show devices command will be selected. Most diagnostics will only open a single device at a time, so the device command will not be used frequently.

read [{b | w | d}] <start> [:<cnt>] [*<reps>]

read [{b | w | d}] <start> [-<end>] [*<reps>]

<start> <end> expressed as number or <symbol> [+<offset>]

The **read** command allows the user to read physical addresses which are defined for the current test. An error occurs if the address specified is not defined for the hardware being tested.

Hardware access may be by byte, by word, or by double word by specifying **b**, **w**, or **d**, respectively. If no hardware access mode is supplied, word mode is assumed.

Start and end addresses may be expressed as a number or a symbol or a symbol plus an offset.

The user may specify a block of data to be read by including a count after the start address. The count is separated from the start address with a colon (:). The user may, alter-

natively, specify the end address of the block. The end address is separated from the start address with a dash (-).

The **reps** option indicates how many times to repeat the range of accesses. A **reps** equal to 0 will loop forever.

write [{b | w | d}] <start> [:<cnt>] [*<reps>] <data>

write [{b | w | d}] <start> [-<end>] [*<reps>] <data>

<start> <end> expressed as number or <symbol> [+<offset>]

The **write** command allows the user to write to physical addresses which are defined for the current test. An error occurs if the address specified is not defined for the hardware being tested.

Hardware access may be by byte, by word, or by double word by specifying **b**, **w**, or **d**, respectively. If no hardware access mode is supplied, word mode is assumed.

Start and end addresses may be expressed as a number or a symbol or a symbol plus an offset.

The user may specify a block of data to be read by including a count after the start address. The count is separated from the start address with a colon (:). The user may, alternatively, specify the end address of the block. The end address is separated from the start address with a dash (-).

The **reps** option indicates how many times to repeat the range of accesses. A **reps** equal to 0 will loop forever.

Command Summary

Set Commands

set phases<plist>
set repeat <n>
set limit <n>
set plimit <n>
set errorlog <fname>

set verbosity quiet
set verbosity brief
set verbosity verbose

set action abort
set action cont
set action skip
set action retry
set action user

Show Commands

show phases
show repeat
show limit
show plimit
show action
show verbosity
show errorlog
show status
summary
version

Control Commands

go <plist>
source <fname>
quit
suspend

Help Commands

help <topic>
info
options

Debug Commands

debug <fname>
select <devname>
examine [{b | w | d}] <start> [:<cnt>] [*<reps>]
examine [{b | w | d}] <start> [-<end>] [*<reps>]
deposit [{b | w | d}] <start> [:<cnt>] [*<reps>] <data>
deposit [{b | w | d}] <start> [-<end>] [*<reps>] <data>

ESV System Diagnostic

Name

esvsysdiag - ESV system diagnostic

Release

Release 1.0, April, 1990

Description

The ESV system diagnostic is designed to isolate a failing field replaceable unit (FRU). Testing is divided into tests for each FRU, which are in turn divided into test phases. Testing is organized to the proper system hierarchy so hardware is tested before it is used in testing of a dependent hardware unit. The diagnostic runs to completion and then reports all the error information which was queued up as the tests were running.

Tests are included for the CPU, GSE, DSP cards, Frame Buffer cards, and the Video Board in that order. A message identifying the FRU under test is displayed when testing for the FRU begins. A message describing each test phase is displayed when it is executed.

The diagnostic runs test phases until an error is encountered. Then the following occur:

- A message indicating the error number of the encountered error is displayed.
- Error number and other information about the error is entered into a queue.
- Test phases depending on the failing hardware are disabled.

Testing is continued at the next phase unless the error is of a type where continued testing would be meaningless. If a test phase has been disabled do to a previous error the message "-- Test disabled" is displayed after the phase description message.

After testing is finished a summary of the errors is presented from the error information queued up as the tests executed. The summary information starts with the number of errors encountered and then provides information on each error. Error information is displayed one error at a time so information won't scroll of the screen until you have viewed it. Error information includes up to three potential FRUs causing the error, with the most likely listed first.

Execution

The ESV system diagnostic runs under UNIX and is executed out of the directory **/usr/people/fstest/diag** by typing **esvsysdiag** at the UNIX prompt. The system diagnostic needs to be run remotely or from an ASCII terminal connected to the mouse/RDC port on the I/O panel. Do not attempt to run it from the workstation keyboard because the graphics hardware under test is also needed to display console text.

When the system diagnostic starts up it automatically determines the system hardware configuration and reports it to you. Be sure and compare the reported to the actual hardware configuration as further testing will be based on this configuration. Any hardware present in the system not reported should be considered faulty and replaced.

After the system configuration is reported, you are prompted to answer some questions to setup the type of testing you want to do. You may skip all the analytic testing and go directly to the Display Visuals if you choose. In this version you may disable the long frame buffer test. And you may set a loop count of how many times to go through the tests excluding the visuals. Each of these options have a default noted in square brackets which will be entered if you simply enter a return.

esvsysdiag**Name****esvsysdiag** - ESV system diagnostic**Synopsis****esvsysdiag [-giacsuvnbq -r <n> -l <n> -p <n> -e <logfile>****Description**

The ESV system diagnostic is a system wide test for the ESV, ARS, FONI and other optional system hardware. The diagnostic will attempt to auto-configure and report optional hardware detected.

The ESV system diagnostic is divided into tests for each FRU, which are in turn divided into test phases. Testing is organized to the proper system hierarchy, so hardware is tested before it is used in testing of a dependent hardware unit. The diagnostic runs to completion, then reports all the error information that was queued up as the test were running. Tests are included for the CPU, GSE, DSP cards, Frame Buffer cards, and the Video card in that order. ARS, FONI, and testing of other options will follow. A message describing each test phase is displayed when the phase is executed. The diagnostic runs test phases until an error is encountered. Then the following occur:

- A message with a brief description and error number of the encountered error is displayed.
- Error number and other information about the error is entered into a queue.
- Test phases depending on the failing hardware are disabled.

Testing is continued at the next phase unless the error is of a type where continued testing would be meaningless. If a test phase has been disabled the message -- Test disabled is displayed after the phase description message. After testing is finished a summary of the errors is presented from the error information queued up as the test executed. The summary information starts with the number of errors encountered, then provides information on each error. Error information is displayed one error at a time so information doesn't scroll off the screen before you view it. Error information includes up to three FRUs that are potential sources of the error, with the most likely listed first.

If the system has an ARS installed it should be detected during the system auto-configure. If it is not detected the HIC card would be a prime suspect. ARS testing will test the HIC, Chick, SHINFO, Memory, TexServer, Smart Memory and the FOP in that order.

Execution

The ESV system diagnostic runs under UNIX and is executed out of the directory `/usr/people/fstest/diag` by typing `esvsysdiag` at the UNIX prompt. The system diagnostic needs to be run from an ASCII terminal connected to the mouse/RDC port on the I/O panel. Do not attempt to run it from the workstation keyboard because the graphics hardware under test is also needed to display console text.

It is necessary for the keyboard to be connected for UNIX. If the keyboard is connected through the RDC and you remove the RDC to connect a terminal, then you must reconnect the keyboard to the I/O panel keyboard connector. When the system diagnostic starts, it automatically determines the system hardware configuration and reports configuration as further testing will be based on this configuration. Any hardware options present in the system but not reported should be considered faulty and replaced.

After the system configuration is reported, you are prompted to answer some questions to setup the type of testing you want to do. If an ARS is present you may choose to test the ARS or the ESV only. For a quick system verification, you may disable the interactive visuals and tests which take a long time to execute. Each of these options has a default noted in parenthesis which is entered when press RETURN. You may execute `esvsysdiag` with an `-i` option which allows you to interact with the TCL command interpreter and focus testing on a selected hardware unit. All the TCL command line options are described in the following section.

Auto-configuring of multiple ARC chick cards is not in place in this release. To test a second chick card the diagnostic must be run with the `i` option so you can interact with TCL and use `select` command to select the second chick card then run tests. More diagnostic information can be obtained from the on-line manual command built into TCL. To execute the manual command the diagnostic must be run in the interactive mode.

Options

This section details the command line options which are allowed by TCL. The command line options are listed here with a description of their effects.

- g** Begin the test phases immediately with no user interaction. This is the default for system level diagnostics.
- i** Run the test interactively. (Opposite of **-g**) This is the default mode for component level diagnostics, unless phases are supplied on the command line. See `<plist>` below.
- v** Set verbosity level to **verbose**. This is the default verbosity level.
- b** Set verbosity level to **brief**.

-
- q** Set verbosity level to **quiet**.
 - a** Set verbosity level to **abort**.
 - c** Set error action to **cont**.
 - s** Set error action to **skip**.
 - u** Set error action **user**. This is the default error action.
 - r <n>** Set the repetition count to **<n>**. **<n>** should be a number with a leading space. The default repetition count is 1.
 - l <n>** Set the test error limit to **<n>**. The default error limit is 1000.
 - p <n>** Set the phase limit to **<n>**. The default phase error limit is 0.
 - e <fname>** Set the error log to **<fname>**. There is no default error log.
 - <plist>** Specify the phase order list. Phase numbers should be separated by commas. Two numbers separated by a minus sign are interpreted as the range of phase numbers from the first to the second, inclusive.
 - ''' If **<plist>** is supplied and the test is run without the **-i** option, the test phases are immediately initiated with no user interaction.

Note that it is not possible to set the error action to retry from the shell command line. The utility of such a capability is questionable.

E&S Diagnostic Software License

PLEASE CAREFULLY READ THE FOLLOWING BEFORE USING THE DIAGNOSTIC SOFTWARE BEING PROVIDED TO YOU BY EVANS & SUTHERLAND COMPUTER CORPORATION. USING THE DIAGNOSTIC SOFTWARE INDICATES YOUR ACCEPTANCE OF EACH OF THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE WITH WHAT THIS AGREEMENT SAYS, DO NOT USE THE SOFTWARE. PROMPTLY RETURN THE EQUIPMENT ALONG WITH ALL THE SOFTWARE (INCLUDING WRITTEN MATERIAL) TO EVANS & SUTHERLAND FOR A REFUND.

In consideration of the payment of a per unit license use fee, Evans & Sutherland Computer Corporation, herein referred to as E&S, grants you a non-exclusive license to use the Diagnostic Software, as defined herein, under the following terms and conditions.

1.0 DEFINITIONS

"Diagnostic Software" refers to those computer program products in both tape and diskette format, the executable version of which customer has licensed from E&S together with all documentation and manuals furnished therewith, and includes any and all future Diagnostic Software shipped to you by E&S.

"Site Location" refers to a physical location usually associated with a single address.

2.0 TITLE

Title to and ownership of the Diagnostic Software, and all copies thereof, shall at all times reside with E&S or with E&S's licensors in the event the Diagnostic Software is provided by E&S under license from a third party.

3.0 GRANT OF LICENSE

E&S grants you a personal non-exclusive and non-transferable right and license to use the Diagnostic Software which is either resident on diskettes or provided on tape for use within a single Site Location, with the specific equipment listed in Addendum A, Description of licensed equipment.

Any attempt to sublicense, transfer, or assign the Diagnostic Software or the rights or obligations under this site license to any party without prior written approval of E&S shall be void.

4.0 SCOPE OF LICENSE

You may make additional copies of each media in support of the use of the Diagnostic Software at said Site Location. Such copies shall not be distributed to other Site Locations without the prior written approval of E&S. You must reproduce and include the copyright notice and proprietary notices of E&S or its licensors on the original and on all copies of the Diagnostic Software.

In the event that E&S provides you with a new version of the Diagnostic Software, or authorizes you to make additional copies of the new versions of the Diagnostic Software to replace the Diagnostic Software licensed hereunder, then such replacement Diagnostic Software and all copies thereof shall be covered by and subject to the terms and conditions of this License Agreement. Copies of old versions of Diagnostic Software shall continue to be subject to this Agreement until destroyed by you or returned E&S.

5.0 CONFIDENTIALITY

The Diagnostic Software constitutes valuable proprietary assets of E&S, embodying substantial creative efforts and significant expenditure of time and money. You hereby agree to observe complete confidentiality with respect to the Diagnostic Software, including but not limited to the following: (a) you agree to limit access to the Diagnostic Software and to assure that anyone who is permitted access to the Diagnostic Software is made aware of and agrees to abide by the obligations imposed on you under this Agreement; (b) you agree not to alter or remove E&S proprietary and copyright notices for the Diagnostic Software and to reproduce and include such notices on any copies or merged portions in any form of the Diagnostic Software; (c) you agree not to attempt to disassemble, decompile or otherwise reverse-engineer the Diagnostic Software; (d) you agree (i) not to make unauthorized copies of all or any portion of the Diagnostic Software, (ii) not to sell, rent, sublicense, give or otherwise disclose, distribute or transfer to any third party any portion of the Diagnostic Software or copies thereof, and (iii) not to install the Diagnostic Software on a service bureau or other remote access system whereby persons or utilities other than you can benefit from the use of the Diagnostic Software.

You agree that in the event of an unauthorized reproduction, transfer or disclosure of any part of the Diagnostic Software, E&S will not have an adequate remedy at law, and therefore injunctive or other equitable relief will be appropriate to restrain such reproduction, transfer or disclosure, threatened or actual.

The provisions regarding restrictions on use and confidentiality contained in this Agreement shall survive any termination of the Agreement.

1.0 MEDIA WARRANTY

E&S warrants to you that the media on which the Diagnostic Software is furnished shall be free from defects in material and workmanship under normal use for a period of ninety (90) days from the date of shipment. E&S will replace any media that fails during the warranty period and is returned to E&S under warranty.

2.0 DIAGNOSTIC SOFTWARE WARRANTIES

E&S warrants that the Diagnostic Software will perform substantially in accordance with the applicable E&S published release note specifications relating to the release of the Software for a period of ninety (90) days from the date the Diagnostic Software is shipped to you by E&S. E&S does not warrant that the functions contained in the Diagnostic Software will meet your requirements or that the operation of the Diagnostic Software will be uninterrupted or error free, or that all defects in the Diagnostic Software will be corrected.

E&S's sole obligation and your exclusive remedy under this warranty shall be limited to E&S's using its best efforts at its own expense during the warranty period, to bring the Diagnostic Software into substantial conformity with applicable specifications and, where such efforts have been successful, to provide you with a corrected version of the Diagnostic Software, or at its option, E&S may provide you with a reasonable work-around solution.

3.0 UPDATES

This license does not grant you any rights, license, or interest in any improvements, modifications, enhancements, or updates to the Diagnostic Software and Documentation. Updates may be obtained at E&S's current rates.

4.0 SUPPORT AND TRAINING

All telephone technical support and training may be obtained at E&S's current rates.

5.0 LIMITATION OF LIABILITY

THE FOREGOING WARRANTIES ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE WARRANTIES ARE NOT TRANSFERABLE.

IN NO EVENT WILL E&S BE LIABLE FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING FROM THE USE, PERFORMANCE OR NON-PERFORMANCE OF THE DIAGNOSTIC SOFTWARE INCLUDING ANY LOST PROFITS OR LOST DATA, EVEN IF E&S HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

BECAUSE SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

E&S's liability to you for actual damages for any cause whatsoever, and regardless of the form of the action, will be limited to the money paid for the E&S software that caused the damages or that is the subject matter of, or is directly related to, the cause of the action.

6.0 TERM

This Agreement, together with E&S's standard Terms and Conditions for sale, and if applicable, standard Terms & Conditions for service contract sale, sets forth the entire agreement with respect to the license of the Diagnostic Software to you by E&S. This license shall remain in effect until terminated. You may terminate this license by destroying all copies, modifications and merged portions of the software in your possession, and providing written notice of such termination and destruction to E&S. The license granted under this Agreement will terminate automatically without notice if you violate any of the terms and conditions of this Agreement. You agree upon such termination to promptly return or certify in writing to E&S the destruction of the Software and all related documentation and all copies of either in any form.

7.0 GOVERNING LAW

This Agreement shall be governed by the laws of the state of Utah, except for that body of law dealing with conflicts of law. If any provision of this Agreement shall be held by a court of competent jurisdiction to be contrary to law, that provision will be enforced to the maximum extent permissible, and the remaining provisions of this Agreement will remain in full force and effect.

8.0 U.S. GOVERNMENT RESTRICTED RIGHTS LEGEND

If this E&S Software is acquired by or on behalf of a unit or agency of the United States Government, this provision applies. This E&S Software (a) was developed at private expense, and no part of it was developed with government funds; (b) is a trade secret of E&S for all purposes of the Freedom of Information Act; (c) is 'commercial computer software' subject to limited utilization as provided in the contract between the vendor and the government entity; and (d) in all respects is proprietary data belonging solely to E&S.

For units of the Department of Defense (DoD), this E&S Software is sold only with 'Restricted Rights' as that term is defined in the DoD supplement to the Federal Acquisition Regulations 52.227-7013 and: use, duplication, or disclosure is subject to restrictions as set forth in subdivision (b) (3) (ii) of the Rights on Technical Data and Computer Software clause 52.227-7013.

YOU ACKNOWLEDGE THAT: (a) YOU HAVE READ THIS ENTIRE AGREEMENT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS; (b) THIS AGREEMENT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE UNDERSTANDING AND CONTRACT BETWEEN US AND SUPERSEDES ANY AND ALL PRIOR OR WRITTEN COMMUNICATIONS RELATING TO THE SUBJECT MATTER HEROF; AND (c) THIS AGREEMENT MAY NOT BE MODIFIED, AMMENDED OR IN ANY WAY ALTERED EXCEPT IN WRITING SIGNED BY BOTH YOU AND E&S.

Should you have any questions concerning this Agreement, please contact Evans & Sutherland Computer Corp., Computer Division, 580 Arapeen Drive, Salt Lake City, UT 84108, (801)-582-5847

- 1. What is the default path to the ESV System Diagnostics?**
- 2. What command could be used to start an interactive diagnostic session on the ESV?**
- 3. What would happen if the system diagnostics were started with Xwindows running?**
- 4. What command can be used from within the system diagnostics to get help with a command's syntax?**
- 5. Where can on-line information be found with regard to the ESV System Diagnostics?**
- 6. What command can be used from within the diagnostics to direct the results to a file for future reference?**
- 7. What commands would be used to run the diagnostic on one dsp board only?**
- 8. Why is it sometimes necessary to run the diagnostics from a debug terminal or from another node on the network?**
- 9. What commands could be used to test the stereo option on the ESV?**
- 10. What cartridge tape contains a copy of the ESV System**

Diagnostics?

What package within the tape?

- 11. Install the diagnostic package into the directory /usr/tmp and start a diagnostic session from this copy.**
- 12. What single command could be used to run phases 10 through 20 five times and store the results in the file /tmp/errors?**
- 13. How can you find the version of the diagnostic installed on your machine?**
- 14. Connect a de-bug terminal to the ESV and start a diagnostic session.**

10. Networking

Network Protocols

The International Standard Organization's Open System Interconnection (ISO/OSI) model is the standard reference model used to describe network protocols. Networking Protocols described here are the various layers of software and hardware that information must pass through to communicate from one workstation to another over some type of networking medium. (See Figure 10-1.)

Low Level

Low level protocols handle the movement of data from one location to another. These protocols are implemented as part of the network hardware on the ESV workstation.

Physical Layer

The physical layer is the actual medium used to communicate over. In our case the medium is Ethernet coaxial cable and the data link layer is the Lance Ethernet chip and control circuitry on the CPU board. The data link detects and possibly corrects errors that may occur in the physical layer.

Higher level

Higher level protocols are implemented in software which address more complex issues such as reliability and information translation.

The network or "IP" layer handles network oriented functions, such as routing and buffering.

Transport Layer

Transport layers include the UDP or TCP layers. These layers control transportation from source to destination; including error checking, flow control, and re-assembling the message after transmission.

Session Layer

The session layer or RPC(remote procedure call), allows C programs to make procedure calls on other machines across the network.

Presentation Layer

A presentation layer known as "XDR" (External Data Representation) translates the data on one machine into a format recognized by any other machine on the network.

Application Layer

The Application layer, highest in this model is comprised of many user applications, mail, ftp, NFS, etc. This is the layer visible to the workstation user.

High Level Protocols	7 - Application	mail ftp	rcp NFS	rlogin NIS	rsh telnet
	6 - Presentation	XDR			
	5 - Session	RPC			
	4 - Transport	TCP		UDP	
	3 - Network	IP (Internetwork)			
Low Level Protocols	2 - Data Link	Ethernet		Point-to-point	
	1 - Physical	Ethernet		Point-to-point	

International Standard Organization's Open System Interconnection Model

Figure 10 - 1 Network Protocols

Network Hardware (802.3/Ethernet)

Controller

The controller refers to the ethernet controller used in the ESV Workstation. The controller used is sometimes referred to as the Lance Chip. The ethernet controller is actually an NCR 53C90, located on the CPU board. Network commands and software refer to it as **la0**.

Transceiver / Auxiliary Unit Interface (AUI) Cable

The ESV is equipped with a short transceiver cable that connects the ethernet port on the I/O bulkhead to a customer supplied transceiver or a multiplexer box. The transceiver cable is comprised of four twisted pairs for carrying transmit, receive, collision detection, and power. The length of a transceiver cable should not exceed a maximum of 50 meters.

Transceivers and Media Access Units (MAUs)

Transceivers and MAUs are devices that combine the functions of a transmitter and receiver to send and receive signals over the transmission media, normally coaxial cable. Transceivers are found in single and multiport versions. Multiport transceivers, multiplexers, generally provide up to eight separate transceiver connections and can be configured so that attached devices connect to an Ethernet network backbone.

Coaxial Cable Thick/Thin

Coaxial cable is normally the medium used for Ethernet communications. Ethernet coax comes in two varieties, thick and thin. Thick is usually more expensive than thin but provides higher capacity per segment and longer lengths per segment.

Thick

100 nodes per segment
1500 m. max / 2 repeaters

Thin

30 nodes per segment
555 m. max / 2 repeaters

Router

A router is used to join two local networks. A router could be any machine with two ethernet interfaces that supports a full functionality of TCP/IP. Routers require two internet addresses and two hostnames. Routers forward packets of a particular protocol from one logical network to another.

Gateway

Gateway and router are different in that a gateway enables networks using different protocols to communicate...translates one protocol to another.

Networking Tools

There are several commands that can be used on the ESV for analyzing network problems. These commands are available to the ESV workstation user. The following will give you a brief description of the command and how they can be used. For a complete description consult the *man Pages* or the MIPS User's Reference Manual.

ping

A quick and easy way to see if you can communicate with a particular host is to use the *ping* command. A datagram is sent to the specified host with instructions for it to be echoed or looped back to the sender. If there is no response, the host could be down or there may be a problem with the network.

The ping command can also be used to isolate network faults, check response time from a host, verify a host's Internet address, or check for packet loss between hosts.

spray

This command sends a stream of variable length packets to a specified host. The length and count of the packets can be varied to match the amount of stress you wish to place on the network. *spray* reports the number of packets received and the transfer rate.

netstat

The *netstat* command displays the contents of various network-related files. There are three forms of output that can be requested using *netstat*. Use of the interval option on *netstat* can provide you with a continuous display of packet traffic.

who & whodo

The *who* command will tell you what current users are logged into the machine and *whodo* tells what processes they are running at that moment.

arp

arp is a program which displays the Internet-to-Ethernet address translation tables used by the address resolution protocol. It can also be used to modify these tables.

rpcinfo

The *rpcinfo* command makes an RPC call to a host on the network and reports findings. There are three options associated with *rpcinfo*. '-p' is used to probe the specified host and print a list of all registered RPC programs. '-u' and '-t' options are used to actually make a call to an RPC program using either UDP or TCP, respectively, and report the response.

ifconfig

Ifconfig is used to check or modify the network interfaces configuration. The network interfaces configuration is normally set at boot by the initialization scripts. The configuration can be changed after boot with the *ifconfig* command. 'la0' is the ethernet network interface for the ESV.

Figure 10 - 2 Examples of Networking Tools

ping pebbles

```

PING pebbles (130.187.85.70): 56 data bytes
64 bytes from 130.187.85.70: icmp_seq=0. time=10. ms
64 bytes from 130.187.85.70: icmp_seq=1. time=0. ms
64 bytes from 130.187.85.70: icmp_seq=2. time=0. ms
64 bytes from 130.187.85.70: icmp_seq=3. time=0. ms
64 bytes from 130.187.85.70: icmp_seq=4. time=0. ms
64 bytes from 130.187.85.70: icmp_seq=5. time=0. ms
64 bytes from 130.187.85.70: icmp_seq=6. time=0. ms

```

spray pebbles

```

sending 1162 packets of lnth 86 to pebbles ...
no packets dropped by pebbles
99 packets/sec, 8577 bytes/sec

```

spray -c 5000 pebbles

```

sending 5000 packets of lnth 86 to pebbles ...
1 packets (0.020%) dropped by pebbles
99 packets/sec, 8596 bytes/sec

```

netstat -i

Name	Mtu	Network	Address	Ipkts	Ierrs	Opkts	Oerrs	Collis
la0	1500	eands	druid	2090686	25	605508	0	0
lo0	1536	loopback	localhost	11460	0	11460	0	0

netstat -r

Routing Tables	Gateway	Flags	Refcnt	Use	Interface
Directory					
localhost	localhost	UH	0	359	lo0
130.186	caesar	UG	0	0	la0
eands	druid	U	30	613208	la0
130.181	caesar	UG	0	0	la0

netstat -I la0 5

input (la0)		output		input (Total)		output			
packets	errs	packets	errs	colls	packets	errs	packets	errs	colls
2091136	25	605744	0	0	2102599	25	617207	0	0
37	0	10	0	0	37	0	10	0	0
33	0	13	0	0	33	0	13	0	0
32	0	14	0	0	32	0	14	0	0
39	0	14	0	0	39	0	14	0	0
38	0	12	0	0	38	0	12	0	0

who

```
brturner   ttyq0    Feb 21 09:42
bsmith     ttyq1    Feb 21 07:49
ccarey     ttyq2    Feb 21 09:59
csupport   ttyq3    Feb 21 10:21
```

whodo

Wed Feb 21 14:23:11 1990 druid

```
ttyq0  brturner  9:42
ttyq0  18045    0:01  csh
ttyq0  18443    0:00  whodo

ttyq1  bsmith   7:49
ttyq1  17455    0:01  csh
ttyq1  17656    0:25  twm

ttyq2  ccarey   9:59
ttyq2  18063    0:01  csh

ttyq3  csupport 10:27
ttyq3  18137    0:01  csh
```

/usr/etc/factaddr

Ehernet Factory Address: 08-00-57-08-00-70

arp -s backhoe 8:0:57:8:0:70

backhoe (130.187.85.251) at 8:0:57:8:0:70 permanent

arp backhoe

```
pebbles (130.187.85.251) at 8:0:57:8:0:70
      {
      |
      v
Internet Address   Ethernet Address
```

rpcinfo -p druid

program	vers	proto	port	
100005	1	udp	1025	mountd
100001	1	udp	1027	rstatd
100001	2	udp	1027	rstatd
100008	1	udp	1030	walld
100002	1	udp	1032	rusersd
100012	1	udp	1034	sprayd
100003	1	udp	2049	nfs

rpcinfo -u druid 100005

program 100005 version 1 ready and waiting

ifconfig la0

la0: flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>
inet 130.187.85.250 netmask ffff0000 broadcast130.187.255.255

ifconfig la0 down**ifconfig la0**

la0: flags=63<BROADCAST,NOTRAILERS>
inet 130.187.85.250 netmask ffff0000 broadcast130.187.255.255

ifconfig la0 up

Network Addressing

Ethernet Address

Ethernet address is a 48 bit hardware address assigned to the ESV during manufacture. These addresses are administered by Xerox Corporation, who originally designed and patented Ethernet. The Ethernet number is assigned to the system as its' hardware address and should remain with that system. The Ethernet address is stored within the NVRAM on the CPU board. The address can be verified from the boot monitor prompt with the *printenv* command. The Ethernet Address must remain with the system. Since the address is stored on the CPU card, exchange of the CPU card requires either the NVRAM chip be transferred from the original CPU to the new CPU or the NVRAM's ethernet entry be changed with the *setenv* command after installation of the new CPU. For reference the Ethernet Address is posted on the rear of the ESV cabinet. Addresses are usually displayed as a hexadecimal number separated by 5 colons.

An example of an Ethernet address is:

8:0:57:8:0:6b

/etc/ethers is the file used to reference between an Ethernet address and host name.

Internet Address

An Internet address is a 32 bit number which represents both your local area network and your host within it. There are three classes of addresses: A, B, and C. The addresses used here at E&S are class B addresses.

The first one two or three bits of an address determine it's class. For example:

- If the first bit of the 32 bit address is a zero the address is a class A.
- If the first bit is a 1 and the second a 0 the address is class B
- If the first two bits are ones with the third a zero it is a class C address.

Examples:

Class A **126.1.2.3**

Class B **191.187.85.20**

Class C **192.6.10.120**

Each number separated by a decimal point is equal to the binary value of its respective eight bit byte of the 32 bit address. The first three digits for a class A network ranges from 1 to 127, from 128 to 191 for a class B, and 192 to 223 for a class C.

The number of bytes comprising the network and host portions of the address differs according to class. The address byte layout by class is as follows:

Class A address: net.host.host.host

Class B address: net.net.host.host

Class C address: net.net.net.host

/etc/networks is the file that can reference between the Internet network number and network name.

The Internet address for all hosts on the network is kept in the */etc/hosts* file. This file is a cross reference between Internet address and host name. Many programs using TCP/IP protocols and some BSD programs use this file. Because of this file the user isn't required to remember Internet addresses. He only needs to know the name of the host.

The following are sample lines from a */etc/hosts* file:

```
130.187.85.38 hooch
130.187.85.39 lacerta
130.187.85.43 tuna
130.187.85.44 slowworm
130.187.85.45 skink
```

/etc/networks file is used to cross reference between Internet network name and number.

This is a sample networks file:

```
loopback 127
sun-ether 192.9.200
eands 130.187
arpanet 10
ucb-ether 46
```

If you wish to connect a local network to a larger network, you must apply

for an address through the agency that administers that network. For example, if you wish to connect to the Internet, you should call the Network Information Center at Stanford University. They are responsible for all Internet network assignments.

/etc/local_hostname is the file that contains your hostname. It is one of the first files you will need to edit at installation. The hostname is normally chosen by the owner or the principal user of the machine. The file should, on its first line contain the name chosen for this machine. It's important that there be no spaces or blank lines preceding this name and the name needs to be unique to the local network.

Example:

druid

netmask 0xff000000 broadcast 192.255.255.255

Note: Networks conforming to the BSD 4.2 protocol use zeros instead of "255"s when specifying the broadcast address.

Network File System

The purpose of NFS is to provide a machine independent network service which allows transparent sharing of file systems or directories among hosts on a network.

Server/Client

The NFS server is the machine that exports file systems to make them available to share with NFS clients. One machine can be both a *server* and a *client* with respect to different file systems.

The server reads or writes files in response to client requests. The client mounts the shared file system or directory exported by the server to an empty directory local to the client. The server uses the */etc/exports* file to specify which file systems or directories to export and who they should be available to.

Daemons

There are two *daemons*, */etc/nfsd4* and */etc/biod*, that must be running on the server to facilitate the "server/client" interaction. The *ps* command may be used to verify the presence of these daemons.

/etc/fstab

The client gains access to shared file systems or directories manually by using the *mount* command or automatically by modifying his */etc/fstab* file. With the shared filesystem added to a clients */etc/fstab*, that filesystem will be mounted automatically each time the workstation is booted.

There are several options that can be added to */etc/fstab*. Consult the *man* page or the *mips* reference manuals prior to modifying this table for an explanation of the fields and options available.

Sample *fstab*:

<i>/dev/root</i>	<i>/</i>	<i>ffs</i>	<i>rwt</i>	<i>0 0</i>
<i>/dev/usr</i>	<i>/usr</i>	<i>ffs</i>	<i>rw</i>	<i>0 0</i>
<i>bambam:/usr1</i>	<i>/usr1</i>	<i>nfs</i>	<i>rw,soft</i>	<i>0 0</i>

NFS Client

Any shared file system can be mounted on your machine as long the server can be reached over the network and the file system has been exported to your machine by the serving machine.

Manually Mounting a File System

To manually mount a file system, become *super user* on the machine and type:

```
mount host_name:/filesystem /mountpoint
```

A file system mount may be done with either the hard or soft option specified. A hard mount causes the client to continue to call the server until it responds. If the server is down a hard mount causes the client to wait indefinitely for the server to respond. A hard mount is the default mount. A soft mount allows the process to fail if the server doesn't respond.

mount - used to manually mount shared filesystems. If used without options will display currently mounted filesystems.

NFS Servers

To *export* a file system:

Become *super user* and add the name of the file system you wish to export in the file `/etc/exports`. Export the filesystem with the command:

```
exportfs -a
```

Be sure the mount and nfs daemons are running. There are normally four nfs daemons running on a system to accommodate multiple NFS RPC calls. The same is true of the Block I/O daemon (biobd). Also make sure portmap is running. This is done with the *ps* command.

If the NFS daemons aren't running, they can be started manually with:

```
/etc/init.d/nfs start
```

NFS Command summary

Remember to consult the man page for format or option questions on any command or file you're unfamiliar with.

There are several useful commands available that will help you with managing an NFS environment:

mount - used to manually mount shared filesystems. If used without options will display currently mounted filesystems.

umount - unmount shared filesystems.

showmount - lists all clients who have remotely mounted a filesystem from the specified host. The `-e` option prints a list of exported filesystems.

nfsstat - prints statistical information about NFS and RPC calling.

The `-s` option allows you to specify a server.

Network Information Services

Beginning with the 2.0 release of the ES/os, support for the MIPS Virtual Information Service is included. Mips VIS provides the capability to use multiple information services across the network. VIS is compatible with the SUN equivalent known as NIS or YP (Yellow Pages).

VIS/NIS is implemented by modifying the library routines which provide interfaces to these information databases. This significantly eases the task of application development because the application itself does not need to know about the different information sources. The following commands are supported with VIS:

yycat	yypasswd	yypbind
yymake	yypoll	yyserv
yypupdated	yypxfr	yypmatch
yypwhich	yypinit	yypasswdd
yypush	yypset	

By default, the OS distribution doesn't come with VIS enabled. Refer to the MIPS software release notes, RISCos 4.51 for a complete explanation of how to configure VIS.

Helpful Tips

On-line manual pages are available for all VIS commands. These commands start with the string yp. The command: **man -k yp** will give you a list of the VIS commands along with a few other unrelated commands.

/usr/bin/domainname can be used to set your domain name interactively. This must be done before building a database or starting a server.

/etc/local_domainname must contain your domain name for VIS to start up automatically during bootup.

/etc/rc2.d/S38nis_local must be added for VIS to start at boot. The contents of this file should be:

```
echo startup yypbind
nohup /etc/yypbind
/usr/bin/yypwhich
```

The sample file **/etc/vis.conf.sample** should be moved to **/etc/vis.conf**.

To use the NIS passwd and group facilities you must add: **+:0:0:::** as the last entry in the **/etc/passwd** file and **+** as the last entry of the **/etc/group** file.

To test VIS on a network where an NIS/VIS server is already running, execute the following commands as root:

```
/usr/bin/domainname <your domain name>
```

```
mv /etc/vis.conf.sample /etc/vis.conf
```

```
/etc/ybind
```

```
/usr/bin/ypwhich
```

If everything is correct, the following command will cause a password file to be output from the NIS/VIS server:

```
ypcat passwd
```

ESDnet Overview

ESDnet is a communications package that provides access between the ESV Workstation and VAX systems running a DECnet Ethernet local area network. With ESDnet you can:

- Copy files to or from DECnet nodes
- List a directory residing on a DECnet node
- Rename, delete or print files on a DECnet node
- Submit to batch files on a DECnet node
- Exchange mail with other DECnet nodes
- Similarly, from other DECnet nodes, such as a VAX, you can perform all of the above operations on the ESV Workstation.

DECnet is sold in two forms: *End-Node* and *Full-Function*. The major difference between these two forms is routing, which is the ability to read packets at an intermediate node and pass on or route the packets to their destination node. Routing is only available in Full-Function DECnet and currently not available in ESDnet.

ESDnet does not currently support diskless workstation nodes.

ES/Dnet allows you to transfer files between systems or to login to the remote system (DECnet node) from your ESV Workstation. This is accomplished by the following two ES/Dnet programs: *dap* and *sethost*.

- *dap* is a file manipulation program.
- *sethost* is a virtual terminal program that allows you to log in to a remote node.

In addition, you can send mail to other nodes using *dnamail*.

Using *dap*

dap is a file manipulation program that allows the ESV Workstation and the DECnet node to use each others' file systems. The commands include file transfer, file management, general utility, and batch submittal commands.

File Transfer and Management Commands

- The **copy** command is a bidirectional command that allows you to copy files between the ESV Workstation and a remote DECnet node. With the **copy** command, you specify a source and a destination in which either can be the remote file and the other the local file.
- The **delete** command deletes a file at the remote node.
- The **directory** command lists the contents of a remote directory.
- The **get** command retrieves a file from a remote DECnet node and stores the file on the local node.
- The **rename** command renames a file at the remote node.
- The **send** command stores a local file on the remote node.
- The **type** command displays a file to the screen.

General Utility Commands

- The **cd** command changes the local current working directory to the directory you specify.
- The **default** command sets the default access control for the remote node including node name, user name, password, and account name.
- The **exit** command exits **dap** and returns you to your previous prompt.
- The **help** command displays help text which includes descriptions of all of the **dap** commands.
- The **pwd** command lists the local current working directory.
- The **shell** command lets you enter shell commands, or leave **dap** and enter a shell.
- The **show** command displays the current default access control for the remote node.

Logging In to a Remote System

Login to a Remote System

The **sethost** command creates a virtual terminal connection from the ESV Workstation to a remote DECnet node. The remote node is specified with the node argument using the node name.

This remote login capability is similar to **rlogin**; however, **sethost** does not attempt to log you in at the remote node. Rather, once **sethost** has accessed the remote node, you then log in to the remote node just as you would if log-

ging on directly. In the case of a VAX system, the VAX's banner and login prompt are displayed, as in:

\$ sethost vaxnet

Username: **jones**

Password:(not echoed)

DIGITAL EQUIPMENT VAX SYSTEM

Last interactive login on Thursday, 25-AUG-1989 08:28

Last non-interactive login on Thursday, 25-AUG-1989 08:09

vaxnet>

You are now logged in to the remote VAX and your terminal is considered a VT100.

Logging off of the remote system returns you to the ESV Workstation's prompt.

Logging In From a Remote System

You can use the VMS **SET HOST** command to log in to your ESV Workstation from a DECnet node. The **sethost_server** accesses the ESV Workstation as a virtual terminal. For example:

Vax> **set host station**

Username: **dand**

Password:

\$

Using dnamail

You can send mail to network nodes using the **dnamail** command. The following example shows the user interaction for the **dnamail** command.

\$ dnamail

Node: **1.25**

To: **dand**

Subject: **Project meeting**

Waiting for connection to peer...

Enter your message below. Press <EOF> when complete or <ABORT> to quit.

Don't forget the project meeting today at 3:00.

<EOF>

Waiting for mail sending confirmation...

Mail Sent

\$

In specifying the node you can use a node name or number. The node number or name can be with or without the " : :".

Note <EOF> is the standard UNIX <EOF> which is generated by pressing the CONTROL and D keys simultaneously.

11. Disk and Tape Drive

Hard Disks and Tape Drive

Objectives:

- Understand when a disk problem is hardware or software related.
- Become capable of replacing or adding a hard disk or tape drive to the ESV Workstation.

Overview

An ESV Workstation cabinet is designed to house two SCSI 5.25' disks and a 1/4' 150 Mb SCSI tape drive. Hard disk options include 380Mbyte, 760Mbyte, and 1.2 Gbyte drives. The CPU's SCSI controller supports a maximum of seven SCSI devices.

Disk and Tape Drives

An ESV Workstation is designed to support two 5.25' hard disk drives and a cartridge tape drive. The disk and tape drives are connected to the CPU via the SCSI bus ribbon cable. The internal portion of the SCSI bus is a 50 pin ribbon cable with keyed connectors.

Figure 11-1 shows the SCSI bus layout. To function properly the bus must be terminated at each end of the physical bus. The CPU can provide termination for one end. The last device on the each end of the bus must provide the termination for that end or have an external terminator attached. A new drive from the manufacture will usually come supplied with the termination devices installed which will need to be removed unless the unit is being placed at the physical end of the bus.

The SCSI bus daisy chain configuration is shown in Figure 11-2.

Drive ID Selection

The drives logical address is selected by installing jumpers in the appropriate location on the Drive Select Header. This is accessible from the back of the drive. Normally, if the Workstation is shipped with a full compliment of drives, there will be drive 0 and drive 1. The tape drive will be drive 6.

The terminating resistor modules will be removed from all but the last drive in a fully configured workstation.

Figure 11-2 shows a typical diskdrive connector and jumper layout.

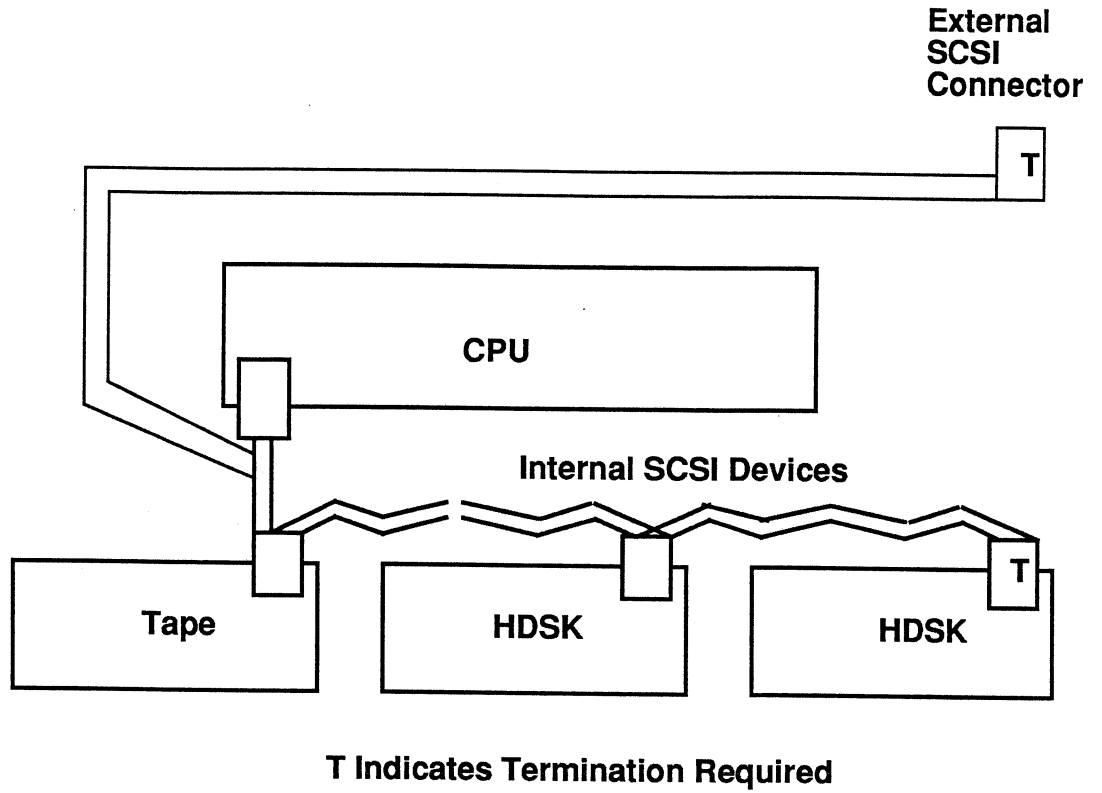


Figure 11 - 1 Interface Cabling Options

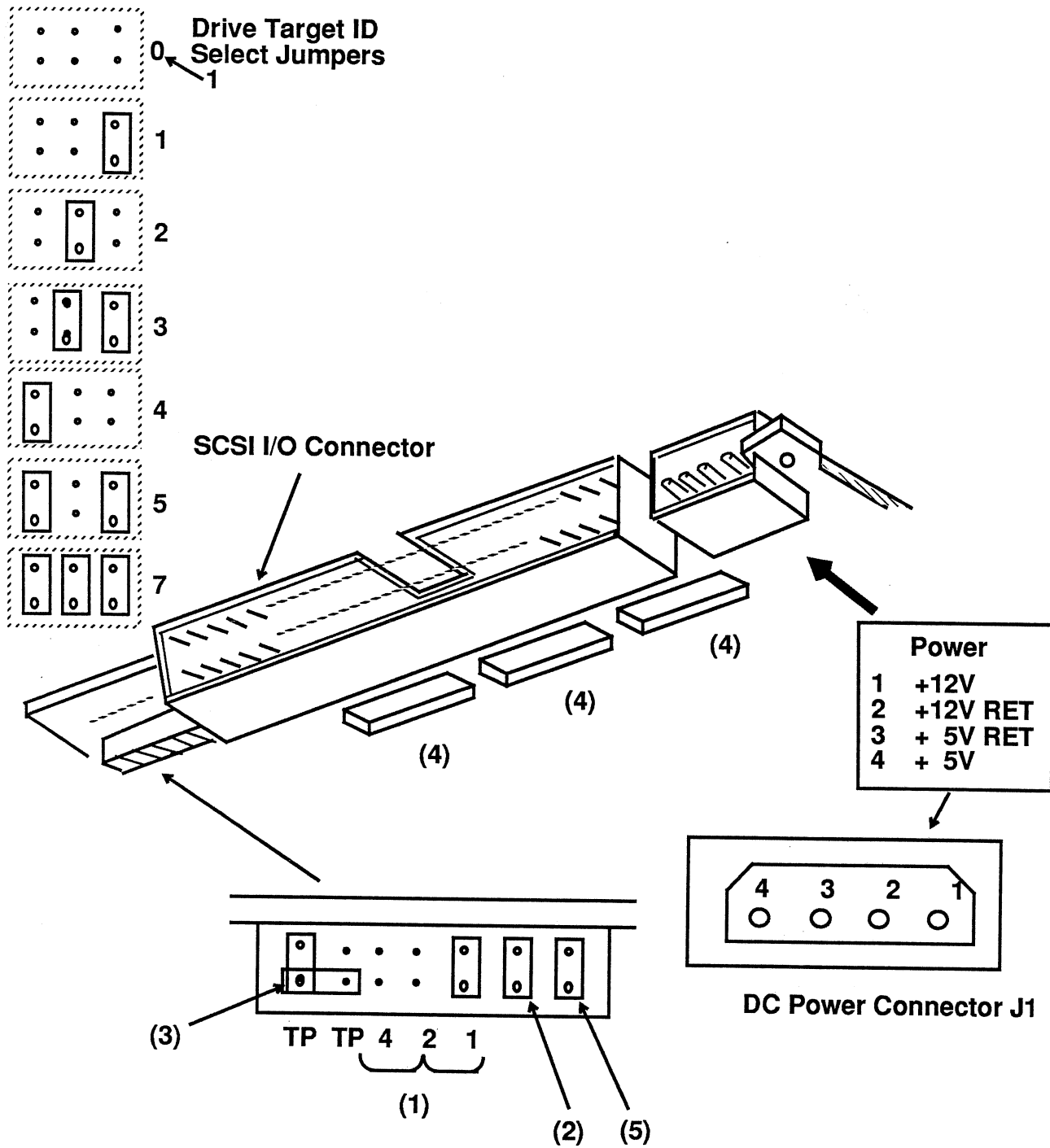
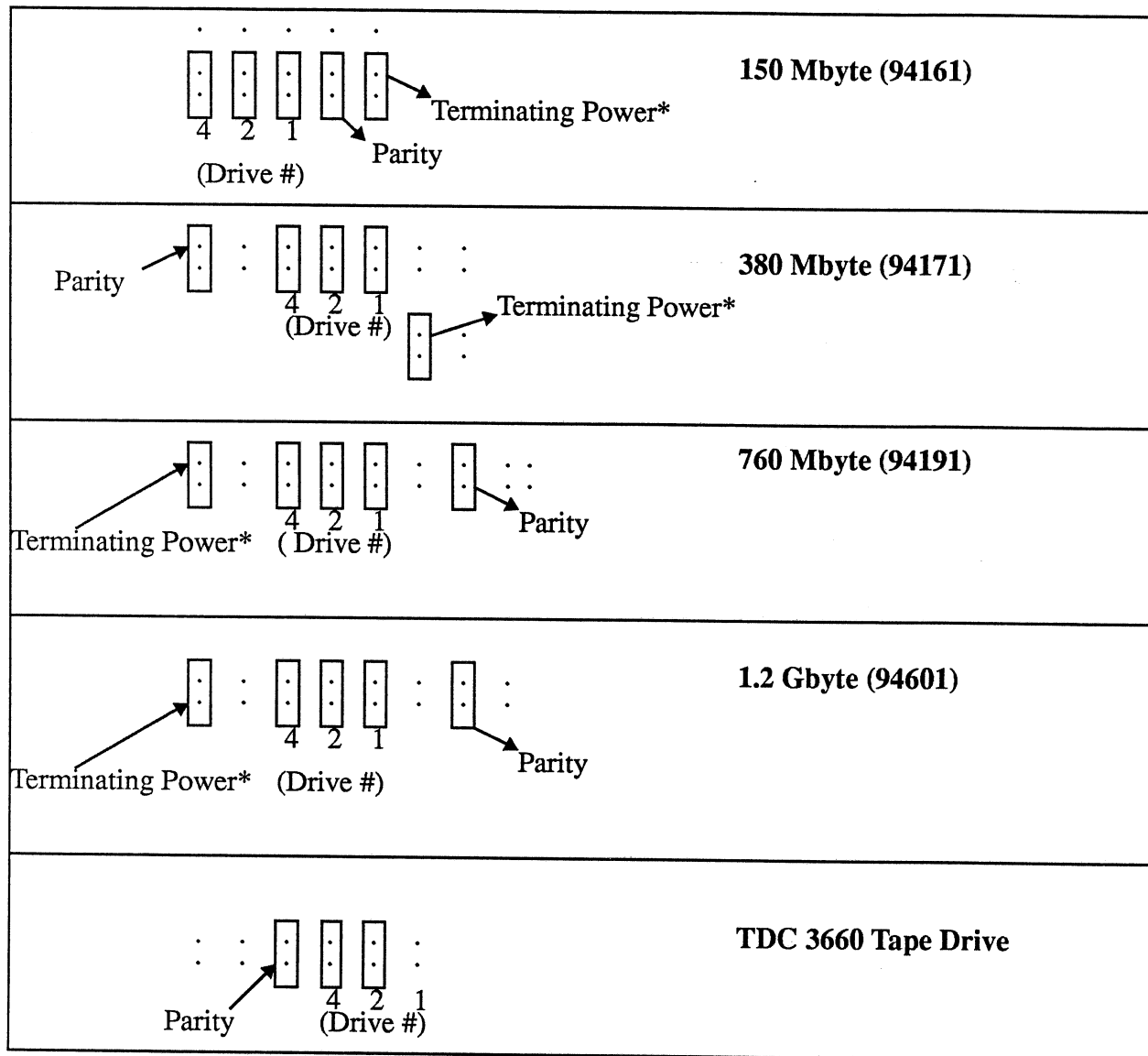


Figure 11 - 2 Drive ID and Option Select Header

Disk and Tape Drive Jumper Layouts



* Terminating Power Jumper is only needed on the drive physically located at the end of the bus. Terminating resistors should also be installed on the last drive.

Drive # is binary coded by jumper position, i.e., jumper in position 1 would be drive #1, no jumpers equals drive #0.

The cartridge tape drive should always be set as drive #6.

Parity jumper should always be installed.

Figure 11-3

Figure 11-2 Notes:

- 1) Drive ID is binary coded by jumper position (most significant bit on the left). A jumper in position 1 would be drive 1, no jumpers would be drive 0.
- 2) This is the motor start option. With a jumper installed here the drive will wait for a Start Unit Command from the CPU before starting the motor. On some of the newer disk drives this is a delay motor start option. With this jumper installed the starting of the motor is delayed some number of milliseconds multiplied by the target ID. It's unnecessary to use this option with the ESV Workstation. This option is used in situations where the power supply can not support the current required to start multiple drives simultaneously.
- 3) If the unit is not terminated the TP jumper need not be installed. If the unit is terminated, (terminating resistors installed), the TP jumper should be in the vertical position, allowing terminator power to be taken from drives' J1 power connector.
- 4) Removable SIP terminating resistors.
- 5) Jumper installed enables parity checking by the drive. (installed normally)

Figure 11-3 provides the jumper layout for all currently supported internal disk and tape drives.

Trouble Shooting Hard Drives

The CDC drives are designed as field replaceable units. Frequently, problems believed to be disk drive failures are not. It is important to recognize this before taking repair action on the disk drive.

The following disk power up sequence is provided to assist in identifying possible disk drive problems. Generally, if performance doesn't follow this sequence either the drive or the power supply should be questioned.

Power Up Sequence

- 1) When power is applied the LED glows steadily while the motor comes up to speed and the drive becomes ready.
- 2) Spindle motor reaches operating speed in approximately 10 seconds. No speed variations should be heard afterward.
- 3) Arm restraint solenoid releases producing an audible indication of its occurrence.
- 4) The drive performs up to 32 velocity adjustment seeks, as evidenced by head motion sounds.
- 5) The drive finds and stays at track zero, comes ready, makes no further noises, except as system commands demand.
- 6) If the drive has successfully powered up, the front panel LED goes dark and the drive is ready. The LED glows with each use, so boot up activity may appear as random LED flickering, as the drive is rapidly selected and deselected.
- 7) If during the power up sequence the internal controller detects a fault condition, the front panel LED flashes. Removing and reapplying power may or may not clear this fault indication. Power supply problems can also create this fault status.

Power Down Sequence:

- 1) Remove Power. The arm restraint solenoid sound should be heard within three seconds after the power is removed.
- 2) The dynamic break relay cuts in seconds later, producing an audible indication of its occurrence. Spindle rotation should stop in about 30 seconds.

Tape Drive

The cartridge tape drive is a Tandberg Data TDC 3660. The drive ID and option jumpers are located on the rear of the drive and are similar to the disk drive. The tape drive is designated as drive or SCSI device 6. The decal on the rear of the drive displays the layout of the jumpers and connectors.

- When a tape is installed into the drive and the door is closed, two things should happen. The green LED on the door should light and you should hear the drive spin the tape searching for the BOT (Beginning of Tape) marker. When the drive has located the BOT it will stop and should be ready to accept commands from the SCSI bus.
- Path name for the cartridge tape is `/dev/rmt/m0`. This is device name used in the tar, dump, or restore commands to specify the cartridge tape drive.
- The drive head should be cleaned regularly, to prevent errors while reading or writing tapes.

The following guidelines can be used to determine cleaning intervals:

<u>Tape Drive Usage</u>	<u>Cleaning Interval</u>
8 hours per day	Daily
Daily	Weekly
Weekly	Monthly

The Tandberg TDC 3660, which we are using in the ESV, is rated at 155 mbyte; but only under certain conditions. The conditions that affect the amount of data that can be stored on a tape are of course the length of the tape, the recording format, speed at which the tape is moving, and the quality of the tape. In order to obtain the maximum rated tape capacity of the TDC 3660 the recording needs to be done in the QIC 150 format. This is the default format for the workstation drive and tapes will be recorded in this format if a quality tape is used.

If the drive detects errors while trying to record with the QIC 150 format it will automatically step down to the QIC 120 format. The difference between these two formats is the number of tracks placed on the tape. QIC 150 puts 18 tracks on the 1/4 inch tape, QIC 120 only 15. This, of course has an effect on the capacity of the tape.

A 600 foot tape recorded in QIC 120 can only hold 125 mbytes. Tandberg only recommends one tape for achieving its maximum rating. The 3M brand DC-600 XTD or as it is now called the DC-6150 is the tape they recommend. Other tapes can be used, but are not recommended by the tape drive vendor. Tapes longer than 600 feet are thinner and are more subject to stretching.

Lower density tapes, such as the DC-600 A will work fine but the drive will most likely step down to the lower capacity QIC 120 format.

You should also consider the utility used to create the tape. There are varying amounts of overhead placed on the tape based on the tape utility used.

Some tape vendors place a mbyte rating on the tape label, but they usually fail to tell you what format, utility or drive they achieved that rating with.

When using the **dump.ffs** command for creating backup tapes on the ESV the default density and size work. But they don't make efficient use of a 600 foot tape written in QIC 150 format. The default **dump.ffs** density is 1600 bits per inch with a length of 2300 feet. These defaults were setup to work best with 9 track reel-to-reel media. The manpage for **dump.ffs** Version 1.2 of the operating system states, "The default density and length (1600 bpi and 2300 ft) seem to work very well for these tapes". This was written assuming the cartridge tape drive was using the QIC 24 format which can only put about 60 mbyte on a 600 ft tape.

"For QIC-24 (Quarter-inch cartridge) tapes, specify a density of 800 and use the default length. For QIC-120 tapes, specify a density of 23000 and a length of 570."

The manpage does not say anything about QIC-150. But, by using the values suggested for QIC-120 and increasing the density to 28750 to match the increased capacity of QIC-150 format you will come close to gaining the most efficient use of a 600 ft tape.

Example:

```
dump.ffs 0udsf 28750 570 /dev/rmt/m0 file_system
```

There are likely an infinite number of density/size combinations that will work giving you varying tape efficiency. But this combination provides better than 120mb/tape and does not appear to push the limits of the Tandberg drive capabilities.

Formatting and Partitioning

Formatting a disk means establishing addressable areas on the medium. Partitioning means assigning logical units to addressable areas.

Formatting

SCSI disks are formatted at the factory and do not need to be formatted. The disk format that is performed by the factory is more rigorous and finds more defects than the mips format program is capable of detecting. Therefore, it is recommended that disks are not formatted unless it is believed that there is something physically wrong with the disk.

The first block of the disk is reserved for data having to do with that specific disk and is called the volume header. The volume header contains device parameter information, the partition table, and the volume directory. The format utility creates a volume header for the disk based on certain default values for recognized disks and user-input values for unrecognized disks.

At present all disks shipped with the ES/V will be recognized disks. Writing the volume header should be the only reason you would use format under normal circumstances. If the disk is suspected of having physically bad blocks, it can be scanned and mapped out with the format utility.

NOTE: SCANNING THE DISK DESTROYS ALL DATA ON THE DISK. Scans of a SCSI disk must be done on the entire disk. You should have a complete backup of the disk before scanning. Complete procedures for the format utility is located in chapter 4 of the MIPS System Administrators Guide.

Booting format from tape:

- 1) Install current version of the ES/os tape into the cartridge tape drive.
- 2) From the >> prompt: `boot -f tqis(, , 2) format.std`
(for 25Mhz systems)

`boot -f tqis(, , 3) format.r200`
(for 33/40Mhz systems)

Partitioning

Partitions on the SCSI hard disks are allocated in a standard arrangement. The following page shows the default partitioning scheme for the system disk. Partition 0 is the root partition, 1 is the swap partition, 6 is the usr partition and 7 is available as extra swap or as an extra partition.

For non-system disks any of these partitions (other than 8 & 10) can be used in any non-overlapping combination for creating filesystems. Partition 8 and its' allocated space must be preserved for the Volume Header information. Hard disks can be re-partitioned by the use of the format utility or the *dvhtool* command.

prvtoc

The `prvtoc` command is used to list volume header information to the screen. You can use this command to get information about various volumes quickly. The following example consists of a single `prvtoc` command and the systems response for the system disk or disk 0:

Figure 11-4 provides an example of the `prvtoc` command and it's output.

prtvtoc /dev/rdsk/isc0d0vh

/dev/rdsk/isc0d0vh (bootfile "/unix") partition map

Dimensions:

512 bytes/sector

840 sectors/track

1 tracks/cylinder

768 cylinders

765 accessible cylinders

Unallocated space:

Start Size

42000-39480

645120-642600

194880-192360

388080-346080

605640-563640

561120-519120

388080-193200

<u>Partition</u>	<u>Tag</u>	<u>Flags</u>	<u>First Sector</u>	<u>Sector Count</u>	<u>Mount Directory</u>
0	4	0	2520	39480	/
1	4	0	605640	39480	
2	4	0	2520	642600	
3	4	0	42000	563640	
4	4	0	42000	346080	
5	4	0	388080	173040	
6	4	0	42000	519120	/usr
7	4	0	561120	44520	
8	0	0	0	2520	
10	6	0	0	645120	
11	4	0	2520	385560	
12	4	0	388080	257040	
13	4	0	561120	84000	
14	4	0	2520	192360	
15	4	0	194880	193200	

Figure 11-4 prtvtoc Command

dvhtool

The dvhtool utility can be used to examine and modify hard disk volume header information. After entering the utility with the dvhtool command, you are given the option of seven commands: (read, vd, pt, dp, write, bootfile, or quit). You must first read the volume header (/dev/rdisk/isc0d#vh), #=disk, in order to examine the information already there.

The other commands: "vd" for the volume directory, "pt" for the partition table, "dp" for the device parameters, "write" to write new data, "bootfile" to replace the boot file (normally /unix), and "quit" to exit the utility.

Adding a Hard Disk

Installing the hardware for a second hard disk is quite easy and self explanatory. There are three things to watch out for and remember.

- Remove the SIP terminators unless it's on the end of the bus.
- Set the jumpers to select the proper drive target ID.
- Avoid pinching either the ribbon cable or the DC harness when mounting the drive to the pull-out tray.

Although the SCSI hard disk is formatted at the factory, a volume header is not written until the disk is added to the system. If a disk were to be added to the workstation after it leaves the factory or a faulty disk is replaced at the customer premise, the volume header needs to be written to the disk.

The easiest way to write the volume header is with the format utility. The format utility can be booted from tape.

Procedure:

- 1) To boot format from tape *(see page 11-9)

Once you have the format utility booted:

device: **dkis**

Enter the LUN number: **0**
Enter the target id: **#** (#= disk target ID)
Choose new drive parameters: **y** (if yes)

Modify device parameters: **n**

Dump partition table: **n**

Modify partition table:

Note: At this point you could modify the default partitions based on the drives application or use.

Perform format? **n**

Scanning destroys disk data, perform scan? **n**

SCSI defect list manipulation, when prompted choose one of

(list, add, delete, quit)

command? **quit**

writing volume header...

Note: With a new disk installed with a volume header written to it (assuming this isn't the system disk but a new extra disk) a new file system must be created on the disk. To do this you need to know which of the 15 partitions available on the disk header will be used. Partition 2 is the entire available disk, so for ease of explanation we'll assume partition 2 will be our new file system.

2) Enter ***newfs.ffs -s ##### /dev/dsk/isc0d1s2 ********

= size of partition 2 in 512 byte sectors

********* = hard disk model number, 94161,94171, etc

3) The file system is now created. Be sure to check that all is well before mounting and entrusting data to it by running the *fsck.ffs* program. Enter:
fsck.ffs /dev/dsk/isc0d1s2

The system output should look something like:

```
/dev/dsk/isc0d1s2
```

```
Last mounted on
```

```
Phase 1 - Check Blocks and Sizes
```

```
Phase 2 - Check Pathnames
```

```
Phase 3 - Check Connectivity
```

```
Phase 4 - Check Reference Counts
```

```
Phase 5 - Check Cyl groups
```

```
2 files, 9 used, xxxx free (xx frags, xxxx blocks,  
\ 0,1% fragmentation)
```

```
**** FILE SYSTEM WAS MODIFIED ****
```

Note: If *fsck* reports problems, remake the file system with *newfs* as above.

- 4) If *fsck* runs without problems the file system can be mounted at this point. Make a directory to mount the file system and then enter the mount command. Enter:

```
mkdir /newfilesystem
```

```
mount /dev/dsk/isc0d1s2 /newfilesystem
```

- 5) Finally, the new file system can be tested with the following commands.

```
ls /newfilesystem
```

```
mkdir /newfilesystem/newuser
```

```
cp /etc/stdcshrc /newfilesystem/newuser/.cshrc
```

```
ls -la /newfilesystem/*
```

- 6) To have the new file system automatically mounted at boot time, add the following entry in */etc/fstab*:
- ```
/dev/dsk/isc0d1s2 /newfilesystem ffs rw,soft 0 0
```

---

## Backup and Restore

After system installation the hard disks should be backed up at regular intervals on cartridge tape or a remote file system. The data contained on a hard disk is only as good as its back up!

### Complete Backup

To perform a complete backup of a filesystem: take the system to single user mode, `init s`. Enter the `dump.ffs` command:

```
dump.ffs 0udsf 28750 570 /dev/rmt/m0 /dev/dsk/isc0d#s*
```

# = disk number, 0 for system disk

\* = the partition number, 0 for root, 6 for usr, etc....

### Incremental Backup

Incremental backups are performed in a similar manner:

```
Enter: dump.ffs 1udsf /dev/rmt/m0 /dev/dsk/isc0d#s*
```

With this message, "1" causes a dump of files that have changed since the lower level 0 dump occurred. The file `/etc/dumpdates` is used by the system to record when the last dump took place and the level.

### Restore

The `restore.ffs` command is used to restore lost files or file systems from tapes created by the `dump.ffs` command.

- 1) Login as `/root`.
- 2) Take the system to single-user mode. This example shows a complete restore of `/usr` from a dump tape.

```
ls /usr
```

```
cd /usr
```

```
restore.ffs rvf /dev/rmt/m0
```

- 3) Verify that the files were restored to `/usr` with an `ls` command.
- 4) Return the system to multi-user.

Restore can also be used in an interactive mode with the `-i` option. This is very useful if you need only restore some of the files from a dump tape. The interactive mode allows you to select a file or files from a dump tape to be restored. See the *man* page for a complete description.

## Appendix 1. Adding Swap Space to Your ESV

The amount of swap space your ESV needs is partially based on the amount of system memory (RAM) installed on the CPU. The *system disk*, disk 0, is formatted and partitioned at the factory with a default of approximately 48 mbytes of swap space. This amount is usually sufficient for ESV systems configured with 16 mbytes of system memory or less. Systems configured with 24 mbytes or more of system memory and running applications with a large demand for virtual memory may want to consider adding additional swap space to their systems. The rule of thumb guide we have found to work well is:

Swap=2 x System Memory + 50mb

To determine the amount of swap you currently have allocated:

```
swap -l
path dev swaplo blocks free
/dev/dsk/isc0d0s1 16,1 0 65536 64233
/dev/dsk/isc0d0s7 16,7 0 27648 26296
```

This shows the default swap configuration, using disk 0 partitions 1 and 7. The total of these partitions gives you 93,184 disk blocks of swap. One disk block equals 512 bytes. So, 93,184 blocks equals 48,710,208 bytes of swap.

Increasing swap, especially on a system with a single disk drive, is not a simple procedure. It may, in some cases, be more advantageous to add a second disk drive and create a swap partition on it. With a single disk drive the only filesystem of any size with free area is normally /usr. If /usr is only say 50% used you may want to take disk space from /usr and add it to swap. To find out what your disk usage is:



```
df
```

| Filesystem | Type | kbytes | use    | avail  | %use | Mounted on |
|------------|------|--------|--------|--------|------|------------|
| /dev/root  | ffs  | 21583  | 13259  | 8324   | 61%  | /          |
| /dev/usr   | ffs  | 589163 | 324973 | 264190 | 55%  | /usr       |

In the above example /usr is 55% used and has 264,190 kbytes available. You may want to sacrifice available space in the /usr partition to increase your swap area. With that in mind, the following procedure should function as a guide for accomplishing this task.

### Backup of the /usr Partition

Since we are going to tamper with the size of the /usr partition, the first thing we will want to do is create a backup copy of /usr on tape. This will require 2 blank 600 ft. cartridge tapes.

```
dump.ffs 0udsf 28750 570 /dev/rmt/m0 /usr
```

The dump will take 30 min to an hour depending on the amount of data in your /usr filesystem. While the dump is taking place you need to determine the number of blocks/sectors you wish to reallocate from the /usr file system, partition 6, to swap, partition 7.

```
prtvtoc /dev/rdisk/isc0d0vh
```

will give you a listing of the partitions and their sector counts.

```
df
```

will tell you the amount of available memory, in kbytes, the /usr filesystem has.

The optimum amount of swap is determined by doubling the amount of system memory and adding 50 mbytes. Assuming the system is using the default amount of swap, about 49 mbytes, you can just double the amount of system memory installed on the CPU and use that as the amount to increase your swap space. So, for a system with 32mbytes of system memory increase swap by 64 mbytes, a system with 64mbytes of system memory increase swap by 128 mbytes, etc...

Make sure /usr has enough available space to accommodate the amount you want to take for swap. Convert the number of bytes you wish to increase swap into a sector/block count by dividing it by 512. For example: If you wish to increase swap by 64mbytes, 64,000,000 divided by 512 equals 125,000.

**Note:** Sectors and Blocks are synonymous with respect to our SCSI disk drives.

Enter the swap increase you've calculated in sectors here: #blks \_\_\_\_\_. This will be the number of blocks you will need to remove from partition 6 and add to partition 7. For an example I will use 125,000 as #blks.

OK, so now you should know the number of blocks you wish to move from /usr to swap and hopefully the dump has completed successfully. We should now be ready to proceed with the swap increase.

```
shutdown -y -i0 -g0
```

```
{After the shutdown has completed, boot to single user.}
```

```
>> boot
```

```
sash: boot -f dkis0/unix initarg=s
```

```
{After the system has completed booting to single user.}
```

```
dvhtool
```

Command? (read, vd, pt, dp, write, bootfile, or quit): read  
 Volume? /dev/dsk/isc0d0vh

Command? (read, vd, pt, dp, write, bootfile, or quit): pt

current contents:

| <u>part</u> | <u>n blks</u> | <u>1st blk</u> | <u>type</u>                                 |
|-------------|---------------|----------------|---------------------------------------------|
| 0:          | 46080         | 2048           | bsd                                         |
| 1:          | 65536         | 1230080        | bsd                                         |
| 2:          | 1293568       | 2048           | bsd                                         |
| 3:          | 1181952       | 48128          | bsd                                         |
| 4:          | 769536        | 48128          | bsd                                         |
| 5:          | 384768        | 817664         | bsd                                         |
| 6:          | 1154304       | 48128          | bsd<--current values for partition 6 (/usr) |
| 7:          | 27648         | 1202432        | bsd<--current values for partition 7        |
| 8:          | 2048          | 0              | volhdr                                      |
| 10:         | 1295616       | 0              | volume                                      |
| 11:         | 815616        | 2048           | bsd                                         |
| 12:         | 477952        | 817664         | bsd                                         |
| 13:         | 93184         | 1202432        | bsd                                         |
| 14:         | 407808        | 2048           | bsd                                         |
| 15:         | 407808        | 409856         | bsd                                         |

## Adding Swap Space

Command? (part nblks 1stblk type, or 1) {Your input here should be in the form of: 6, current n\_bkls minus #blks you entered above, current 1st\_blk, bsd }

Example: 6 1029304 48128 bsd

Command? (part nblks 1stblk type, or 1) {Your input here should be in the form of: 7, current nblks plus #blks you entered above, current 1st\_blk minus #blks you entered above, bsd }

Example: 7 152648 1077432 bsd

Command? (part nblks 1stblk type, or l) l

current contents:

| <u>part</u> | <u>n blks</u> | <u>1st blk</u> | <u>type</u>           |
|-------------|---------------|----------------|-----------------------|
| 0:          | 46080         | 2048           | bsd                   |
| 1:          | 65536         | 1230080        | bsd                   |
| 2:          | 1293568       | 2048           | bsd                   |
| 3:          | 1181952       | 48128          | bsd                   |
| 4:          | 769536        | 48128          | bsd                   |
| 5:          | 384768        | 817664         | bsd                   |
| 6:          | 1029304       | 48128          | bsd<---verify change  |
| 7:          | 152648        | 1077432        | bsd<---verify changes |
| 8:          | 2048          | 0              | volhdr                |
| 10:         | 1295616       | 0              | volume                |
| 11:         | 815616        | 2048           | bsd                   |
| 12:         | 477952        | 817664         | bsd                   |
| 13:         | 93184         | 1202432        | bsd                   |
| 14:         | 407808        | 2048           | bsd                   |
| 15:         | 407808        | 409856         | bsd                   |

Command? (part nblks 1stblk type, or l) <CR>

Command? (read, vd, pt, dp, write, bootfile, or quit): write

Warning: partition 7 not cylinder aligned {disregard warning messages here}

Volume? (/dev/dsk/isc0d0vh) <CR>

Command? (read, vd, pt, dp, write, bootfile, or quit): quit

*{Now that you have resized partitions, recreate the /usr filesystem}*

```
newfs.ffs -s ##### /dev/dsk/isc0d0s6 *****
```

*{where ##### equals the new size of partition 6 in blocks and \*\*\*\*\* equals the model number of the drive 94171, 94191, 94601...}*

```
fsck.ffs /dev/dsk/isc0d0s6
```

```
mount /dev/dsk/isc0d0s6 /usr
```

```
cd /usr
```

*{Insert tape #1 from your dump of /usr into the tape drive.}*

```
restore.ffs -rvf /dev/rmt/m0
```

*{Restore takes about the same amount of time as the dump did.}*

```
/etc/shutdown -y -i0 -g0
```

*{After shutdown is complete, reboot normally, and verify additional swap space.}*

## Appendix 2. Disk Models

This is a table that may be helpful in situations where you need to know the model number of the disk drive you are working with. (e.g. newfs.ffs) To use this table you must know the cylinder count for the drive in question. To find this number use the following command: `prtvtoc /dev/rdisk/isc0d#vh`. Where # equals the SCSI drive ID. Root permission is required. From the output of the prtvtoc command find the number of cylinders for your device under 'Dimensions:'. Use that number in the following table to determine your drives' model number.

| Cylinders     | Model Number | Unformatted Capacity |
|---------------|--------------|----------------------|
| 1189          | 94161        | 150Mb                |
| 1325 or 2501* | 94171        | 380Mb                |
| 5061          | 94191        | 760Mb                |
| 7917          | 94601        | 1.2Gb                |

\* Possibility for 2 different cylinder counts for the same drive is due to a difference in versions of the format utility used to write the volume header. Changes in other dimensions make up for the difference in cylinder count.

## Appendix 3. Booting from Tape

### Booting to Single User/Miniroot with the ES/os 2.2 Tape

The following procedure allows you to boot to single user using swap partition 1. This may be helpful if the normal boot procedure (**auto**) fails and you wish to start the operating system from tape. Once you have booted from tape corrections can be made to the original root partition that will allow you to boot normally. This procedure does not overwrite any existing data.

```
>> init
>> boot -f tqis(,2)sash.std {note: replace sash.std with sash.r200 if 33 or 40 Mhz}
sash: cp -b 16k tqis(,4) dkis(,1)
sash: boot -f dkis(,6)unix.std root=isc0d0s1
{note: for 33 or 40 Mhz.. boot -f dkis(,8)unix.r200 root=isc0d0s1}
#
```

Now that you are at the pound sign prompt the original root partition, partition 0, can be mounted:

```
mkdir /oldroot
fsck.ffs /dev/dsk/isc0d0s0
mount /dev/dsk/isc0d0s0 /oldroot
```

With the original root partition mounted you can now **cd** into it to make any corrections you may need to the file preventing you from booting normally. However, if you wish to use the **vi** editor, you will also need to mount the original **usr** partition, **/dev/dsk/isc0d0s6**. The **vi** editor is located under **/usr/bin**.



## Appendix 4. System Disk Upgrade

### System Disk Upgrade

This procedure should help the trained FE through the process of upgrading a customers ESV from a 320mb to a 760mb system disk. ES/os 1.3 requires, in some cases, more system disk capacity than the 320mb drive can provide. This was written assuming the ESV contains only one 320mb disk drive.

Follow this procedure with care. Remember, the existing disk drive contains data that is very valuable to the customer. If unexpected errors or messages occur, stop and analyze or call dispatch for assistance.

The upgrade kit should contain: 1 Model 94191 SCSI Disk Drive, 4 mounting screws, 1 drive select jumper, 3 cartridge tapes 600 ft, and following instructions.

You will need *root* privilege to complete this procedure, so arrangements should be made with the customer for either root password or their assistance at login.

During the procedure characters in **bold** are characters or commands to be entered. Lines beginning with "**#**" are command lines. Italic lines enclosed with {} are notes.

As a precaution, perform a level 0 backup of both the root and usr filesystems.

**dump .ffs 0udsf 28750 570 /dev/rmt/m0 / ....creates root backup on 1 tape.**

**dump .ffs 0udsf 28750 570 /dev/rmt/m0 /usr ..... /usr backup on 2 tapes.**

Label the tapes for future reference.

Login as root, verify that no one else is logged in (**who**), shut the ESV down.

```
(shutdown -y -i0 -g0).
```

After the shutdown is complete power the machine off.

Remove the disk tray by removing the front panel screws, the upper ribbon cable from the CPU, slide the tray out far enough to disconnect the power connector, and then the rest of the way out and place the tray on top of the ESV, taking care not to scratch the finish. (Packing material or cardboard could be used to protect the top of the ESV.)

Unpack the Model 94191 drive. Remove it's SIP terminators and set the drive ID to 1. (See the attached product specification sheet or the Installation Manual packed with the drive.)

Install the 94191 on the tray between the tape drive and the existing 94171 using the 4 screws and connecting the SCSI ribbon and the DC connectors. Do not put the tray back into the ESV.

With the tray on top of the ESV, connect the SCSI ribbon to the CPU and the power connector from the tray to the ESV. (You may need to cut a tie wrap to free the power cable from the ESV enough to make this connection.)

Restore power to the ESV.

Install ES/0s 1.3 tape into the tape drive.

```
>> boot -f tqis(,2)format.std
```

```
name of device? dkis
```

```
LUN number? 0
```

```
target id? 1
```

```
dk:1 can't read volume header
```

```
vendor: cdc
product: 94191-15
Disk has block caching enabled
Drive has 5061 cylinders
256 sectors per cylinder
234 sectors left over

dump device parameters (y if yes)? <CR>
modify device parameters (y if yes)? <CR>

dump partition table (y if yes)? <CR>
modify partition table (y if yes)? <CR>

formatting destroys ALL SCSI disk data, perform format (y if yes)? <CR>
formatting wasn't done, perform scan anyway (y if yes)? <CR>

SCSI defect list manipulation, when prompted choose one of (list, add, delete, quit)
command? quit

writing volume header...
exit(0) called

>> auto

Remove the tape from the drive and wait for the system to boot.

Login as root.

{ Verify the volume header on the new drive and note the sector count for partitions 0 and 6.}
```

# prtvtoc /dev/rdsk/isc0d1vh

\* /dev/rdsk/isc0d1vh (bootfile "/unix") partition map

\* Dimensions:

- \* 512 bytes/sector
- \* 256 sectors/track
- \* 1 tracks/cylinder
- \* 5061 cylinders
- \* 5053 accessible cylinders

\* Unallocated space:

|  | Start   | Size     |
|--|---------|----------|
|  | 48128   | -46080   |
|  | 1295616 | -1293568 |
|  | 409856  | -407808  |

| Partition | Tag | Flags | First Sector | Sector Count | Mount Directory         |
|-----------|-----|-------|--------------|--------------|-------------------------|
| 0         | 4   | 0     | 2048         | 46080        | <---size of partition 0 |
| 1         | 4   | 0     | 574720       | 65536        |                         |
| 2         | 4   | 0     | 2048         | 638208       |                         |
| 3         | 4   | 0     | 48128        | 526592       |                         |
| 4         | 4   | 0     | 48128        | 332544       |                         |
| 5         | 4   | 0     | 817664       | 166400       |                         |
| 6         | 4   | 0     | 48128        | 498944       | <---size of partition 6 |
| 7         | 4   | 0     | 547072       | 27648        |                         |
| 8         | 0   | 0     | 0            | 2048         |                         |
| 10        | 6   | 0     | 0            | 640256       |                         |
| 11        | 4   | 0     | 2048         | 378624       |                         |
| 12        | 4   | 0     | 380672       | 259584       |                         |
| 13        | 4   | 0     | 547072       | 93184        |                         |
| 14        | 4   | 0     | 2048         | 189184       |                         |
| 15        | 4   | 0     | 191232       | 189184       |                         |

*{This creates a filesystem on partition 0 of the new disk. 46080 should match the size of partition 0 from the previous prtvtoc command.}*

**# newfs.ffs -s 46080 /dev/dsk/isc0d1s0 94191**

/dev/dsk/isc0d1s0:46080 sectors in 180 cylinders of 1 tracks, 256 sectors

23.6Mb in 12 cyl groups (16 c/g, 2.10Mb/g, 832 i/g)

super-block backups (for fsck -b#) at:

32, 4128, 8224, 12320, 16416, 20512, 24608, 28704, 32800, 36896,  
40992, 45088,

rotational delay between contiguous blocks changes from 7ms to 0ms

*{This creates a filesystem on partition 6 of the new disk. 498944 should match the size of partition 6 from the prtvtoc command.}*

**# newfs.ffs -s 498944 /dev/dsk/isc0d1s6 94191**

/dev/dsk/isc0d1s6:1154304 sectors in 4509 cylinders of 1 tracks, 256 sectors

591.0Mb in 282 cyl groups (16 c/g, 2.10Mb/g, 896 i/g)

super-block backups (for fsck -b#) at:

32, 4128, 8224, 12320, 16416, 20512, 24608, 28704, 32800, 36896,  
40992, 45088, 49184, 53280, 57376, 61472, 65568, 69664, 73760, 77856,  
81952, 86048, 90144, 94240, 98336, 102432, 106528, 110624, 114720, 118816,  
122912, 127008, 131104, 135200, 139296, 143392, 147488, 151584, 155680, 159776,  
163872, 167968, 172064, 176160, 180256, 184352, 188448, 192544, 196640, 200736,  
204832, 208928, 213024, 217120, 221216, 225312, 229408, 233504, 237600, 241696,  
245792, 249888, 253984, 258080, 262176, 266272, 270368, 274464, 278560, 282656,  
286752, 290848, 294944, 299040, 303136, 307232, 311328, 315424, 319520, 323616,  
327712, 331808, 335904, 340000, 344096, 348192, 352288, 356384, 360480, 364576,  
368672, 372768, 376864, 380960, 385056, 389152, 393248, 397344, 401440, 405536,  
409632, 413728, 417824, 421920, 426016, 430112, 434208, 438304, 442400, 446496,

System Disk Upgrade

```

450592, 454688, 458784, 462880, 466976, 471072, 475168, 479264, 483360, 487456,
491552, 495648, 499744, 503840, 507936, 512032, 516128, 520224, 524320, 528416,
532512, 536608, 540704, 544800, 548896, 552992, 557088, 561184, 565280, 569376,
573472, 577568, 581664, 585760, 589856, 593952, 598048, 602144, 606240, 610336,
614432, 618528, 622624, 626720, 630816, 634912, 639008, 643104, 647200, 651296,
655392, 659488, 663584, 667680, 671776, 675872, 679968, 684064, 688160, 692256,
696352, 700448, 704544, 708640, 712736, 716832, 720928, 725024, 729120, 733216,
737312, 741408, 745504, 749600, 753696, 757792, 761888, 765984, 770080, 774176,
778272, 782368, 786464, 790560, 794656, 798752, 802848, 806944, 811040, 815136,
819232, 823328, 827424, 831520, 835616, 839712, 843808, 847904, 852000, 856096,
860192, 864288, 868384, 872480, 876576, 880672, 884768, 888864, 892960, 897056,
901152, 905248, 909344, 913440, 917536, 921632, 925728, 929824, 933920, 938016,
942112, 946208, 950304, 954400, 958496, 962592, 966688, 970784, 974880, 978976,
983072, 987168, 991264, 995360, 999456, 1003552, 1007648, 1011744, 1015840, 1019936,
1024032, 1028128, 1032224, 1036320, 1040416, 1044512, 1048608, 1052704, 1056800, 1060896,
1064992, 1069088, 1073184, 1077280, 1081376, 1085472, 1089568, 1093664, 1097760, 1101856,
1105952, 1110048, 1114144, 1118240, 1122336, 1126432, 1130528, 1134624, 1138720, 1142816,
1146912, 1151008,

```

*{about 2 min. between this line of output and the next}*

rotational delay between contiguous blocks changes from 7ms to 0ms

*{Do a file system check on the 2 filesystems you just created}*

```

fsck.ffs /dev/dsk/isc0d1s0
** /dev/dsk/isc0d1s0
** Last mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames

```

```

** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2 files, 9 used, 21574 free (14 frags, 2695 blocks, 0.1% fragmentation)

**** FILE SYSTEM WAS MODIFIED ****

 # fsck.ffs /dev/dsk/isc0d1s6
** /dev/dsk/isc0d1s6
** Last Mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2 files, 9 used, 541026 free (10 frags, 67627 blocks, 0.0% fragmentation)

**** FILE SYSTEM WAS MODIFIED ****

```

*{Make a mount directory for the new root filesystem to mount to}*

```
mkdir /newroot
```

*{Mount the new root filesystem}*

```
mount /dev/dsk/isc0d1s0 /newroot
cd /newroot
```

*{Make a mount directory for the new usr filesystem to mount to}*

```
mkdir usr
{Mount the new usr filesystem}
```

```
mount /dev/dsk/isc0d1s6 /newroot/usr
```

*{Check to make sure your new filesystems are there}*

```
df
Filesystem kbytes used avail capacity Mounted on
/dev/root 18432 10736 7696 58% /
/dev/usr 243280 205896 37384 85% /usr
/dev/dsk/isc0d1s0 21576 2168 19408 10% /newroot
/dev/dsk/isc0d1s6 541032 54112 486920 10% /newroot/usr
```

```
cd /newroot
```

*{Make sure your current directory is /newroot}*

```
pwd
/newroot
```

*{This copies from the existing root to the new disks' root...takes 3 min.}*

```
dump.ffs 0f - / | restore.ffs rf -
DUMP: Date of this level 0 dump: Tue Nov 7 15:44:30 1990
DUMP: Date of last level 0 dump: Tue Nov 7 09:36:21 1990
DUMP: Dumping /dev/root (/) to standard output
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 9969 tape blocks on 0.26 tape(s).
```



```
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
Warning: ./lost+found: File exists
Warning: ./usr: File exists
./tmp/rstmdir626481870: not found on tape
./tmp/rstmode626481870: not found on tape
./dev/log: unknown file mode 010666
DUMP: DUMP: 9971 tape blocks on 1 tape(s)
DUMP: DUMP IS DONE
```

*{Verify the /newroot filesystem has more kbytes in the use column than / does}*

```
df
Filesystem Type kbytes use avail %use Mounted on
/dev/root ffs 18439 10748 7691 58% /
/dev/usr ffs 243286 205902 37384 85% /usr
/dev/dsk/isc0d1s0 ffs 21583 11130 10453 52% /newroot
/dev/dsk/isc0d1s6 ffs 541035 54105 486930 10% /newroot/usr
```

**# cd usr**

*{Make sure your current directory is /newroot/usr}*

```
pwd
/newroot/usr
```

*{This copies the existing /usr to the new disks' /usr...takes about 25 min.}*

```
dump.ffs 0f - /usr | restore.ffs rf -
DUMP: Date of this level 0 dump: Tue Nov 7 15:59:10 1989
DUMP: Date of last level 0 dump: Tue Nov 7 09:55:46 1989
```

```
DUMP: Dumping /dev/usr (/usr) to standard output
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 187205 tape blocks on 4.81 tape(s).
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: 18.10% done, finished in 0:22
./lib/cron/FIFO: unknown file mode 010600
DUMP: 43.48% done, finished in 0:13
DUMP: 64.69% done, finished in 0:08
DUMP: 85.64% done, finished in 0:03
DUMP: DUMP: 187144 tape blocks on 1 tape(s)
DUMP: DUMP IS DONE
```

*{Verify the /newroot/usr filesystem has more kbytes in the use column than /usr does}*

```
df
Filesystem Type kbytes use avail %use Mounted on
/dev/root ffs 18439 10749 7690 58% /
/dev/usr ffs 243286 205904 37382 85% /usr
/dev/dsk/isc0d1s0 ffs 21583 11130 10453 52% /newroot
/dev/dsk/isc0d1s6 ffs 541035 236160 304875 44% /newroot/usr
```

*{Now use dvhtool to add sash to the new disks volume header}*

```
dvhtool
```

```
Command? (read, vd, pt, dp, write, bootfile, or quit): read
Volume? /dev/dsk/isc0d1vh
```

```
Command? (read, vd, pt, dp, write, bootfile, or quit): vd
```

```
current contents:
file len lbn

Command? (d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, or l)a /stand/sash sash
Command? (d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, or l)l

current contents:
file len lbn
sash: 223480 -1

Command? (d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, or l)<CR>

Command? (read, vd, pt, dp, write, bootfile, or quit): write
Volume? (/dev/dsk/isc0d1vh)<CR>

Command? (read, vd, pt, dp, write, bootfile, or quit): quit

cd /
shutdown -y -i0 -g0
```

Wait for shutdown to complete and remove power from the ESV.

**At this point what you do depends on whether the customer is keeping his original drive or not.**

**If the customer is keeping their original drive:**

Remove jumper from the model 94191's drive select field, making it drive 0, and install a jumper on the original drives' select field, making it drive 1. (see attached specification. sheets)

Install the drive tray back into the ESV.

Restore power to the ESV and boot normally.

The original drive will not be mounted. But it is available to the customer to use as they see fit. The old root and usr filesystems could be mounted or new filesystems could be created using any of the usable partitions. If new filesystems are created the old data will be gone. If there are any questions with regard to this call dispatch for assistance.

**If the customer is not keeping their original drive:**

Remove both disk drives from the tray.

Remove the jumper from the model 94191's drive select field, making it drive 0, reinstall the sip terminators (see attached specification sheet), and reinstall the 94191 where the original drive was.

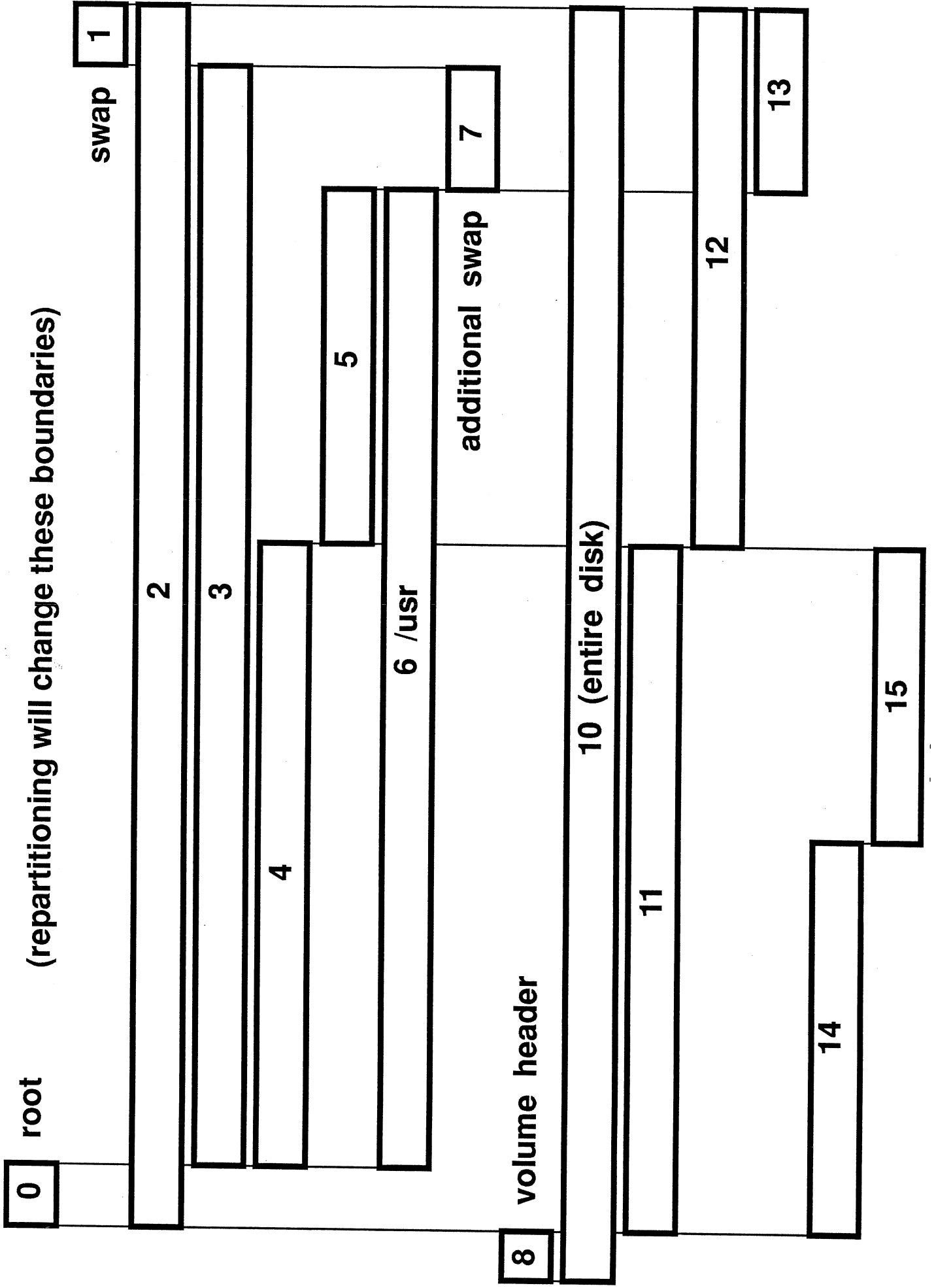
Install the drive tray back into the ESV.

Restore power to the ESV and boot normally.

**You may now proceed with a normal update install of the 1.3 OS and its associated software.**

# Appendix 5. ESV \_efault Disk Partitioning

(repartitioning will change these boundaries)



## Fstest? What's a Fstest?

This may be the response you get someday from a user when you inquire as to the whereabouts of the fstest account. The fstest account is a luxury on the ESV that some users, especially those with small disk drives, won't feel they can afford.

With that in mind, here is a procedure that makes use of a system disk swap partition that will allow you to do basic things like run diagnostics. Any swap partition can be used as long as it contains at least 25000 blocks and the swap -l command shows all blocks free.

Please keep in mind this procedure was written using partition 7 of a 380 mb drive on a system running ES/os 1.2. But you should be able to adapt it to any model of disk drive or swap partition by just changing a few of the variables. Also:

All lines beginning with "#" are command lines.

Words inclosed with { } are notes I have entered.

The tape we are installing fstest from is the ES/PEX Server Tape 1.2.

The well prepared Field Engineer should have a copy of the ES/os and the ES/PEX tapes in their possession. The following should serve as an outline for getting your fstest directory back, at least temporarily.

**{Check for available swap.}**

```
swap -l
path dev swaplo blocks free
/dev/dsk/isc0d0s1 16,1 0 39480 39480
/dev/dsk/isc0d0s7 16,7 0 44520 44520
```

**{Delete partition 7 from available swap.}**

```
swap -d /dev/dsk/isc0d0s7 0
```

**{Verify partition 7 has been removed from swap.}**

```
swap -l
path dev swaplo blocks free
/dev/dsk/isc0d0s1 16,1 0 39480 39480
```

**{Double check size of partition 7 with prtvtoc.}**

```
prtvtoc /dev/rdisk/isc0d0vh
* /dev/rdisk/isc0d0vh (bootfile "/unix") partition map
*
* Dimensions:
* 512 bytes/sector
* 840 sectors/track
* 1 tracks/cylinder
* 768 cylinders
* 765 accessible cylinders
* Unallocated space:
```

|   | Start  | Size    | Tag | Flags | First Sector | Sector Count                   | Mount Directory |
|---|--------|---------|-----|-------|--------------|--------------------------------|-----------------|
| * | 2520   | -472    | 4   | 0     | 2520         | 39480                          | /               |
| * | 384000 | -381952 | 4   | 0     | 605640       | 39480                          |                 |
| * | 645120 | -643072 | 4   | 0     | 2048         | 643072                         |                 |
| * | 193024 | -190504 | 4   | 0     | 48128        | 531456                         |                 |
| * | 561120 | -512992 | 4   | 0     | 48128        | 335872                         |                 |
| * | 579584 | -531456 | 4   | 0     | 384000       | 167936                         |                 |
| * | 384000 | -190976 | 4   | 0     | 42000        | 519120                         |                 |
| * | 551936 | -167936 | 4   | 0     | 561120       | 44520<----{size of prtition 7} |                 |
| * | 645120 | -93184  | 0   | 0     | 0            | 2520                           |                 |
| * | 645120 | -84000  | 6   | 0     | 0            | 645120                         |                 |
| * |        |         | 4   | 0     | 2048         | 381952                         |                 |
| * |        |         | 4   | 0     | 384000       | 261120                         |                 |
| * |        |         | 4   | 0     | 551936       | 93184                          |                 |
| * |        |         | 4   | 0     | 2048         | 190976                         |                 |
| * |        |         | 4   | 0     | 193024       | 190976                         |                 |



**{Creat a new filesystem on partition 7. Note: 44520 is size of the partition from prtvtoc and 94181 is the model of the drive. Use the correct variables for your system!}**

```
#newfs.ffs -s 44520 /dev/dsk/isc0d0s7 94181
Warning: 32 sector(s) in last cylinder unallocated
/dev/dsk/isc0d0s7: 44512 sectors in 174 cylinders of 1 tracks, 256
sectors
 22.8Mb in 11 cyl groups (16 c/g, 2.10Mb/g, 896 i/g)
super-block backups (for fsck -b#) at:
 32, 4128, 8224, 12320, 16416, 20512, 24608, 28704, 32800, 36896,
40992,
rotational delay between contiguous blocks changes from 7ms to 0ms
```

**{Creat a mount point for the new filesystem.}**

```
#mkdir /new_fstest
```

**{Mount the new filesystem.}**

```
#mount /dev/dsk/isc0d0s7 /new_fstest
```

**{Check to see that the new filesystem is mounted.}**

```
#df
Filesystem Type kbytes use avail %use Mounted on
/dev/root ffs 18439 10744 7695 58% /
/dev/usr ffs 243286 205461 37825 84% /usr
/dev/dsk/isc0d0s7 ffs 20831 2093 18738 10% /new_fstest
```

**{Begin the install.}**

# /usr/pkg/bin/inst

MIPS software package installation

Install package relative to where [/]? **/new\_fstest**

**{Insert ES/PEX Server Tape into the tape drive}**

Please mount the (first, if multiple tapes) distribution tape, then  
press return... **<CR>**  
Rewinding the tape...  
Verifying the tape id... ok

Extracting packaging information tree... tar: blocksize = 8

=====  
selecting subpackages =====

Install subpackage executables\_s (y n) [y]? **<CR>**

Install subpackage library\_s (y n) [y]? **n**

Install subpackage pexs-man (y n) [y]? **n**

Install subpackage fstest (y n) [y]? **<CR>**

Selected subpackages:

fstest

Is this what you want (y n) [y]? **<CR>**

=====  
setting system clock/calendar  
=====

The current value of the clock is: Thu Nov 2 10:27:21 MST 1990  
Is the clock correct (y n) [y]? <CR>

=====  
verifying single-usr mode  
=====

This system is not presently in a single-user run level. Installation of a package can fail if performed at this level. We recommend that the system be brought to a single user run level (using "init S") prior to performing the installation.

Are you absolutely sure you wish to continue (y n) [n]? y

=====  
perserving local files  
=====

No present list or findmods list for fstest- perserve not executed.

=====  
verifying disk space  
=====

The system will now be checked to verify that there is enough disk space with the current configuration to successfully install the package (and any selected optional subpackages). For large packages (especially operating system packages), this can be time consuming...

There is enough space.

=====  
stripping old links  
=====

Stripping old links for fstest....

===== extracting files from subpackage archives =====

Rewinding the tape...

Verifying the tape id... ok

Forward spacing the tape...

Forward spacing the tape...

Loading subpackage: fstest... tar: blocksize = 8

Forward spacing the tape...

Rewinding the tape...

===== running comply =====

running first comply pass...

running second comply pass...

There were no comply messages from the second pass.

===== cleaning up old versions =====

An attempt will now be made to clean up any files left over from previous versions of the software which has just been installed.

Searching for old versions to remove...

===== restoring perserved user files =====

No perserve list or findmods list for fstest- no files restored.

===== cleaning up =====

Remove install tools (y n) [y]? <CR>

===== installation complete =====

```
#cd /new_fstest/usr/people/people/fstest/diag
#ls
bogus.c.dump
dma_from_gif_test.dump
dsp_debug.dump
dsp_memory_test.dump
dsp_registers_test.dump
dsp_test.dump
error.dump
esv_sysdiag
gbus_arbiter_order_test.dump
gbus_arbiter_test.dump
gif_arbiter_order_test.dump
gif_arbiter_test.dump
lch_data_lines_test.dump
lch_empty_test.dump
lch_fifo_patterns_test.dump
vrint_controller_and_counter_test.dump
lch_parallel_lod_test.dump
lch_serial_lod_test.dump
local_memory_test.dump
mandril.i
memory_block_transfer_test.dump
ofifo_arbiter_test.dump
ofifo_hf_test.dump
pio_comm_test.dump
pp_vecs
ppa_fifo_arbiter_test.dump
ppa_fifo_empty_test.dump
sio_comm_test.dump
sync_block_transfer_test.dump
sync_memory_test.dump
```

**{You can now run ./esvsysdiag as you normally would  
from the fstest home directory.}**

**{After you are through with the diagnostics or anything else you may have needed from the fstest directory, the following will restore the system back to the configuration you found it.}**

```
cd /
```

```
umount /new_fstest
{Unmount the temporary filesystem you created.}
```

```
swap -a /dev/dsk/isc0d0s7 0 44520
{Add the partition you took from swap back to swap.}
```

**{Verify the swap partitions are restored to what they were originally.}**

```
swap -l
path dev swaplo blocks free
/dev/dsk/isc0d0s1 16,1 0 39480 39480
/dev/dsk/isc0d0s7 16,7 0 44520 44520
```

**{Verify the mounted filesystems are back to what they were.}**

```
df
Filesystem Type kbytes use avail %use Mounted on
/dev/root ffs 18439 10745 7694 58% /
/dev/usr ffs 243286 205469 37817 84% /usr
```

## ESV Maintenance Quiz

---

- 1. How many DSP boards must be installed for the ESV to work?**
- 2. How can you find the ethernet address for the ESV?**
- 3. What is the minimum number of memory modules required for the 25Mhz CPU? for the 33/40Mhz CPU?**
- 4. Which memory module connectors must be equipped with memory modules on the 25Mhz CPU? on the 33/40Mhz?**
- 5. Can 16 Mbyte memory modules be used with 4 Mbyte modules in the ESV?**
- 6. On the 25Mhz CPU, which of the four sets of jumper pins at U900 need jumpers for 4Mb modules? for 16Mb modules?**
- 7. How can you be sure that machine dependent variables, such as ethernet address, stay with the machine when replacing the CPU board?**
- 8. Can memory modules from a 33/40Mhz CPU be used on a 25Mhz CPU?**
- 9. What card slot would DSP2 be found in, assuming the Video card to be in slot 1 and the CPU card in slot 10?**
- 10. Can Frame Buffer 0 be exchanged with Frame Buffer 1?**
- 11. Which board would be most suspect if there was no video on**

## ESV Maintenance Quiz

---

the display and the LEDs on the CPU flashed a value of 0x41 during the non-visible power-on confidence tests?

12. Which board would be most suspect if one of the visible power-on confidence tests failed?

13. Which of the ports on the rear of the cabinet would you connect a de-bug terminal?

14. What is the directory path to the system diagnostics?

15. Within which software package could the system diagnostics be found on tape?

16. What command would you use for an immediate shutdown of the ESV?

17. What command would you use to boot the ESV to the single-user run level?

18. What is the difference between a Scratch Install and an Update Install?

19. What command could be used to determine the version of the Operating System installed on the ESV?

20. What file contains a record of E&S software packages installed on the ESV?

21. Where can you find the instructions for loading E&S software



the /usr filesystem?

**32. What partitions of the system disk are normally used for swap space?**

**33. What command can be used to print the volume headers table of contents for disk drive 1?**

**34. What utility can be used to modify the disk's volume header?**

**35. What command can be used to display the status of all currently mounted filesystems?**

**36. What command can be used to list available swap space?**

**37. What command can be used to display all of the processes currently running?**

**38. What command can be used to display the status of the ethernet interface (la0)?**

**39. What command can be used to quickly check the ability to communicate between two ESVs over ethernet?**

**40. What file contains the ESV's hostname, broadcast address, and netmask?**

**41. What file determines the ESV's internet address?**

**42. What command could be used to do a full dump of the /usr**

## ESV Maintenance Quiz

---

**filesystem to cartridge tape?**

**43. Where could you find a record of system messages sent to the console?**

**44. Where would you look for core dumps saved by the system?**

**45. What utility could be used to analyze a core dump?**