COURSE INTRODUCTION

OVERALL COURSE OBJECTIVE

The IRIS-Series 3000 Workstation Differences course is intended to provide the IRIS-Series 4D experienced field maintenance personnel with the skills and knowledge needed to perform remedial maintenance procedures.

TARGET AUDIENCE

The target audience for this course are SGI Field Engineers, Technical Field Specialists, and Product Support Engineers who have successfully completed the core Hardware Maintenance program. Ideally, the participant will have remedial maintenance experience with the core product prior to commencing this course.

PREREQUISITES

To meet the entry-level criteria for participation in the course, prospective students *must* meet the following requirements:

- have successfully completed the IRIS System Accelerator Course.
- have successfully completed the IRIS 4D-Series Hardware Maintenance Course.

COURSE ADMINISTRATION

This course is administered in a self-paced format and utilizes competency-based methodology. In a competency-based program, the participant proceeds through each instructional unit only after meeting the objectives described for the previous one. This implies that, at each instructional unit, the participant is tested, based on the stated objectives *before* proceeding to the next level. This ensures success by providing the participant with the time required to achieve the objective(s). If the participant does not perform to the standard defined for the objective(s), additional study is usually required before he or she will be allowed to progress to the next unit.

The methodology permits the participant to study at his or her own comfortable pace and to make, in some cases, his or her own decisions about the sequencing of instructional units and selection of learning resources.

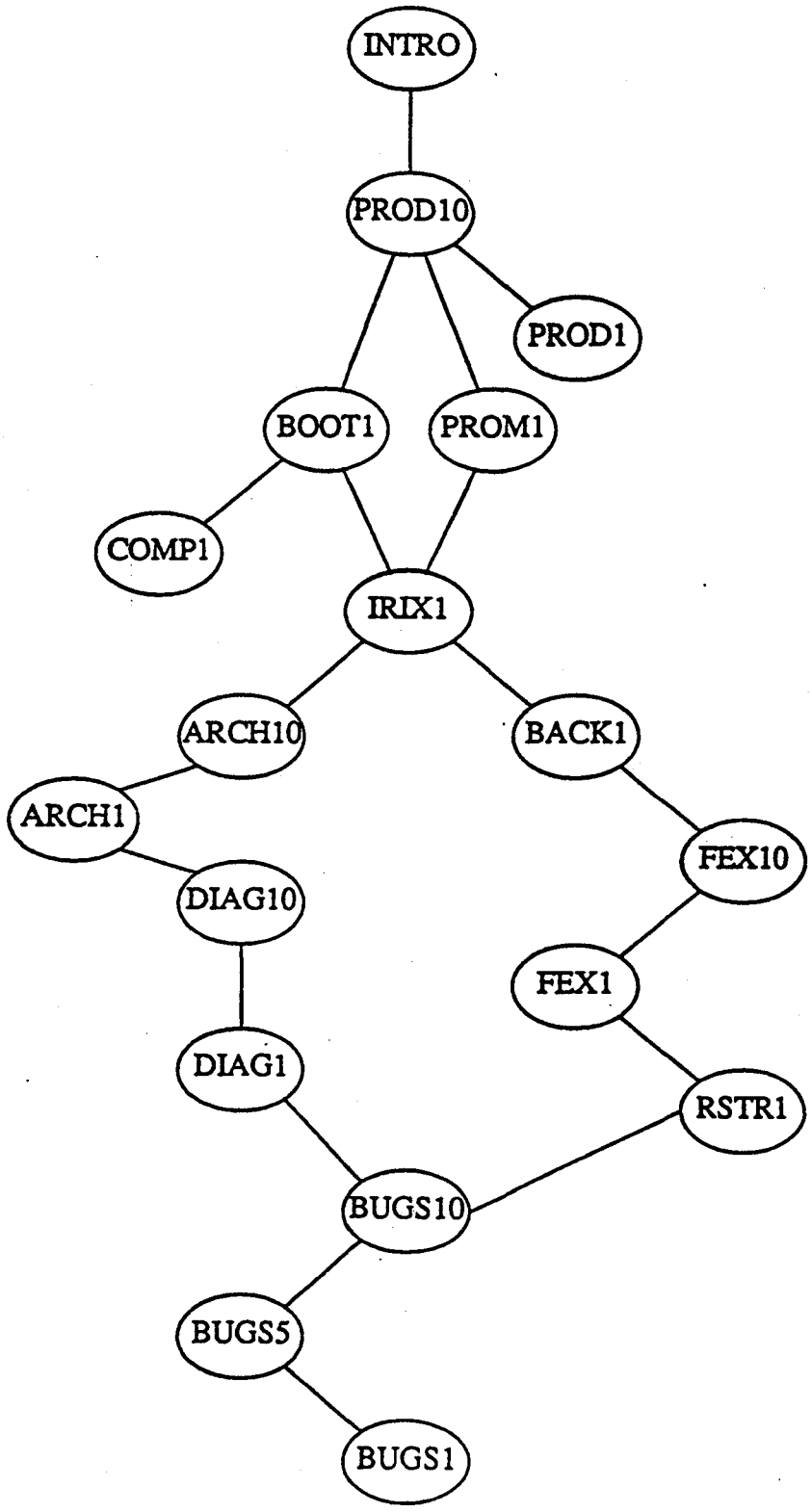
PERFORMANCE MEASUREMENT

Participants studying this course are, by prior core course evaluation, already well qualified to proceed through this course unattended. This course is designed to be successfully completed by *all* participants *as long as they faithfully follow the course procedures.*

REFERENCES

- This Guide
- IRIS-3000 Series Programmer's Manual, Volume 1A and 1B
- IRIS-3000 Series Owner's Guide
- IRIS-3000 Series Diagnostic Tape

COURSE MAP


INTRO
PROD10
PROD1
BOOT1
PROM1
COMP1
IRIX1
ARCH10
BACK1
ARCH1
FEX10
DIAG10
FEX1
DIAG1
RSTR1
BUGS10
BUGS5
BUGS1

MODULE OBJECTIVES

Intro

At the completion of this lesson, the participant will be able to:

- State the overall objectives of the course.
- State how the course will be administered.
- Identify the prerequisites of the course.
- State the performance requirements and how they will be measured.
- Describe the course procedures.
- Describe the topics that will be presented during this course.

Prod10

At the completion of this lesson, the participant will be able to:

- Recognize the major software components.
- Given a blank block diagram, correctly label the major circuit board assemblies with their correct designation.
- Identify each IRIS 3000 subsystem and configuration.
- Differentiate the features of the various models in the IRIS 3000-Series product line.

Prod1

At the completion of this lesson, the participant will be able to:

- Given a list of document titles, match the correct one with its appropriate content.

Prom1

At the completion of this lesson, the participant will be able to:

- Describe the five general functions of the PROM Monitor firmware.
- recognize the correct syntax of the PROM Monitor commands.
- Match the PROM Monitor device acronyms with their correct device.
- Describe the function and usage of the PROM Monitor commands.

Boot1

At the completion of this lesson, the participant will be able to:

- Locate configuration switches 1 through 9.
- Match configuration switches 1 through 9 with their correct functions.
- Boot the system into IRIX single-user mode.
- Describe the system configuration from the information displayed at the console monitor during system "boot".
- Perform "graceful" system shutdown procedures.

Comp1

At the completion of this lesson, the participant will be able to:

- Locate, remove and reinstall all field replaceable units.
- Locate hardware configuration switches.
- Correctly configure main memory PCBs.
- Correctly configure bitplane PCBs.
- Correctly configure the IP2 PCB.

- Recognize the correct slot location for PCBs given various system configurations.
- Correctly connect outboard connectors.

Irix1

At the completion of this lesson, the participant will be able to:

- Recognize the shell command differences differences between IRIX version 3.6 and IRIX System 5.3 version 3.1.
- Recognize system configuration file differences.
- Given a system environmental requirement, correctly configure it to conform to that configuration.

Arch10

At the completion of this lesson, the participant will be able to:

- Describe the physical organization of the CPU and I/O subsystems.
- Identify the major functional blocks of the CPU and I/O subsystems.
- Match the major functional blocks of the CPU and I/O subsystems with their correct descriptions.
- Locate and verify the correct setting of each configuration switch/jumper on the IP2, FP1, and IM1 boards.

Arch1

At the completion of this lesson, the participant will be able to:

- Describe the physical organization of BP3 Bit Plane boards.
- Match the color programming mode to its correct description.
- Map the system bitplane identifiers to the correct BP3 board.

- Identify the logical functions of the GF3 board.
- Identify the logical functions of the UC4 board.
- Identify the logical functions of the DC3 board.

Back1

At the completion of this lesson, the participant will be able to:

- Recognize the function and organization of the "mkboot" tape.
- Create a "mkboot" backup of the "root" filesystem.
- Create "root" and "usr" filesystem backups using "cpio".

Fex10

At the completion of this lesson, the participant will be able to:

- Correctly load the appropriate "fex" program from a mkboot tape.
- Enter the "FE" portion of "fex".

Fex1

At the completion of this lesson, the participant will be able to:

- Use "bad block" commands to list, modify, and add to the disk's bad block table.
- Format the disk drive.
- Exercise the disk drive.

Rstr1

At the completion of this lesson, the participant will be able to:

- Restore the "root" filesystem using "fex".

- Restore the "usr" filesystem using "fex".
- Label the disk device using the "sgilabel" command.
- Label each filesystem using the "labelit" command.

Diag10

At the completion of this lesson, the participant will be able to:

- Match the diagnostic test program with its' functional description.
- Match diagnostic programs with their associated workstation models.
- Load and execute diagnostic test programs.
- Execute CPU, FPU, and memory diagnostic tests.
- Interpret CPU, FPU, and memory diagnostic test messages.

Diag1

At the completion of this lesson, the participant will be able to:

- Execute graphics subsystem diagnostic tests.
- Interpret graphics subsystem diagnostic test messages.

Bugs10

At the completion of this lesson, the participant will be able to:

- Recognize PROM monitor program configuration messages.
- Recognize IRIX boot process configuration messages.
- Match IRIX run-levels with their correct function.
- Recognize functions of IRIX configuration shell scripts and files.
- List the minimum board configuration required to:
  - boot IRIX

- initialize the graphics subsystem

Bugs5

At the completion of this lesson, the participant will be able to:

- Isolate general system failures to the Field Replaceable Unit.

Bugs1

At the completion of this lesson, the participant will be able to:

- Isolate graphics subsystem failures to the Field Replaceable Unit.

Last1

At the completion of this lesson, the participant will be able to:

- Summarize the overall differences between the various workstations described during this course.
- Summarize the differences between the IRIS 3000-Series and IRIS 4D-Series workstations.

Contents

COMP1 1-1
- Introduction 1
- Objectives 1
- Criterion Test 1
- Resources 1
  - Equipment You'll Need 2
- SECTION 1: Instructional Procedure 4
- SECTION 2: Diagrams 8
  - **DIAGRAM 1: 3000-Series I/O Panel Cable Connections** 9
  - **DIAGRAM 2: Basic 68020 PCB assignments - page 1** 10
  - **DIAGRAM 3: Basic 68020 PCB assignments - page 2** 11
  - **DIAGRAM 4: IP2 interconnect - page 1** 12
  - **DIAGRAM 5: IP2 interconnect - page 2** 13
  - **DIAGRAM 6: Ethernet interconnect** 14
  - **DIAGRAM 7: DSD interconnect** 15
  - **DIAGRAM 8: Graphics sub-system interconnect** 16
  - **DIAGRAM 9: PCB switch settings - page 1** 17
  - **DIAGRAM 10: PCB switch settings - page 2** 18
  - **DIAGRAM 11: PCB switch settings - page 3** 19
  - **DIAGRAM 12: PCB switch settings - page 4** 20
  - **DIAGRAM 13: I/O compartment components location** 21
  - **DIAGRAM 14: Power compartment components location** 22
- SECTION 3: Test Items 23

COMP1

Introduction

In this module, you will get your first chance to actually get your hands on the inside of a workstation. This lab procedure will guide through the process of removing and replacing each FRU.

Objectives

At the completion of this lesson, you will be able to:

- Locate, remove and reinstall all field replaceable units.
- Locate hardware configuration switches.
- Correctly configure main memory PCBs.
- Correctly configure bitplane PCBs.
- Correctly configure the IP2 PCB.
- Recognize the correct slot location for PCBs given various system configurations.
- Correctly connect outboard connectors.

Criterion Test

Correctly remove and replace all designated FRUs and perform configuration procedures on IP2, IM1, and BP3 boards.

Resources

This guide

3000-Series Owner's Manual

Equipment You'll Need

To complete the component replacement lab session you will need the following tools and equipment:

- Anti-static wrist strap
- No. 2 Phillips Screwdriver
- No. 1 Slotted Screwdriver (long shank)
- 5/32-inch hex key wrench

This module contains three sections:

- **SECTION 1:**

  This section is contains the instructional procedure and explanations for removing and replacing field replaceable components.

- **SECTION 2:**

  This section contains diagrams for reference while performing the procedure outlined in Section 1. Section 1.

- **SECTION 3:**

  This section contains review questions that you should complete after studying the material in this module.

SECTION 1: Instructional Procedure

Follow the steps of the lab project carefully. When you are through, power-on your workstation and insure that it functions correctly.

STEP 1: Perform a reboot command.

STEP 2: Power off the workstation.

Step 3: Remove the power cord from the back of the workstation.

Step 4: Remove the following panels:

- **Front panel:**
  Removed by inserting a screw driver or other thin object into the seam that separates the front panel and side panels, and twisting screw driver slightly. The front panel should pop off.
- **Top panel:**
  Removed by firmly grasping the front edge of the panel and lifting up. Like the front panel, this panel is held in place by "snap in" type connectors.
- **Card cage panel:**
  This panel is on the front of the workstation and held in place by phillip head screws. You do not have to completely remove the screws.
- **I/O compartment panel:**
  This panel is located on top of the workstation, it is the larger of two panels on top and held in place by phillip head screws. Like the card cage panel, its screws do not have to be completely removed to remove the panel.
- **Power compartment panel:**
  This is the smaller panel on top of the workstation. It too is held in place by phillip head screws.

STEP 5: Rear panel connections.

Using diagram #1, verify the connections at the rear of the workstation.

Two notes of interest:

- The monitor used on the 2400 Turbos is different from the monitor used on the 30X0 workstations. The RGB cable connections are clearly called out on the 30X0 monitors, but the way each line is terminated is different.

  On the old style monitors (2400T) the line termination is controlled by switches on the back panel of the monitor, whereas, terminators on the new style monitors (30X0) are attached to the monitor using a beaded chain.

  The principle of termination remains the same for both type of monitors; only the last monitor in a string of monitors will be terminated.

  The line termination must be 75 ohms for normal sync and color drive.

- The keyboard used with the old style monitors needs the *Junction Box*. The keyboard and mouse plug into the "J-box" and the "J-box" is connected to port 1 of the workstation.

  The new style keyboards contain the logic that is contained in the "J-box", therefore, the mouse plugs into the keyboard and the keyboard is connected to port 1. (The new style keyboards allow the mouse to plug into the *right* or *left* end of the keyboard)

Now, check the cable connections and monitor RGB terminators. When you are satisfied continue with the project.

STEP 6: Card cage configuration.

Using diagrams 2-12, remove all the cards from your workstation.

NOTE:
DO NOT handle the cards in a manner that brings your body in contact with the card edge connectors. DAMAGE may occur caused by *STATIC DISCHARGE* into the card. When you remove or move cards about, use the ESD wrist straps provided at each workstation and use the special bags that cards are shipped in.

- Verify switch settings on cards with switch packs.
- Look at other function jumpers on the cards, but DO NOT move any of the jumpers. Remember, if you replace a card in the field, insure that all the jumpers on the replacement card match the card removed.
- Look for part numbers, serial numbers, and revision levels on the cards.
  - SGI cards will have their numbers along the edge somewhere.
  - Cards purchased from vendors will not have an SGI part number on them, i.e. DSD, Ethernet, Storager II.
- Re-install the cards and cable together per diagrams 2-12.

STEP 7: Removing disk drives.

Using diagrams 7 and 13 and a *long neck* screw driver, remove the disk and tape drives.

- Using diagram 12, verify the address jumper settings on the disk PCB's.
- Re-install the drives and cables per diagrams 7 and 17.

STEP 8: Power compartment components.

Using diagrams 14-17, verify the location and cabling of the various components in the power compartment.

DO NOT remove the power supply but locate the screws that fasten them. You may loosen the screws just to verify that you have located the correct ones.

Also, DO NOT remove any of the power supply harness wires, you will get to do this some day in the field, just verify their routes.

STEP 9: Clean up and test.

Re-install all parts and panels and power on your workstation and verify that it functions O.K.

SECTION 2: Diagrams

DIAGRAM 1: 3000-Series I/O Panel Cable Connections

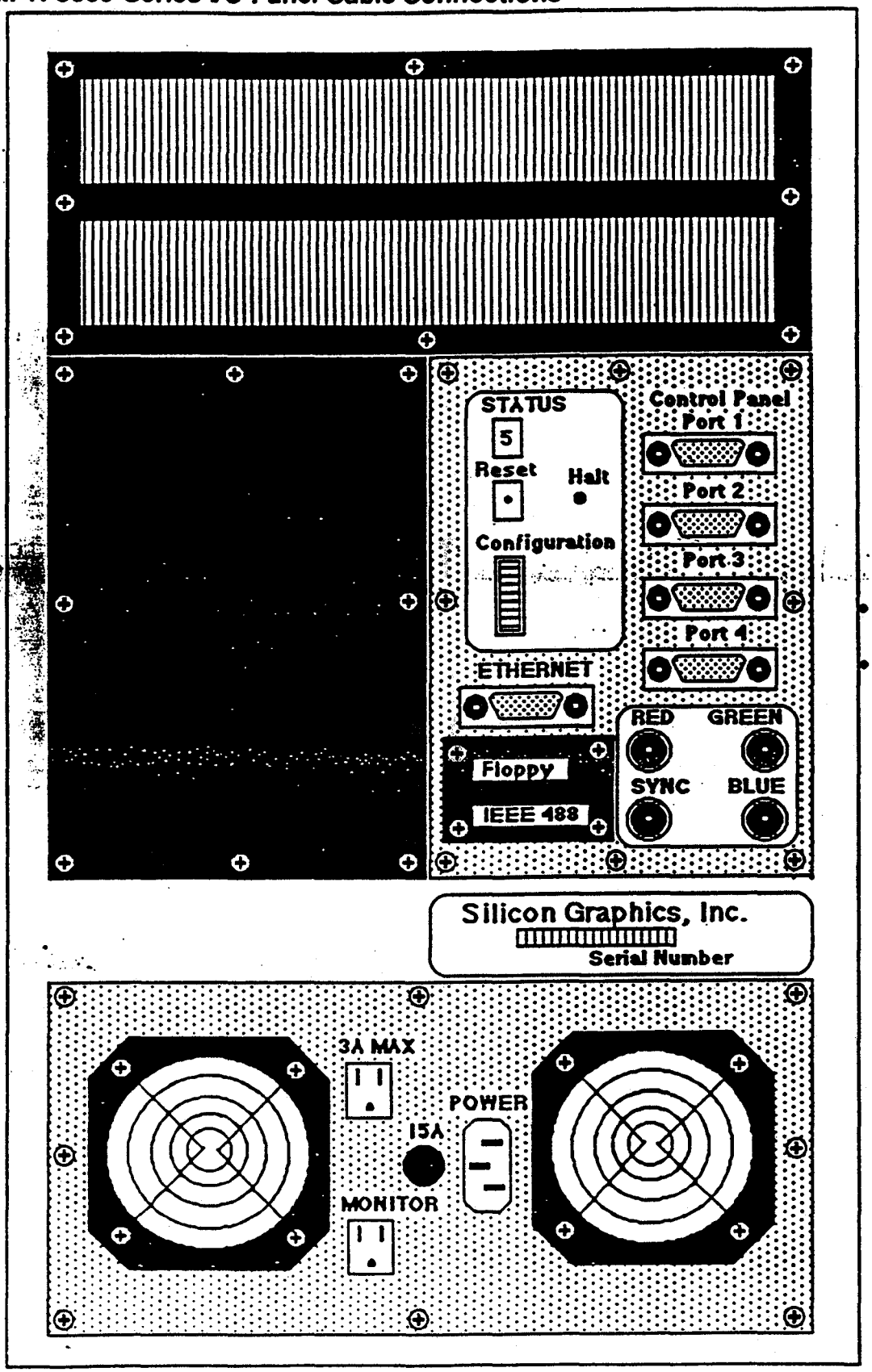
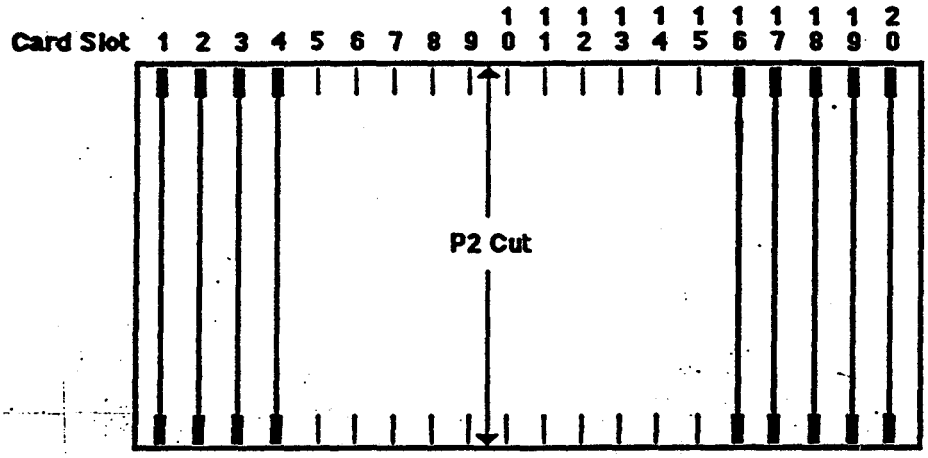


DIAGRAM 2: Basic 68020 PCB assignments - page 1

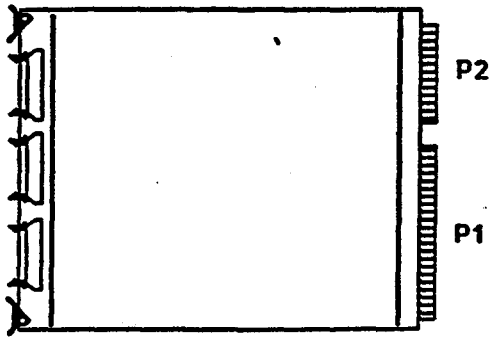
4 Meg, No FPA



Front View of Card Cage

SLOT	PCB	DESCRIPTION
1	IM1	Memory Board
2	IP2	Processor
3	Enet	Ethernet
4	DSD	Disk / Tape Controller
5	Option	For Location of Options, See Next Page.
6	Option	
7	Option	
8	Option	
9	Option	
10	Option	
11	Option	
12	Option	
13	Option	
14	Option	
15	Option	
16	BP3	BR Plane
17	BP3	BR Plane
18	DC4	Display Controller
19	UC4	Update Controller
20	GF2	Frame Buffer

Basic PCB Slot Assignments



PCB

Contents

BOOT1 1-1

Introduction 1

Objectives 1

Criterion Test 1

Resources 2

SECTION 1:Starting Up the System 4

The Boot Process 4

Auto-configuration 5

System Device Names 7

User Operation Modes 10

System Shutdown 11

SECTION 2:Boot Exercise 12

Test Items 16

BOOT1

Introduction

In this module, you will learn about the PROM monitor firmware and the procedures for "booting" the workstation. You will also learn how to determine the workstation hardware configuration from the information displayed at the console during startup. You will note, too, the differences between the 3000-Series and 4D-Series workstation--most notably the absence of the *sash* environment.

Objectives

At the completion of this lesson, you will be able to:

- Locate configuration switches 1 through 9.
- Match configuration switches 1 through 9 with their correct functions.
- Boot the system into IRIX single-user mode.
- Describe the system configuration from the information displayed at the console monitor during system "boot".
- Perform "graceful" system shutdown procedures.

Criterion Test

Perform system bootstrap and shutdown procedures, describing system hardware configuration, including:

- Main memory size
- Configuration switch settings
- Peripheral controller configuration
- Disk drive capacity

Resources

This guide

IRIS Series 3000 Owner's Manual

This module contains three sections:

- **SECTION 1:**

  This section describes the procedure for booting the IRIS workstation.

- **SECTION 2:**

  This section contains a procedure for displaying the PROM Monitor "help" menu, booting the system, and performing a "graceful" shutdown.

- **SECTION 3:**

  This section contains test items that you should complete after studying the material in this module.

SECTION 1:Starting Up the System

The Boot Process

When you enter the boot command from the PROM monitor the IRIX kernel is loaded into memory and IRIX then configures the system and displays a map for you. The map informs the user what equipment is or is not installed on the IRIS, how much memory is available, and what partition the IRIX "root" file system is mounted on.

Simple diagnostics are also executed to test for the presence of Multibus controllers and attached devices.

Below is some of the output displayed when the boot process begins. This display "blinks" on the screen, followed by the display shown on the next page.

```
iris> b
SGI Extent Filesystem
Loading: si:0a:defaultboot
  Text: 038318 bytes
  Data: 0113d8 bytes
  Bss:   024d7c bytes
Jumping to loaded program @ 20000400
```

After this message is displayed, the screen is cleared and the message shown on the next page is output sequentially as the configuration process proceeds.

Auto-configuration

Following is the display that occurs at the IRIS console terminal as the configuration process proceeds. Some of the data changes from software release to release, but for the most part it remains the same.

On the next page descriptions of the equipment mnemonics, shown in this display, are given. (Remember, this output occurs after the output shown on the previous page is displayed and the screen is cleared).

NOTE:
After the line "INIT: SINGLE USER MODE" IRIX will display the contents of file *motd*. This is the *message-of-the-day* file. The user can place any message he/she wants displayed everytime IRIX is booted. Once the message is displayed, the SINGLE USER prompt (#) is displayed.

```
SYSTEM 5 IRIX #0 [Wed May 7 04:49:59 PDT 1986]
(C) Copyright 1986 - Silicon Graphics Inc.
real = 4194304
kmem = 561152
user = 3633152
bufs = 819200 (max=16k)
dsd0 not installed
qic0 not installed
sii0 at mbio 0x07200 ipl 5
si0 (Hitachi 512-17 name: Hitachi 512-17) slave 0
si1 not installed
sf0 floppy (80/2/8) slave 2
siq0 at mbio 0x73fc ipl 5
sq0 (qic02 cartridge tape) slave 0
iph0 not installed
tmt0 not installed
ik0 not installed
ib0 not installed
ib1 not installed
px0 not installed
cd0 not installed
cd1 not installed
cd2 not installed
cd3 not installed
hy0 not installed
ex0 (FW 2.5 HW 4.0) (0800.1400.3948) at mbio 0x7ffc ipl 2
fpa installed
lpen not installed
kernel debugger disabled.
root on si0a
swap on si0b. swplo=0 nswap=64000

INIT: SINGLE USER MODE

#
```

System Device Names

The following text defines each mnemonic assigned to each component of an IRIS workstation. The auto-configuration process will inform the user which controller boards and peripheral devices are present; listing what multibus address has been assigned the board and the number of each device.

MNEMONIC	DESCRIPTION
dsd0	This is the *Disk Controller* for the 72 MB hard disk, the floppy disk, and the 1/4" cartridge tape drive. If this board is installed, a message like the following is displayed next to the mnemonic: **at mbio 0x7f00 ipl 1.** mbio = Multibus. 0x7f00 = Address per board jumpers. ipl 1 = Interrupt priority as determined by position of board on the bus.
qic0	This is the *Quarter inch tape drive* that is attached to the **dsd** controller. Its assigned device address is **0**. Only one tape drive can be attached to the dsd controller.
md*n*	This is the *72 MB disk drive.* Two of these drives can be attached to the **dsd** controller: **md0** and **md1.**
si*n*	This is the *170 MB (or 380 MB) disk drive.* Two of these drives can be attached to the Storager II controller: **si0** and **si1.**
mf0	This mnemonic indicates the *Floppy Disk Drive* that could be attached to the **dsd** controller.
iph0	This is the *SMD Disk Controller.* If installed, the mbio address and interrupt level would be displayed as it was for the dsd board. This controller can have two 474 MB disk drives attached to it.

MNEMONIC	DESCRIPTION
ipn	If the SMD controller was installed, this would indicate the presence of the *474 MB* disk drives: **ip0** and/or **ip1**. This mnemonic is not displayed if the controller is not installed.
sii0	This indicates that the *Storager II Disk/Tape Controller* is installed. This board can handle two **170 MB (or 380 MB)** disk drives and several optional **1/2"** and **1/4"** tape drives. If installed, the mbio address and interrupt level information would be present.
siq*n*	This would indicate the presence of the optional *1/4"* cartridge tape drives that could be attached to the **Storager II** board. Where n = a number.
tmt*n*	This is the mnemonic for the optional *1/2" tape drives* that could be attached to the **Ciprico Tapemaster** board. Where n = a number.
ik0	This indicates the *Color Printer controller.* An IKON board would be installed.
ex0	This mnemonic indicates the presence of the *Excelan Ethernet Controller Board.* Like any board installed, the mbio address is given and the interrupt level is indicated if installed, but some additional information is displayed that a customer may require, that is, the **Ethernet Address**. This address will look like the following: **(0800.1400.3948)**.
ex1	This mnemonic indicates the presence of the *2nd Ethernet Controller Board.*

MNEMONIC	DESCRIPTION
fpa	This indicates that the IRIS has the *Floating Point board* installed.
lpen	This is the *Light pen* option.
ib0	This refers to the IEEE-488 Parallel interface controller, port 1.
ib1	This refers to the IEEE-488 Parallel interface controller, port 2.
px0	This mnemonic is associated with the IBM 3270 Coax Emulator controller.
cd0	This refers to the 8-Port Serial Interface Controller.
cd1	This refers to the second 8-Port Serial Interface Controller.
cd2	This refers to the third 8-Port Serial Interface Controller.
cd3	This refers to the fourth 8-Port Serial Interface Controller.
hy0	This refers to the Hyperchannel Interface controller.

User Operation Modes

Three modes of operation are available to the IRIX user: **SINGLE- USER, and MULTI-USER.**

SINGLE-USER This mode can only support one user. It is usually used by the system administrator for file system repair and maintenance and other functions where one person requires exclusive use of the computer.

- The system boots into single-user.
- The single-user has full access privileges of all files.
- Only one file system is typically available and that is *ROOT.* Other file systems could be mounted if access is required to them.

MULTI-USER This mode supports several users simultaneously.

- Users login using assigned account names. New accounts are created by the system administrator. The system is shipped with six accounts: *root, rootcsh, rootsh, guest, demos,* and, *mexdemos.*
- User has full access privileges to files within that account.
- All file systems are generally available; you can read, write, and execute yours, usually read the rest of the system.

System Shutdown

Always shut the system down using the command **reboot**.

This "syncs" the file systems.

- You must be the super-user or single-user to execute **reboot**.
- You should notify other users first using the **wall** command.
- Reboot will kill all running processes.
- Workstation control returns to the PROM monitor.

******* A WARNING *******

NEVER shut your system down using the RESET switch. If you do, the file system on disk will probably be corrupted (trashed). Reset should only be used as a last resort when IRIX is hung.

SECTION 2:Boot Exercise

This is a simple exercise that will introduce you to the subjects just discussed by this lesson - Displaying the PROM monitor help menu, booting IRIX using the defaultboot file, entering multi-user mode from single-user mode (where you entered when the boot process is complete), entering super-user mode from multi-user, and rebooting the system.

1. Check the *configuration switches* on the back panel and verify that all the switches are closed. This will cause a boot from the hard disk when you enter the PROM monitor boot command.

2. Power on your IRIS workstation using the front panel switch. You should see the initial PROM monitor message appear at your terminal:

```
IRIS (IP2 - Revision B) Monitor Version 3.0.7 December
20, 1985
Memory Size  4mb (Physical Map (1mb/bit) 0x0000000f)
Configuration Switch: 0x0000
  Multibus Window (2mb) at Megabytes 0 and 1.
  Multibus accessible memory (1mb) begins
    at Physical memory page 300,
    at Virtual address 2000000.
iris>
```

3. Enter **h** at the **iris>** prompt.
   This will cause the PROM monitor *help menu* to be displayed. (see previous page entitled PROM MONITOR HELP MENU.

4. Enter **y** at the **Continue (y or n)?:** prompt.
   This will cause the remainder of the help menu to be displayed. (see previous page entitled PROM MONITOR HELP MENU - continued.)

5. Enter **b** at the **iris>** prompt.
   This will cause the IRIX kernel to be loaded into memory. The file *defaultboot* will be searched for on the hard disk and it will boot the kernel. (remember, the configuration switches are selecting the hard disk as the boot device)

   The data, shown on the page entitled THE BOOTING PROCESS: AUTOCONFIGURATION, will flash on your screen. This is followed by the configuration data shown on the page entitled AUTOCONFIGURATION OUTPUT.

   When the prompt # appears, you will be in *single-user mode.*

6. Enter **ls** at the # prompt.
   This will display the names of all the files and directories that make up the *ROOT* (/) directory. (more about this later)

7. Enter **multi** at the # prompt.
   The following (minus the inserted instructional text) appears on your screen.

```
#
INIT: New run level: 2

Is the date [day month date time time-zone year] correct?
(y or n) y

Do you want to check filesystem consistency? (y or n) y
```

```
    /dev/si0a
    File system: root Volume: SGI

    ** Phase 1 - Check Blocks and Sizes
    ** Phase 2 - Check Pathnames
    ** Phase 3 - Check Connectivity
    ** Phase 4 - Check Reference Counts
    ** Phase 5 - Check Free List
    xxx files yyyyy K blocks zzzz K free

    /dev/rsi0f
    File system: usr Volume: SGI

    ** Phase 1 - Check Blocks and Sizes
    ** Phase 2 - Check Pathnames
    ** Phase 3 - Check Connectivity
    ** Phase 4 - Check Reference Counts
    ** Phase 5 - Check Free List
    xxx files yyyyy K blocks zzzz K free
/dev/si1g mounted on /d
/dev/si0f mounted on /usr
Resetting locks and logs
Hostname: doc
Standard daemons: syslogd lpsched.
Internet daemons: routed portmap inetd.
NFS daemons: nfsd biod.
Mailer daemons: sendmail.
More standard daemons: cron.
Preserved editor files
Cleared /tmp


doc login:
```

After the check is complete, IRIX informs you that several *daemons* were started. These are background programs that are always running. It also informed you that the */usr* and any other filesystem was mounted and that various "house-cleaning" tasks were performed.

8. Login as **root.**

9. Enter **whoami** to the **#f4 prompt.**
   The account name (*root*) will echo back to you.

10. Enter **reboot** to the **#f4 prompt.**

    You are going to *sync* the disks and re-enter the PROM monitor.

    The following message blinks on the screen, the screen is cleared and the PROM monitor message is again displayed.

```
unmounting: 0705 0700


IRIS (IP2 - Revision B) Monitor Version 3.0.7 December
20, 1985
Memory Size  4mb (Physical Map (1mb/bit) 0x0000000f)
Configuration Switch: 0x0100
  Multibus Window (2mb) at Megabytes 0 and 1.
  Multibus accessible memory (1mb) begins
    at Physical memory page 300,
    at Virtual address 2000000.
iris>
```

11. End of exercise.

Test Items

1. Give the mnemonic name for the following devices:

   dsd0 _____

   mf0 _____

   nx0 _____

   sq0 _____

   sii0 _____

   nx0 _____

Contents

PROM1 1-1
- Introduction 1
- Objectives 1
- Criterion Test 1
- Sample Test Item 1
- Resources 2
- PROM Monitor 3
- SECTION 1: The PROM Monitor 4
- BOOTING IRIX 5
- IRIS CONFIGURATION SWITCHES 6
- PROM MONITOR COMMANDS 7
- PROM MONITOR CMDS - continued 8
- PROM MONITOR COMMAND SYNTAX 9
- PROM MONITOR HELP MENU 11
- PROM MONITOR HELP MENU - continued 12
- PROM MONITOR COMMAND EXAMPLES 13
- SECTION 2: Test Items 15

PROM1

Introduction

At the lowest level of workstation control, the PROM Monitor provides functions required to boot IRIX, perform some configuration, and invoke diagnostic programs. In some respects, it is similar to the PROM Monitor in the 4D-series systems.

Objectives

At the completion of this lesson, you will be able to:

- Describe the five general functions of the PROM Monitor firmware.
- Recognize the correct syntax of the PROM Monitor program load commands.
- Match the PROM Monitor device acronyms with their correct device.
- Describe the function and usage of the PROM Monitor commands.

Criterion Test

Correctly perform the following PROM Monitor procedures:

- boot from a device other than the default.
- display the current set configuration values.
- list directory contents from a selected device.

Sample Test Item

Show the PROM Monitor command you would type to cause your workstation to boot over the network from the file *vmunix* from the remote workstation named *olympus*.

Resources

This guide

PROM Monitor

This module contains two sections:

- **SECTION 1:**

  This section describes the functions and commands available form the PROM Monitor.

- **SECTION 2:**

  This section contains test items that you should complete after studying the material in this module.

SECTION 1: The PROM Monitor

The lowest level of software (in the IRIS workstation) is contained on the CPU board; held in proms and called the **PROM MONITOR.**

- Is a dumb terminal driver.
- Has a hard disk interface.
- Reads configuration switches.
- Will boot IRIX.
- Allows limited diagnostic functions.

IRIS CONFIGURATION SWITCHES

As the table shows, switches 1 through 4 select the device from which the IRIS is to be booted. Switch 5 specifies whether the IRIS should perform an automatic boot or a PROM monitor boot. Switch 6 determines whether or not system information is displayed on the screen after the IRIS is reset. Switch 7 selects the display monitor type.

NOTE:
IRIS workstations are shipped with all the switches in the closed position.

Configuration Switches			
Switch	*Switch Name*	*Position*[1]	*Meaning*
1 - 4	Boot environment	CCCC OCCC COCC OOCC OCOC	Hard disk boot Cartridge tape boot Floppy disk boot Network boot PROM monitor
5	Autoboot	C O	PROM Monitor boot Automatic boot
6	Quiet mode	C O	Display system information Don't display system information
7	Monitor select	C O	Display on primary monitor Display on secondary monitor
8-9	Reserved	C	

IRIS Configuration Switches

BOOTING IRIX

The IRIS boot environment is determined by the setting of configuration switches 1 through 4 on the cabinet back panel. Two basic boot options are available: **AUTOMATIC BOOT** and **PROM MONITOR BOOT.**

- Automatic Boot
  At powerup, the IRIS tries to boot from a file called *defaultboot* on the device specified by the configuration switches (see table on next page).

- PROM monitor boot
  At power up, the IRIS enters the PROM monitor and waits for further boot instructions.

Since the IRIS can be booted from different devices (*hard disks, tape drives, etc.*), the PROM monitor provides the ls command, a version of the IRIX *ls(1)* command, for displaying the names of files on the attached devices.

PROM MONITOR COMMANDS

The PROM monitor offers several commands: some can be used, others are for manufacturing personal and should not be used.

Following is a list of the commands you should be familiar with. Later in the course you will get to use the commands while performing a lab project.

COMMAND	DESCRIPTION
b	Load and begin execution of a boot file. The filename can be supplied as an argument with the boot command, or if no filename is given, then a file called *defaultboot* is searched for on the boot device. The boot device can be named as an argument or if no name is given, then the configuration switches will decide what device to boot from.
n	Same as **b**, but boot is from "XNS" or "TCP" (network boot).
ls	Used to list files on the selected device.
h	Print the help message.

PROM MONITOR CMDS - continued

COMMAND	DESCRIPTION
set	Print the current set values.
set debug 0/1	Set debug mode. This is a toggle switch, it allows you to inject *noise* into the boot process. Noise is a term used by the engineers and if an error is detected during the boot, then the process is halted and an error message is displayed.
set display	Used to enter the state of the configuration switches. The following steps must be followed when you change the switches: 1. reboot. 2. Change switch settings. 3. Execute set display. 4. Depress reset.

PROM MONITOR COMMAND SYNTAX

When you are in the PROM monitor and request the help menu, the display that is printed is somewhat confusing. (The display is reproduced on the next two pages in your workbook)

The load type commands (**b, n, l**) and the list command (**ls**) may be executed with optional parameters supplied with the command. The optional parameters are: **MEDIA, DEVSPEC,** and **file.**

MEDIA: Media is a **2** or **3** character field that defines what type of device you will boot from or list files from. The available devices are: **hd, ct, fd, ip, sd, md, st, mt, sf, mf, tcp, xns,** and **rom.**

Of these commands, you will only need to use: **hd, ct, fd, tcp,** or **xns.** This is because the PROM monitor will look to see the actual type of device attached and access that device using the correct device type.

i.e. For disk drives the following mnemonics are used: **ip, sd,** or **md.** Each type of available disk drive has a different mnemonic, but, the generic mnemonic **hd** can be used to load from any of the disk types. *This would not apply if the system had two disks with the same device number (say si0 and ip0), then you would have to use the specific mnemonic to boot from the desired drive.*

This same principle applies to the tape (use **ct**) or floppy (use **fd**) devices.

DEVSPEC: Devspec will be one of the following:

hostname	Name of the host, the MEDIA must be either **tcp** or **xns.**
unit	The unit number of the device (0, 1, ...).
<unit><fs>	This is the unit number and filesystem (a-h). The MEDIA must be a disk drive.

file: This is the name of the boot file. If not given, the PROM monitor looks for the file *defaultboot*.

PROM MONITOR HELP MENU

```
iris> h
General Monitor Commands (All numeric values in hex):
 MEDIA is one of the following:
        hd          - hard disk.(look for ip, sd, then md)
        ct          - cartridge tape. (look for st, then mt)
        fd          - floppy disk. (look for sf, then mf)
        ip          - interphase disk. (474 MB drive)
        sd(or si)   - storager disk. (170 MB drive)
        md          - midas disk. (72 MB drive)
        st(or sq)   - storager cartidge tape. (1/2" tape)
        mt(or mq)   - midas cartidge tape. (1/4" tape)
        sf          - storager floppy disk.
        mf          - midas floppy disk.
        tcp         - network.
        xns         - network.
        rom         - EPROM board.
 DEVSPEC is one of the following:
        host name       - Name of the host. (MEDIA must be xns)
        unit            - unit number of device (0, 1, ...).
                          (MEDIA must be a tape or disk device)
        <unit><fs>      - unit number and filesystem (a-h)
                          (MEDIA must be a disk device.)
        address           - multibus address.
                          (MEDIA must be a EPROM board.)

[MEDIA.DEVSPEC:][file]  load and begin execution of the named file
                                    file defaults to defaultboot
                                    SPECs are from switch settings

b [MEDIA.DEVSPEC:][file]        same as above

n [DEVICE:][file]               same as b with MEDIA = xns

ls [MEDIA.DEVSPEC:][file]       list the files on the device

l [MEDIA.DEVSPEC:][file]        load but don't begin execution of the file

g address [stack]               start executing at specified address.
                                the stack address is a option.

Continue (y or n)?: y
```

PROM MONITOR HELP MENU - continued

hl?	print this help message.
set	print the current set values.
set media MEDIA	set the default boot media.
set devspec DEVSPEC	set the default boot device spec.
set debug 0/1	set the debug mode.
set display	set display options from switch settings.
set dcr BITS OPTION	set DC4 bits and option.
exit	reset the PROMS.
fm{b|w|l} ADDR VALUE [INCR] [CNT]	fill memory as byte, word or long starting at ADDR, with initial VALUE, incrementing VALUE by INCR for CNT times. (INCR defaults to 0; CNT defaults to 1)
dm{b|w|l} ADDR [CNT]	display memory as byte, word or long. (CNT defaults to 1)
em{b|w|l} ADDR	edit memory interactively as byte, word or long.
dpr	display processor registers.
epr REGISTER	edit the given processor register. (sr, vbr, cacr, caar, sfc, dfc).

iris>

One the next page are some examples of PROM monitor commands with explanations.

PROM MONITOR COMMAND EXAMPLES

Following are six examples of PROM monitor commands.

1. **b ct0:sifex**

   The above command will boot a file named *sifex* from tape drive 0.

   b = monitor command
   ct = MEDIA option
   0 = DEVSPEC
   sifex = file

2. **b hd1:vmunix1**

   This command will boot a file named *vmunix1* from disk drive 1.

   b =monitor command
   hd =MEDIA option
   1 =DEVSPEC
   vmunix1 =file

3. **n elvin:/vmunix**

   This command will boot the IRIS over the Ethernet. The host system is called *elvin* and the boot file is *vmunix*.

   n =monitor command
   elvin = DEVICE
   /vmunix = Pathname to file *vmunix*.

4. **ls ct0:**

   This command will list the file names of the first file on tape.

5. **ls hd0a:**

   This command lists the first level of directories on disk partition *a* of drive 0.

6. **b**

   This command boots the IRIS using the defaults defined by the switches.

7. set

The set command gives low-level configuration information about the IP2 board. The following is displayed:

```
iris>set
Clock is running:
   Time:                    Alarms:Date:Registers:
 12:45:14      00:00:00        01/10[3]/89      2a 0e 50 80
Saved RAM: magic 0x8161958, boot 0x1, power 0x0, dcr 0x52
Boot Environment:
  Media: hd
  Devspec: 0
Misc Environment:
  Version: 3.0.7 December 2, 1985
  Serial Number: 65026
  IP2 Revision A with 8 Meg of memory
  FPA installed
iris>
```

SECTION 2: Test Items

1. Show the PROM Monitor command you would type to cause your workstation to boot over the network from the file *vmunix* from the remote workstation named *olympus*.

2. When you boot the workstation, what file does the PROM Monitor look for if no file name is supplied as an argument to the boot command?

3. Show the PROM Monitor command to list the contents of the first file on a cartridge tape.

Contents

INTRO 1-1

Introduction 1
Objectives 1
Criterion Test 1
Resources 1
About this course 2
Course Procedures 2
Course Format 2
Criterion Tests 2
The Course Map 3
Module Selection 3
Monitoring Your Progress 5
When You Have Finished 5
A Word About Troubleshooting 5

Introduction

Objectives

At the completion of this lesson, the participant will be able to:

- State the overall objectives of the course.
- Identify the prerequisites of the course.
- State the performance requirements and how performance will be measured.
- Describe the course procedures.
- Describe the topics that will be presented during this course.

Criterion Test

Resources

Course Description

About this course

Before starting study in this course, you should quickly refer to the *Course Description* at the beginning of this guide and read the following sections:

1. Overall Course Objectives
2. Course Prerequisites
3. Performance Measurement

Course Procedures

Read this section before going any further!!

Following are the rules you should follow during this course. Please read this entire section before proceeding any further.

Course Format

This is a self-paced, self-study course.

To complete this course, you will progress a section a section at a time. Each section is called a "module".

Each module consists of one or more "objectives" which describe specifically what you will be able to do upon completion of that module. You study the module's material until you are able to achieve the objectives. Once you feel competent, you simply complete a Criterion Test. If you complete the Criterion Test successfully, you may proceed on to the next module. If not, you simply study the material until you are able to complete the test successfully.

Remember, this course is designed for your success in mind! Take as little or as long as you need to finish.

Criterion Tests

After the module objectives on the first page of each module is the Criterion Test. This will either state specifically what, and how well, you must do to show competence on a module. Sometimes, an example Criterion Test Item may be given. If so, it will be representative of the actual Criterion Test.

The Course Map

The course map is your guide to studying the modules comprising this course. Each ellipse represents one module whose title is shown inside the ellipse.

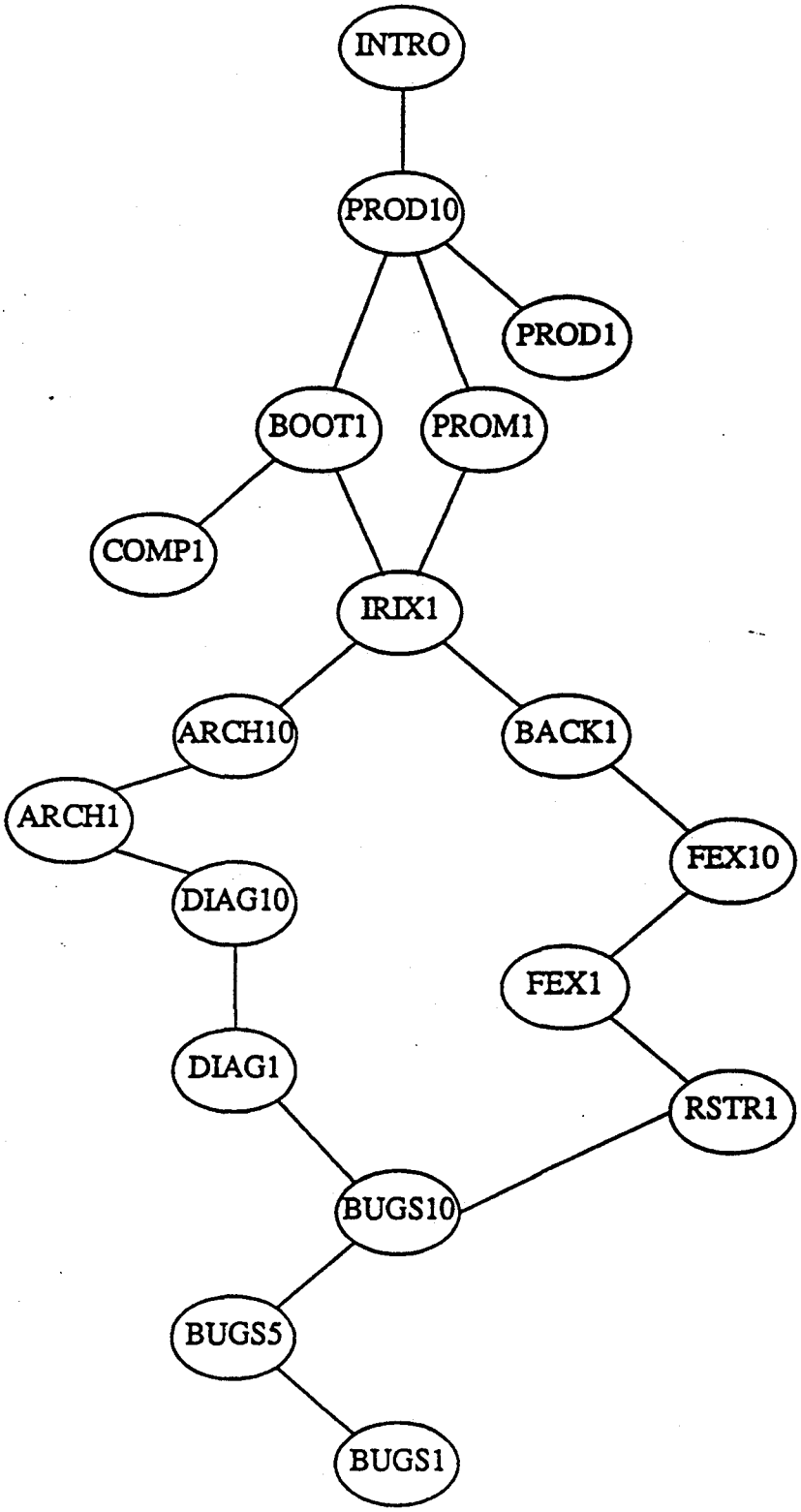
Module Selection

The lines connecting each module show you the path, or order, you must take to complete the course. Starting at the top of the map, in this module, you simply work your way down until all modules have been finished, and you're done!

Notice that some modules have more than one line leaving them. Here, you may make your own decision about which of the next modules you prefer to study first. For example, module *PROD10* has three lines leaving it to modules *BOOT1, PROM1, or PROD1*. You should complete *PROD1* after completing *PROD10* since it is the same topic area. You then have the option to select from either module *PROM1* or *BOOT1*. Before studying in module *UNIX1* you must have completed the modules leading into it.

To select a module from the course map, follow these rules:

- *Start at the top and work toward the bottom.*
- *Where more than one line leaves a module, select either one to study next.*


INTRO
PROD10
PROD1
BOOT1
PROM1
COMP1
IRIX1
ARCH10
BACK1
ARCH1
FEX10
DIAG10
FEX1
DIAG1
RSTR1
BUGS10
BUGS5
BUGS1

Monitoring Your Progress

After you complete a module Criterion Test, mark it off on the Course Map by initialing it and including the the date you completed it.

When You Have Finished

When you have completed all modules, you should complete the final review questions, and the course critique. These, along with a copy of your completed Course Map, should mailed to Technical Support Education in the enclosed envelope. A Certificate of Completion will be mailed back to you when we receive these items.

Remember to do the following upon completing the course:

- *Complete the final review questions.*
- *Complete the Course Critique form.*
- *Make a copy of your completed Course Map.*
- *Return mail these items to Technical Support Education in the enclosed envelope.*

A Word About Troubleshooting

This self-study course includes some troubleshooting problems. When you get to the "BUG" modules, your local Field Specialist will administer a series of induced system failures. You will then proceed to systematically isolate them to the FRU using the tools and techniques learned from previous modules. This technique is similar to that of the *core* course you attended at SGI headquarters.

Good Luck!!

Contents

PROD1 1-1
Introduction 1
Objectives 1
Sample Test Item 1
Resources 1
Support Documentation 3
SECTION 3: Test Items 5

PROD1

Introduction

This module describes the system documentation available with 3000-Series workstations.

Objectives

At the completion of this lesson, you will be able to:

- Given a list of document titles, match the correct one with its appropriate content.

Sample Test Item

In which volume and in which section would you find information regarding the *inittab* file?

Resources

This guide

IRIS 3000-Series Workstation Documentation set

This module contains two sections:

- **SECTION 1:**

  This section is contains general information about IRIS workstation documentation.

- **SECTION 2:**

  This section contains review questions that you should complete after studying the material in this module.

Support Documentation

There are several manuals available that support the IRIS workstations. Following is a list of the manuals that a you will most likely need.

Each system is shipped with these manuals plus some additional manuals not listed.

VOL 1A: This manual contains **CHAPTER 1** of the eight chapter programmer's manual. It contains descriptions of all the basic shell commands.

VOL 1B: This manual contains **CHAPTERS 2-8** of the eight chapter programmer's manual.

- **Chapter 2:** Contains descriptions of the system calls. These are commands that **cannot** be executed individually, but used as code in a C-program.
- **Chapter 3:** Contains descriptions of available subroutines. These **cannot** be executed individually, but used in high level compiled programs.
- **Chapter 4:** Contains descriptions of the **format and contents** of various files used in the UNIX system.
- **Chapter 5:** Contains descriptions of miscellaneous facilities such as macro packages, character set tables, etc.
- **Chapter 6:** Contains descriptions of all the UNIX games such as craps, hangman, etc.
- **Chapter 7:** Contains descriptions of special files that refer to specific hardware peripherals and UNIX System device drivers.
- **Chapter 8:** Contains some basic information concerning system administration.

VOL 2A: This manual contains information and tutorials on various UNIX features such as ED, VI, Troff, etc.

VOL 2B: This is the high level language reference manual. Contains information on various UNIX langauges and programs such as FORTRAN, Fsck, Lint, etc.

SERIES 3000: This manual contains information concerning Install, Test, System Administration, and workstation operation.

SECTION 3: Test Items

None

1. In which volume and in which section would you find information regarding the *inittab* file?

2. In which volume and in which section would you find the document describing the *fsck* utility?

3. In which volume and in which section would you find information describing the */dev/floppy* device?

Contents

PROD10 1-1
- Introduction 1
- Objectives 1
- Sample Test Item 1
- Resources 1
- SECTION 1:IRIS 3000-Series Architecture 3
  - IRIS WORKSTATION SUBSYSTEMS 3
  - IRIS WORKSTATION OVERVIEWS 4
  - 2400 Turbo Specifications 5
  - 3010 Specifications 6
  - 3020 Specifications 7
  - 3030 Specifications 8
  - Workstation Software 9
  - Basic System Block Diagram 10
  - Workstation Summaries 11
- SECTION 3: Test Items 13

PROD10

Introduction

This module is basically a compilation of the features of 3000-Series workstations. In it you will find tables comparing the major configuration and capacities for each workstation supported by SGI. Included are tables for the 2000-Series and 1000-Series workstations and terminals.

Objectives

At the completion of this lesson, you will be able to:

- Recognize the major system software components.
- Given a blank block diagram, correctly label the major circuit board assemblies with their correct designation.
- Identify each IRIS 3000 subsystem and configuration.
- Differentiate the features of the various models in the IRIS 3000-Series product line.

Sample Test Item

On an unlabeled block diagram, correctly label each block with the correct component for any 3000-Series workstation.

Resources

This guide

This module contains three sections:

- **SECTION 1:**

  This section is contains general information about IRIS workstation architecture.

- **SECTION 2:**

  This section contains tables comparing the major features of the workstations in the 3000, 2000, and 1000-Series product lines.

- **SECTION 3:**

  This section contains review questions that you should complete after studying the material in this module.

SECTION 1:IRIS 3000-Series Architecture

IRIS WORKSTATION SUBSYSTEMS

The data flow portion of an IRIS workstation is divided into four subsystems. Each subsystem is comprised of one or more printed circuit boards (PCB).

- 68020 Processor
- Memory (up to 16 MB)
- Floating Point Accelerator (optional)

- 2 Geometry Accelerators
- 12 Geometry Engines

- Framebuffer Controller
- Update Controller
- Display Controller
- Display Memory (up to 8 boards)

- Tape/Disk Controller
- Ethernet Controller
- Hyper-Channel (optional)
- IEEE-488 (optional)
- IBM 327X Interface (optional)
- Color Printer Controller (optional)

IRIS WORKSTATION OVERVIEWS

The following gives general information about each of the IRIS products that use the 68020 CPU. The next four pages in your student workbook give the total specifications for all four workstations.

The IRIS 3010 Terminal is a high-performance, high-resolution, UNIX-based terminal for real-time 3-D color graphics and computing applications. The IRIS Terminal utilizes the 32-bit Motorola 68020 CPU and can function as either a **host-dependent** terminal or an **execute-only** workstation. When functioning as a host-dependent terminal, the applications program resides in the host computer system. The IRIS 3010 can be linked to an IRIS Workstation or a variety of other host systems via an RS-232C link or optionally via Ethernet. As an execute-only workstation, programs can be developed, debugged, and tested on an IRIS 3020 Workstation or an IRIS 3030 Workstation and then loaded and executed, without modification, on a stand-alone 3010 Terminal.

These workstations are high-performance, UNIX-based engineering workstations with a 72 MB disk drive and controller designed for real-time 3-D color graphics and computing applications. They utilize the 32-bit Motorola 68020 CPU and can operate as a stand-alone personal workstation, a node in a networked workstation environment, or integrated with other computer systems via Ethernet.

The 2400 Turbo is an upgrade of the IRIS 2400 workstation. It is the same system as the IRIS 3020 except for the following areas: Cabinet, Power supply, Chassis, and Monitor.

The IRIS 3030 workstation is the same as the 3020, but it has a 170 MB disk drive.

2400 Turbo Specifications

NOTE:
The Turbo option is an upgrade to a 68020 CPU. Many 2000-Series workstations have been upgraded to the 68020, most notably the 2400T. The specifications below describe the Turbo option.

System Specifications

Processors:
- 16 MHz MC68020 central processor 32-bit internal registers, 32-bit address space
- 10 125-nanosecond Geometry Engines
- 16-bit bit-slice frame buffer controller Independent microcoded display processor with 32 KB memory for fonts, textures, and cursors

CPU Memory:
- 2 MB dynamic RAM with parity error detection, expandable to 16 MB (Expandable to 12 MB on IRIS 2300)
- 256 KB EPROM for hardware initialization, self-configuration, and diagnostics

Communications:
- Four RS-232C ports for keyboard and serial communications (up to 19.2K baud)

Standard Software:
- UNIX System V operating system with Berkeley 4.2 and local enhancements
- UNIX kernel with demand paging and IRIS terminal software (IRIS 2300)
- Extent File System
- C compiler and development environment (IRIS 2400 and IRIS 2500)
- IRIS Graphics Library II™
- IRIS Window Manager™

Options:

Hardware:
- Z clipping
- 2 or 4 MB CPU memory
- Floating point accelerator

Software:
- FORTRAN and Pascal compilers
- Terminal software
- EMACS text editor
- GKS library, level 2b

3010 Specifications

System Specifications

Processors:
- 16 MHz MC68020 central processor 32-bit internal registers, 32-bit address space
- 10 125-nanosecond Geometry Engines
- 16-bit bit-slice frame buffer controller Independent microcoded display processor with 32 KB memory for fonts, textures, and cursors

CPU Memory:
- 4 MB dynamic RAM with parity error detection, expandable to 12 MB
- 256 KB EPROM for hardware initialization, self-configuration, and diagnostics, expandable to four 256/512 KB EPROMS

Image Memory:
- 8 1024 x 1024 bit-planes standard, expandable to 32 with 16-bit Z-buffer

Video Interface:
- RGB levels 0.7 V p-p into 75 ohms
- Separate composite 2 V p-p sync into 75 ohms
- 60 Hz non-interlaced 1024 x 768 resolution frame
- Other frame resolutions and rates available
  - 33 Hz interlaced 1024 x 768
  - 30 Hz interlaced 636 x 485
  - 25 Hz interlaced 780 x 575
- Genlock available with 485 and 575 visible line frames

Color Range:
- Color map mode (8- or 12-bit, single or double buffered)
- 4096 simultaneous colors, double buffered, displayable from palette of 16.7M
- RGB mode (24-bit), 16.7 million colors displayable

Standard Peripherals:
- 3½" Winchester disk, 22 MB unformatted, using an ST-506 interface
- 1 MB 5.25" floppy disk drive
- 83-key up-down encoded keyboard with user definable keys
- 19" diagonal non-interlaced RGB tilt and swivel monitor
- Optical mouse X-Y encoder with three buttons

Communications:
- Four RS-232C ports for keyboard and serial communication (up to 19.2K baud)

Standard Software:
- IRIS Graphics Library II
- UNIX kernel with demand paging and IRIS Terminal software
- IRIS Window Manager

Chassis:
- 20-slot Multibus™ card cage
- 720 watt power supply

Options

Hardware:
- Z-clipping
- 2 or 4 MB CPU memory cards
- 4 bit-plane image memory cards
- Floating point accelerator
- 60" rack mounted chassis

Peripheral:
- 11" x 11" digitizer tablet
- Dial and button box
- Color printer and controller
- 15" diagonal non-interlaced RGB tilt and swivel monitor
- 19" diagonal 30 Hz interlaced RGB monitor

Communication:
- Ethernet with XNS or TCP/IP software
- IBM Link for 3278-9 emulation and file transfer

Physical and Environmental Specifications

Power Requirements:
- AC voltage 93-132 or 186-264 VAC (factory set)
- AC frequency 47-63 Hz
- Chassis: 1250 VA, 1000 W, 3410 BTU/hr
- 19" monitor: 225 VA, 150 W, 512 BTU/hr

Size and Weight:
- 19" monitor: 18.5"H x 20"W x 21.5"D (51 x 48 x 54 cm), 84 lb. (38 Kg)
- Chassis: 29"H x 18"W x 27"D (74 x 46 x 69 cm) 190 lb. (86 Kg)

Environment:
- Operating: 50-86°F (10-30°C), 20-80% relative humidity, no condensation
- Shipping/storage: 32-122°F (0-50°C), 10-90% relative humidity, no condensation

LT-3010-02 Printed in U.S.A. 6-86

3020 Specifications

System Specifications

Processors:
- 16 MHz MC68020 central processor 32-bit internal registers, 32-bit address space
- 10 125-nanosecond Geometry Engines
- 16-bit bit-slice frame buffer controller Independent microcoded display processor with 32 KB memory for fonts, textures, and cursors

CPU Memory:
- 4 MB dynamic RAM with parity error detection, expandable to 16 MB
- 256 KB EPROM for hardware initialization, self-configuration, and diagnostics, expandable to four 256/512 KB EPROMS

Image Memory:
- 8 1024 x 1024 bit-planes standard, expandable to 32 with 16-bit Z-buffer

Video Interface:
- RGB levels 0.7 V p-p into 75 ohms
- Separate composite 2 V p-p sync into 75 ohms
- 60 Hz non-interlaced 1024 x 768 resolution frame
- Other frame resolutions and rates available
  - 33 Hz interlaced 1024 x 768
  - 30 Hz interlaced 636 x 485
  - 25 Hz interlaced 780 x 575
- Genlock available with 485 and 575 visible line frames

Color Range:
- Color map mode (8- or 12-bit, single or double buffered)
- 4096 simultaneous colors, double buffered, displayable from palette of 16.7 million
- RGB mode (24-bit), 16.7 million colors displayable

Standard Peripherals:
- 72 MB unformatted 5.25" Winchester disk drive using an ST-506 interface
- 83-key up-down encoded keyboard with user definable keys
- 19" diagonal 60 Hz non-interlaced RGB tilt and swivel monitor
- Optical mouse X-Y encoder with three buttons

Communications:
- Ethernet local area network with XNS software
- Four RS-232C ports for keyboard and serial communications (up to 19.2K baud)

Standard Software:
- UNIX System V operating system with Berkeley 4.2 and local enhancements
- C compiler and development environment
- IRIS Graphics Library II
- IRIS Window Manager

Chassis
- 20-slot Multibus™ card cage
- 720 watt power supply

Options

Hardware:
- Z clipping
- 2 or 4 MB CPU memory cards
- 4 bit-plane image memory cards
- Floating point accelerator
- 60" rack mounted chassis

Peripheral:
- Floppy disk drive
- Second 72 MB or 170 MB Winchester disk drive
- 60 MB ¼" cartridge tape drive
- ½" tape drive and controller
- Color printer and controller
- 11" x 11" digitizer tablet
- Dial and button box
- Programming Terminal
- 19" diagonal 30 Hz interlaced RGB monitor
- 15" diagonal 60 Hz non-interlaced RGB tilt and swivel monitor

Software:
- FORTRAN and Pascal compilers
- Terminal software
- EMACS text editor
- GKS library, level 2b

Communication:
- TCP/IP Ethernet software
- IBM link for 3278-9 emulation and file transfer

Physical and Environmental Specifications

Power Requirements:
- AC voltage 93-132 or 186-264 VAC (factory set)
- AC frequency 47-63 Hz
- Chassis: 1250 VA, 1000 W, 3410 BTU/hr
- 19" monitor: 225 VA, 150 W, 512 BTU/hr

Size and Weight:
- 19" monitor: 18.5" H x 20" W x 21.5" D (51 x 48 x 54 cm), 84 lb. (38 Kg)
  Chassis: 29" H x 18" W x 27" D (74 x 46 x 69 cm), 190 lb. (86 Kg)

Environment:
- Operating: 50-86°F (10-30°C), 20-80% relative humidity, no condensation.
- Shipping/storage: 32-122°F (0-50°C), 10-90% relative humidity, no condensation.

DS-3020-02 Printed in U.S.A. 6 86

3030 Specifications

System Specifications

Processors:
- 16 MHz MC68020 central processor, 32-bit internal registers, 32-bit address space
- 10 125-nanosecond Geometry Engines
- 16-bit bit-slice frame buffer controller Independent microcoded display processor with 32 KB memory for fonts, textures, and cursors

CPU Memory:
- 4 MB dynamic RAM with parity error detection, expandable to 16 MB
- 256 KB EPROM for hardware initialization, self-configuration, and diagnostics, expandable to four 256/512 KB EPROMS

Image Memory:
- 8 1024 x 1024 bit-planes standard, expandable to 32 with 16-bit Z-buffer

Video Interface:
- RGB levels 0.7 V p-p into 75 ohms
- Separate composite 2 V p-p sync into 75 ohms
- 60 Hz non-interlaced 1024 x 768 resolution frame
- Other frame resolutions and rates available
  - 33 Hz interlaced 1024 x 768
  - 30 Hz interlaced 636 x 485
  - 25 Hz interlaced 780 x 575
- Genlock available with 485 and 575 visible line frames

Color Range:
- Color map mode (8- or 12-bit, single or double buffered)
- 4096 simultaneous colors, double buffered, displayable from palette of 16.7 million
- RGB mode (24-bit), 16.7 million colors displayable

Standard Peripherals:
- 170 MB unformatted 5.25" Winchester disk drive, using an ESDI interface
- 83-key up-down encoded keyboard with user definable keys
- 19" diagonal 60 Hz non-interlaced RGB tilt and swivel monitor
- Optical mouse X-Y encoder with three buttons

Communications:
- Ethernet local area network with XNS software
- Four RS-232C ports for keyboard and serial communications (up to 19.2K baud)

Standard Software:
- UNIX System V operating system with Berkeley 4.2 and local enhancements
- C compiler and development environment
- IRIS Graphics Library II
- IRIS Window Manager

Chassis
- 20-slot Multibus™ card cage
- 720 watt power supply

Options

Hardware:
- Z clipping
- 2 or 4 MB CPU memory cards
- 4 bit-plane image memory cards
- Floating point accelerator
- 60" rack mounted chassis

Peripheral:
- Floppy disk drive
- Second 170 MB Winchester disk drive
- 60 MB ¼" cartridge tape drive
- ½" tape drive and controller
- Color printer and controller
- 11" x 11" digitizer tablet
- Dial and button box
- Programming Terminal
- 19" diagonal 30 Hz interlaced RGB monitor
- 15" diagonal 60 Hz non-interlaced RGB tilt and swivel monitor

Software:
- FORTRAN and Pascal compilers
- Terminal software
- EMACS text editor
- GKS library, level 2b

Communication:
- TCP/IP Ethernet software
- IBM link for 3278-9 emulation and file transfer

Physical and Environmental Specifications

Power Requirements:
- AC voltage 93-132 or 186-264 VAC (factory set)
- AC frequency 47-63 Hz
- Chassis: 1250 VA, 1000 W, 3410 BTU/hr
- 19" monitor: 225 VA, 150 W, 512 BTU/hr

Size and Weight:
- 19" monitor: 18.5" H x 20" W x 21.5" D (51 x 48 x 54 cm), 84 lb. (38 Kg)
  Chassis: 29" H x 18" W x 27" D (74 x 46 x 69 cm), 190 lb. (86 Kg)

Environment:
- Operating: 50-86°F (10-30°C), 20-80% relative humidity, no condensation.
- Shipping/storage: 32-122°F (0-50°C), 10-90% relative humidity, no condensation.

DS-3030-02 Printed in U.S.A. 6/86

Workstation Software

- Hard Disk Interface
- Initial System Diagnostics
- Initial System Configuration
- Booting UNIX

- Berkeley 4.2 Enhancements
- Local Enhancements

C COMPILER

WINDOW MANAGER

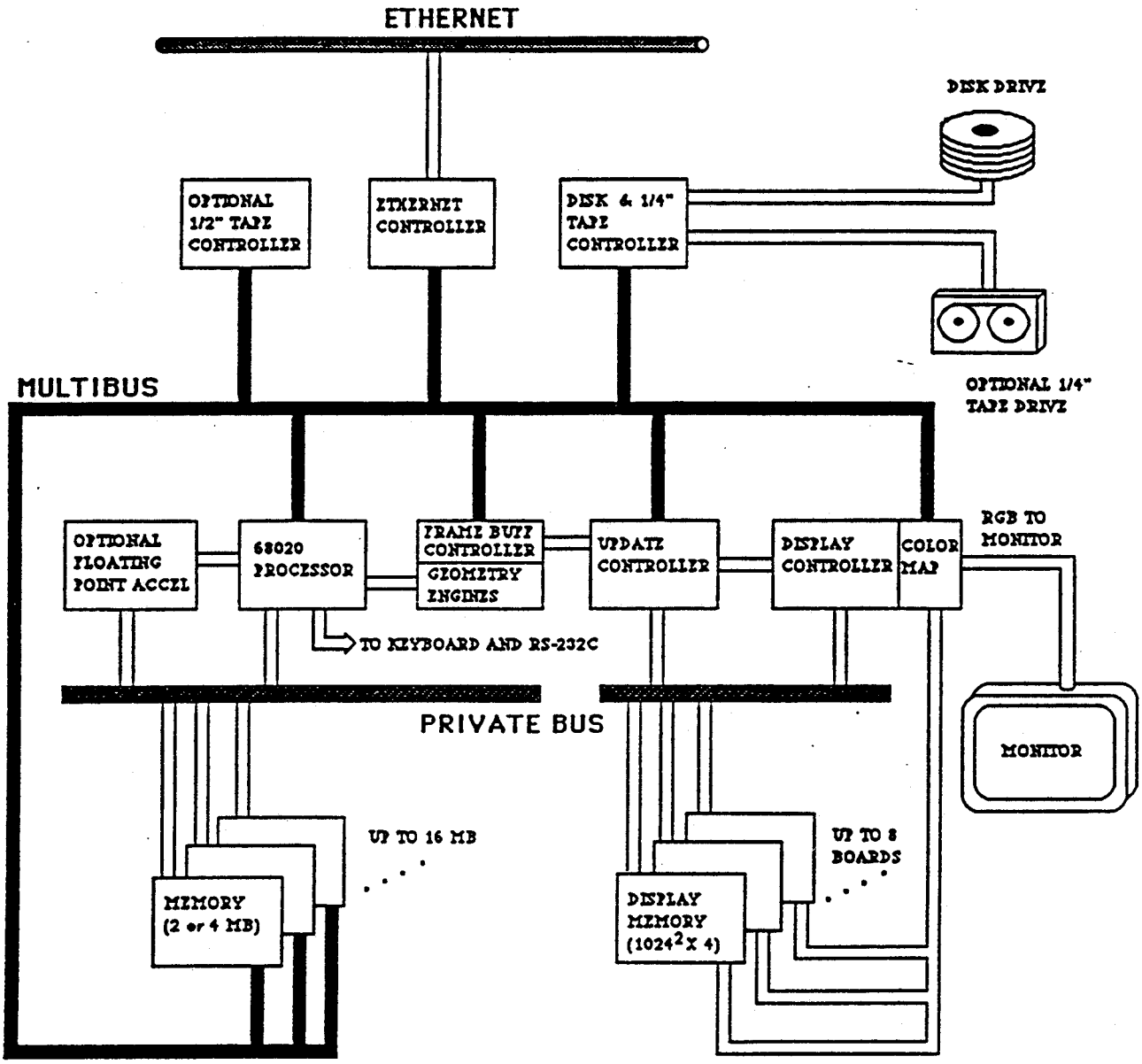
FORTRAN and PASCAL COMPILERS (optional)

EMACS TEXT EDITOR

TCP/IP ETHERNET (optional with release 3.5, standard on 3.6)

IBM 327X EMULATION (optional)

Basic System Block Diagram



Workstation Summaries

Exhibit 1. IRIS 3000-Series Workstation Summary

Model	3010	3020	3030	3110	3115	3120(B)	3130
CPU	68020	68020	68020	68020	68020	68020	68020
Speed	16Mhz	16Mhz	16Mhz	16Mhz	16Mhz	16Mhz	16Mhz
Memory	4.0MB	4.0MB	4.0MB	4.0MB	4.0MB	4.0MB	8.0MB
Bitplanes	8	8	8	8	8	8	32
GEs	10	10	10	12	12	12	12
Speed	8Mhz	8Mhz	8Mhz	10Mhz	10Mhz	10Mhz	10Mhz
Disk	22Mb	71Mb	170Mb	22Mb	72Mb	72MB(Note 1)	170Mb
Disk Intf	ST506	ST506	SMD	ST506	ST506	ESDI(Note 2)	ESDI
Monitor	60Hz T/S	60Hz T/S	60Hz T/S	60Hz T/S	60Hz T/S	60Hz T/S	60Hz T/S
Mouse	Optical	Optical	Optical	Optical	Optical	Optical	Optical
Cardcage	20 slot	20 slot	20 slot	20 slot	20 slot	20 slot	20 slot
Multibus	Parallel	Parallel	Parallel	Parallel	Parallel	Parallel	Parallel
Pwr Supply	720 W	720 W	720 W	720 W	720 W	720 W	720 W

Exhibit 2. IRIS 2000-Series Workstation Summary

Model	2000	2200	2300	2400	2500	2300T(Note 1)	2400T(Note 1)	2500T(Note 1)
CPU	68010	68010	68010	68010	68010	68020	68020	68020
Speed	10Mhz	10Mhz	10Mhz	10Mhz	10Mhz	16Mhz	16Mhz	16Mhz
Memory	1.5MB	1.5MB	1.5MB	1.5MB	1.5MB	4.0MB	4.0MB	4.0MB
Bitplanes	8	8	8	8	8	8	8	8
GEs	10	10	10	10	10	10	10	10
Speed	8Mhz	8Mhz	8Mhz	8Mhz	8Mhz	8Mhz	8Mhz	8Mhz
Disk	none	none	22Mb	72Mb	474Mb	22MB	72Mb	474Mb
Disk Intf	none	none	ST506	ST506	SMD	ST506	ST506	SMD
Monitor	60Hz N/I	60Hz N/I	60Hz N/I	60Hz N/I	60Hz N/I	60Hz N/I	60Hz N/I	60Hz N/I
Mouse	Mech	Mech	Mech	Mech	Mech	Mech	Mech	Mech
Cardcage	10 slot	20 slot	20 slot	20 slot	Rack mount	20 slot	20 slot	Rack mount
Multibus	Serial	Serial	Serial	Serial	Serial	Serial	Serial	Serial
Pwr Supply	300 W	600 W	600 W	600 W	600 W	600 W(Note 2	600 W(Note 2	600 W(Note 2

Exhibit 3. IRIS 1000-Series Workstation Summary

Model	1000	1200	1400	1500
CPU	68000	68000	68010	68010
Speed	8Mhz	8Mhz	10Mhz	10Mhz
Memory	0.75MB	0.75MB	1.5MB	1.5MB
Bitplanes	8	8	8	8
GEs	10	10	10	10
Speed	4Mhz	4Mhz	6Mhz	6Mhz
Disk	none	none	72Mb	474Mb
Disk Intf	none	none	ST506	SMD
Monitor	30Hz Interlaced	30Hz Interlaced	30Hz Interlaced	30Hz Interlaced
Mouse	Mechanical	Mechanical	Mechanical	Mechanical
Cardcage	10 slot	20 slot	20 slot	Rack mount
Multibus	Serial	Serial	Serial	Serial
Pwr Supply	300 W	600 W	600 W	600 W

SECTION 3: Test Items

On the diagram below, label the unlabeled blocks:

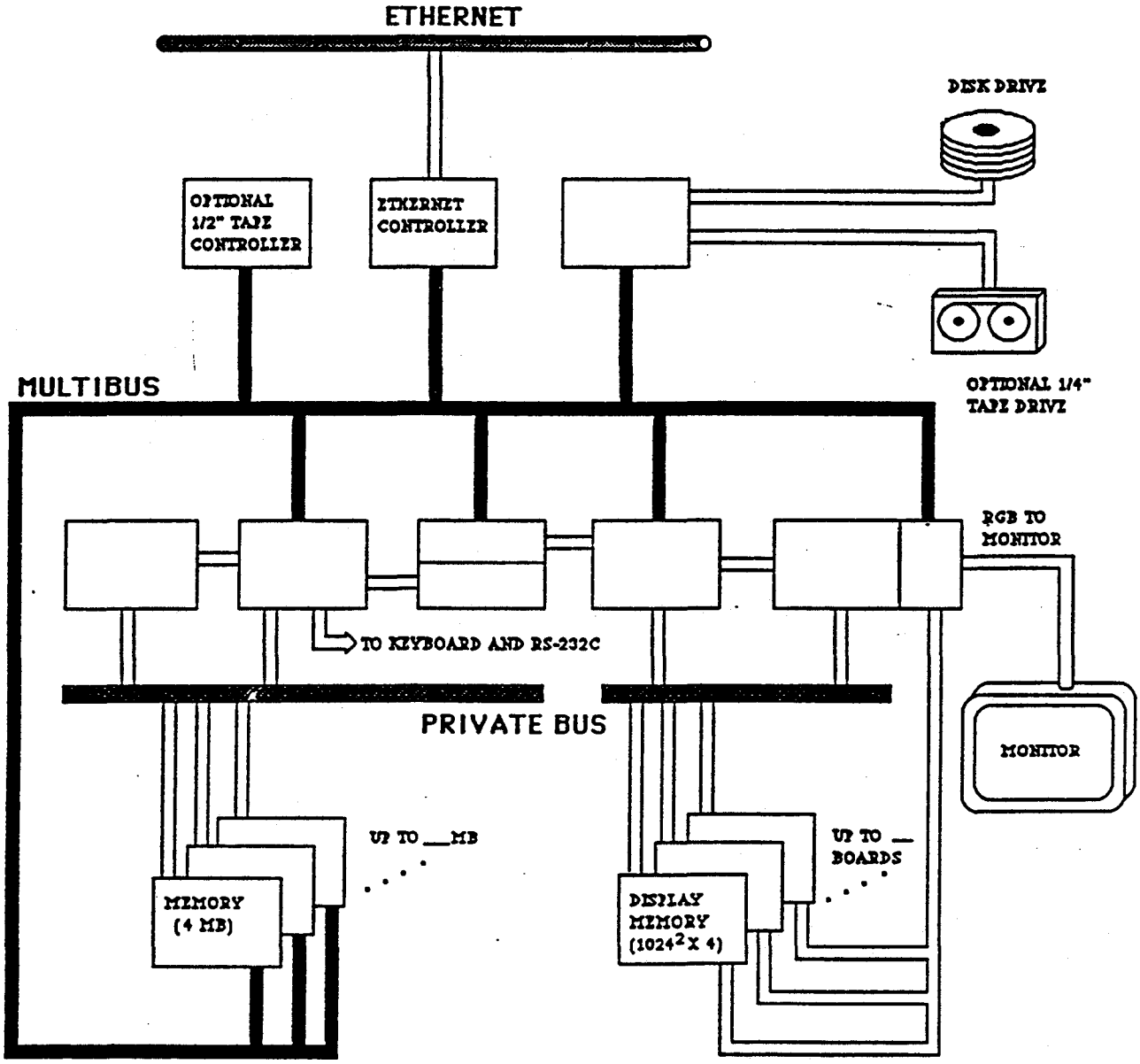


DIAGRAM 3: Basic 68020 PCB assignments - page 2

Processor Side
(without Floating Point Accelerator)

SLOT	1	2	3	4	5	6	7	8	9
	IM1	IP2	ENET	DSD	(IEEE)				(CG1)
	(IM1)	IM1	IP2	ENET	DSD	(IEEE)			(CG1)
	(IM1)	(IM1)	IM1	IP2	ENET	DSD	(IEEE)		(CG1)
	(IM1)	(IM1)	(IM1)	IM1	IP2	ENET	DSD	(IEEE)	(CG1)

( ) Indicates an Option

Processor Side
(with Floating Point Accelator)

SLOT	1	2	3	4	5	6	7	8	9
	IM1	(FP1)	IP2	ENET	DSD	(IEEE)			(CG1)
	(IM1)	IM1	(FP1)	IP2	ENET	DSD	(IEEE)		(CG1)
	(IM1)	(IM1)	IM1	(FP1)	IP2	ENET	DSD	(IEEE)	(CG1)
	(IM1)	(IM1)	(IM1)	IM1	(FP1)	IP2	ENET	DSD	(CG1)

( ) Indicates an Option

Graphics Side

Slot	10	11	12	13	14	15	16	17	18	19	20
							BP3	BP3	DC4	UC4	GF2
						(BP3)	BP3	BP3	DC4	UC4	GF2
					(BP3)	(BP3)	BP3	BP3	DC4	UC4	GF2
				(BP3)	(BP3)	(BP3)	BP3	BP3	DC4	UC4	GF2
			(BP3)	(BP3)	(BP3)	(BP3)	BP3	BP3	DC4	UC4	GF2
		(BP3)	(BP3)	(BP3)	(BP3)	(BP3)	BP3	BP3	DC4	UC4	GF2
	(BP3)	(BP3)	(BP3)	(BP3)	(BP3)	(BP3)	BP3	BP3	DC4	UC4	GF2

DIAGRAM 4: IP2 Interconnect - page 1

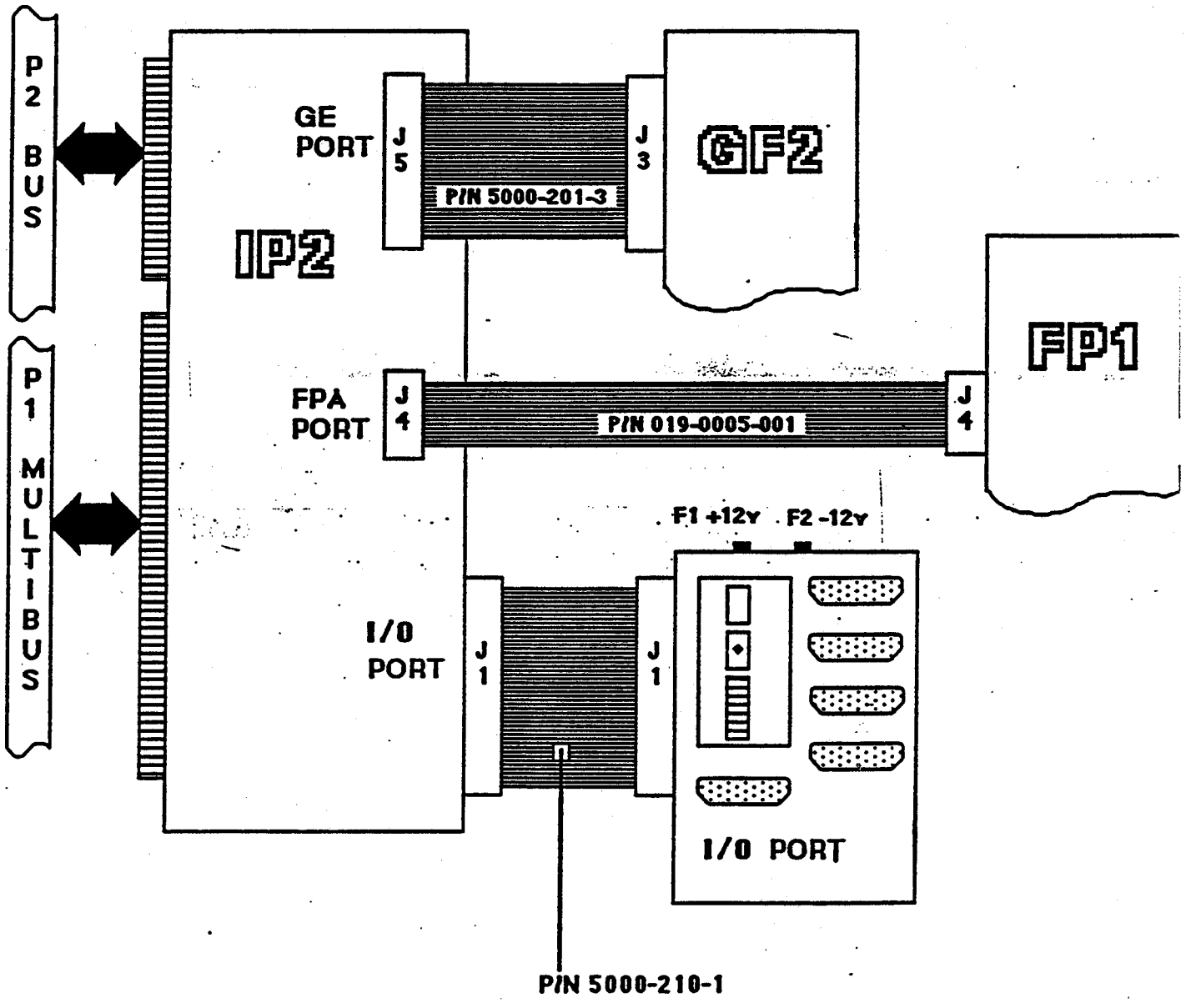
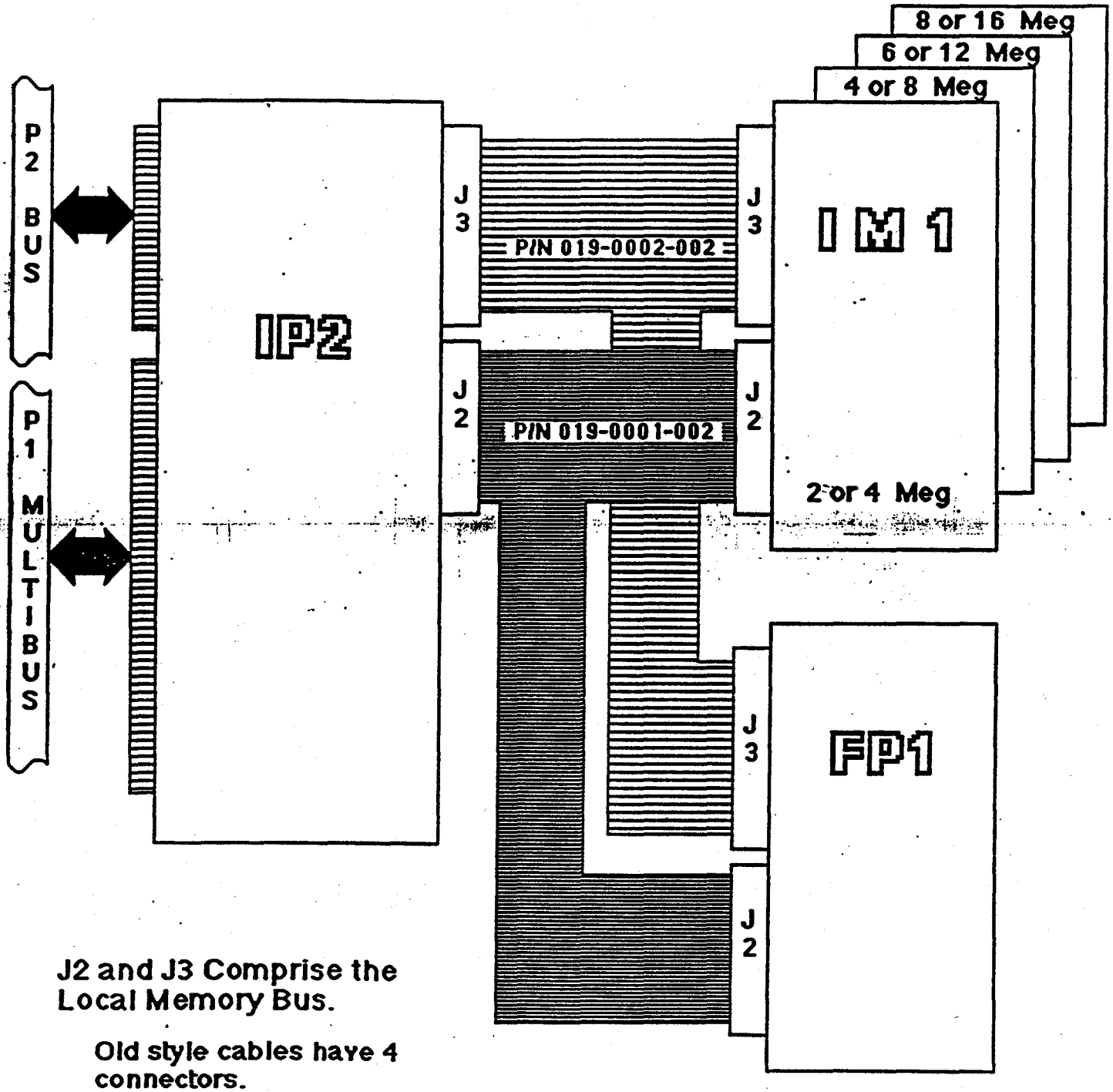


DIAGRAM 5: IP2 Interconnect - page 2



J2 and J3 Comprise the Local Memory Bus.

Old style cables have 4 connectors.

New style cables have 6 connectors.

Note: Only a system with 4 memory boards and an FP1 will use all connectors.

DIAGRAM 6: Ethernet Interconnect

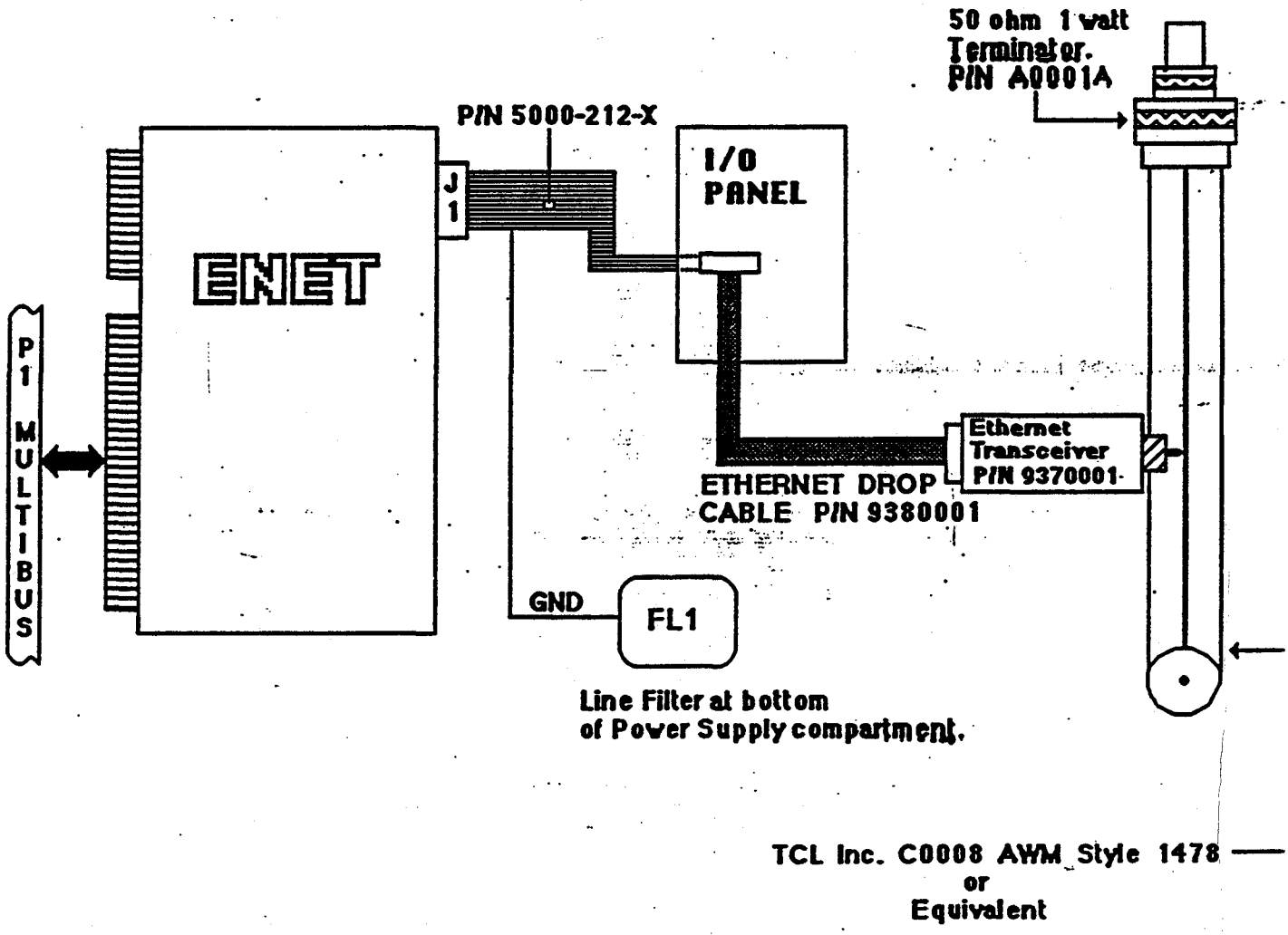


DIAGRAM 7: DSD interconnect

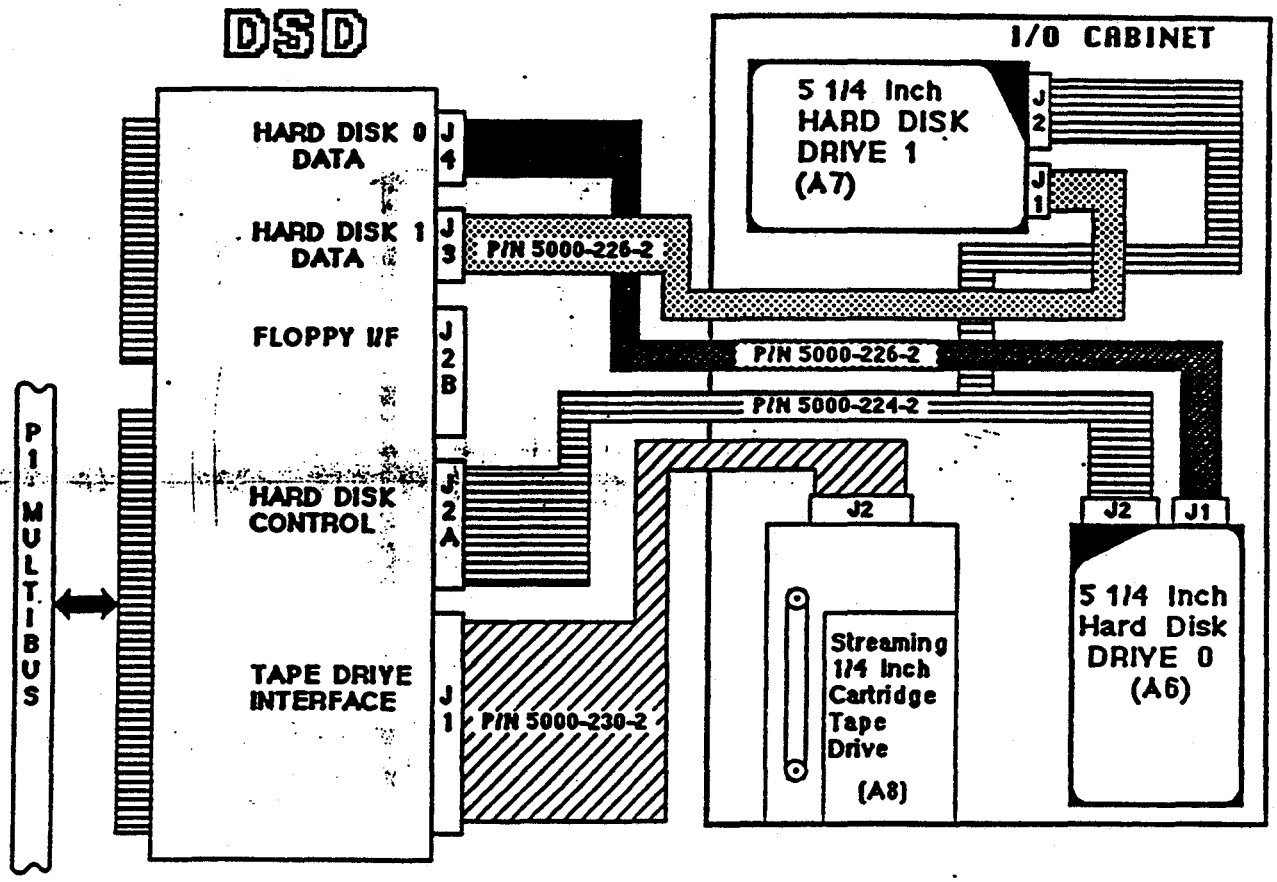


DIAGRAM 8: Graphics sub-system interconnect

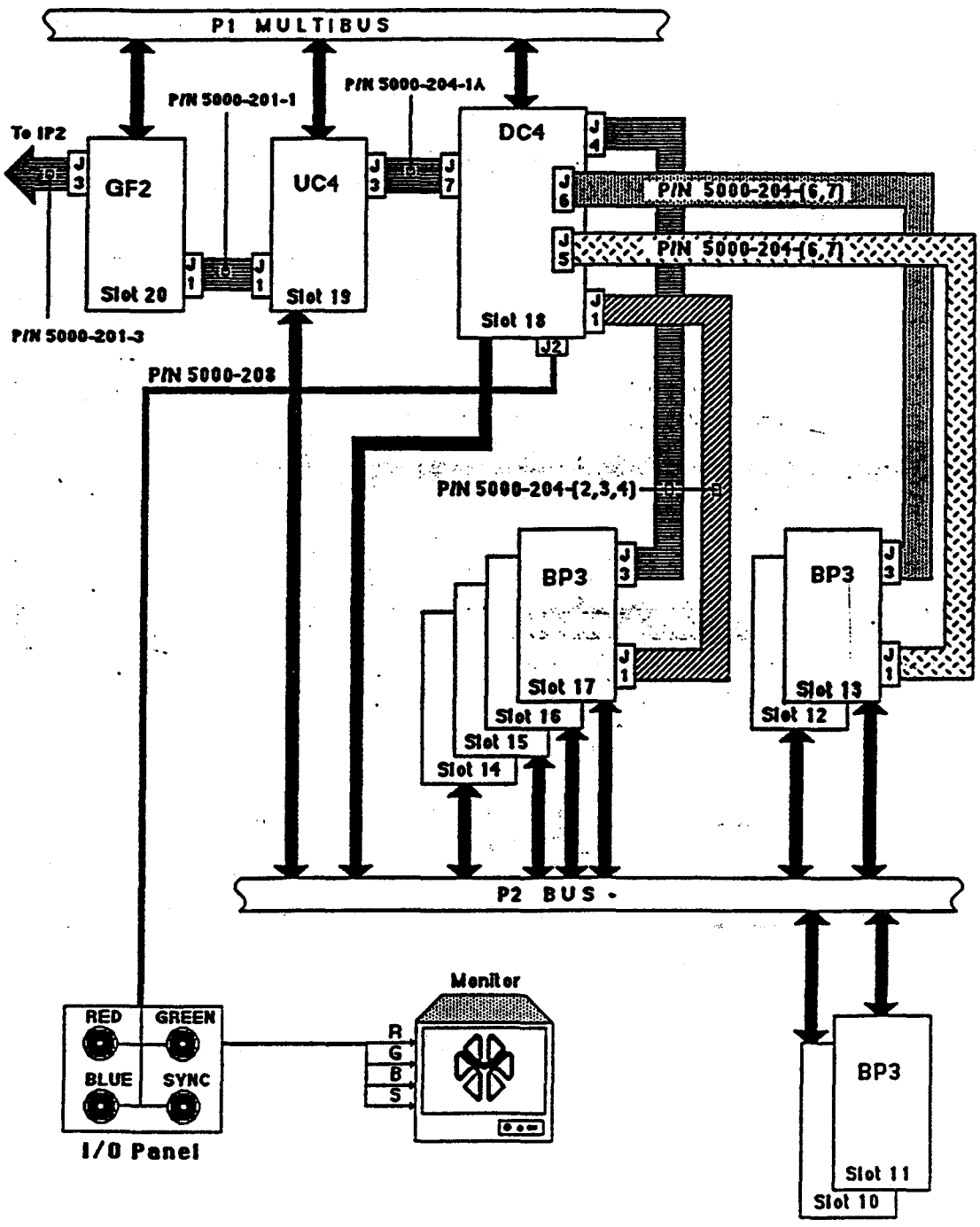


DIAGRAM 9: PCB switch settings - page 1

Memory Switches: The switch pack is located on the front edge of each board. Each pack contains four switches.

		Switch			
		1	2	3	4
Board	1	C	C	C	C
	2	C	C	C	O
	3	C	C	O	C
	4	C	C	O	O

"C" = Closed
"O" = Open

Bit Plane Switches: The switch pack is located on the front edge of each board. There are four switches per pack, and the setting produces a hex address starting at 8.

		Switch			
Addr	Slot	1	2	3	4
8	17	O	C	C	C
9	16	O	C	C	O
A	15	O	C	O	C
B	14	O	C	O	O
C	13	O	O	C	C
D	12	O	O	C	O
E	11	O	O	O	C
F	10	O	O	O	O

"C" = Closed
"O" = Open

DIAGRAM 10: PCB switch settings - page 2

IP2 - Switch S2

Category	Switch setting 1	2	3	4	5	6	7	8	Meaning
master/slave	C	X	X	X	X	X	X	X	Master processor
	O	X	X	X	X	X	X	X	Slave processor
RS232 Speed for ports 1-3	X	X	X	C	C	X	X	X	9600 baud
	X	X	X	C	O	X	X	X	300 baud
	X	X	X	O	C	X	X	X	1200 baud
	X	X	X	O	O	X	X	X	19.2K baud
Display types Primary/Secondary	X	X	X	X	X	C	C	C	NI/NI (DC4:5000-090-03)
	X	X	X	X	X	C	C	O	NI/I (DC4:5000-090-03)
	X	X	X	X	X	C	O	C	NI/RS (DC4:5000-090-02)
	X	X	X	X	X	C	O	O	NI/EU (DC4:5000-090-05)
	X	X	X	X	X	O	C	C	I/NI (DC4:5000-090-03)
	X	X	X	X	X	O	C	O	I/I (DC4:5000-090-03)
	X	X	X	X	X	O	O	C	I/RS (DC4:5000-090-04)
	X	X	X	X	X	O	O	O	I/EU (DC4:5000-090-06)

NI-non-interlaced 60 Hz
I-interlaced 30 Hz
RS-RS170A
EU-European

DIAGRAM 11: PCB switch settings - page 3

IP2 - Switch S1

Category	Switch setting 1 8	2 7	3 6	4 5	5 4	6 3	7 2	8 1	(S1) (BACK)	Meaning
Boot Environment	X	X	X	X	C	C	C	C		(0) disk boot
	X	X	X	X	C	C	C	O		(1) tape boot
	X	X	X	X	C	C	O	C		(2) floppy boot
	X	X	X	X	C	C	O	O		(3) network boot
	X	X	X	X	C	O	C	C		(4) not used
	X	X	X	X	C	O	C	O		(5) PROM monitor (ie no device)
	X	X	X	X	C	O	O	C		(6) Diagnostic PROM board boot
	X	X	X	X	C	O	O	O		(7) not used
	X	X	X	X	O	C	C	C		(8) not used
	X	X	X	X	O	C	C	O		(9) ip - interphase boot
	X	X	X	X	O	C	O	C		(a) st - storager 2 tape boot
	X	X	X	X	O	C	O	O		(b) sf - storager 2 floppy boot
	X	X	X	X	O	O	C	C		(c) sd - storager 2 disk boot
	X	X	X	X	O	O	C	O		(d) mt - DSD tape boot
	X	X	X	X	O	O	O	C		(e) mf - DSD floppy boot
	X	X	X	X	O	O	O	O		(f) md - DSD disk boot
Autoboot	X	X	X	C	X	X	X	X		don't autoboot
	X	X	X	O	X	X	X	X		autoboot
Quiet	X	X	C	X	X	X	X	X		print hardware information
	X	X	O	X	X	X	X	X		don't print information
Display	X	C	X	X	X	X	X	X		Use primary monitor for display
	X	O	X	X	X	X	X	X		Use secondary monitor

DIAGRAM 12: PCB switch settings - page 4

Disk Drives

The following illustrations show you where to locate the address jumpers for the different types of disk drives found in SGI workstations. If you replace a disk drive, insure all the other jumpers on the new drive match the drive that you remove.

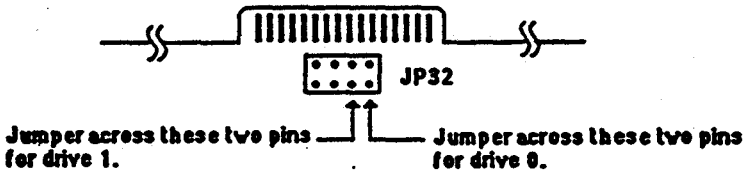
Vertex 72 MB Drive

Address selection is controlled by switch pack "J6" on the logic PCB located on the under side of the disk drive. J6 is located near and between the two edge pin connectors.

For drive 0: Switch 4 is on, switch 7 is off, and switch 8 is on.
For drive 1: Switch 4 is on, switch 7 is on, and switch 8 is off.

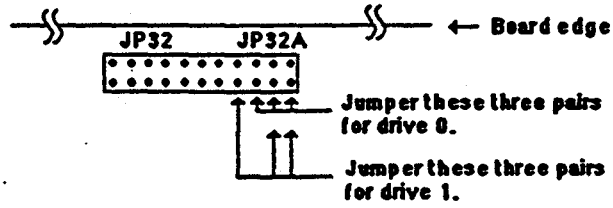
Hitachi 85 MB Drive

Address selection is done using jumpers across pins at jumper platform "JP32". This platform is located near the large pin edge connector. The following illustration shows how to set the jumpers.



Hitachi 170 MB Drive

For this drive address selection is set using jumpers on jumper platform JP32A. It is located along the edge of the logic PCB next to JP32. The following illustration shows what pins are used.



Fujitsu 170 MB Drive

For this drive address selection is set using jumpers on a jumper platform that has no name. It has 16 pairs of pins and is located at the end of the logic board that is away from the edge pin connectors. It is next to a large square chip. See following illustration for desired pin connections.

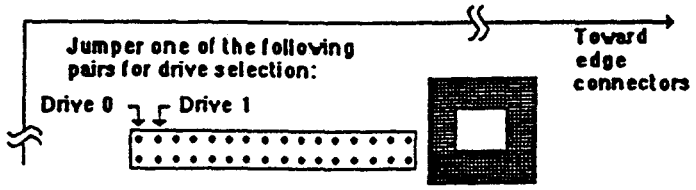
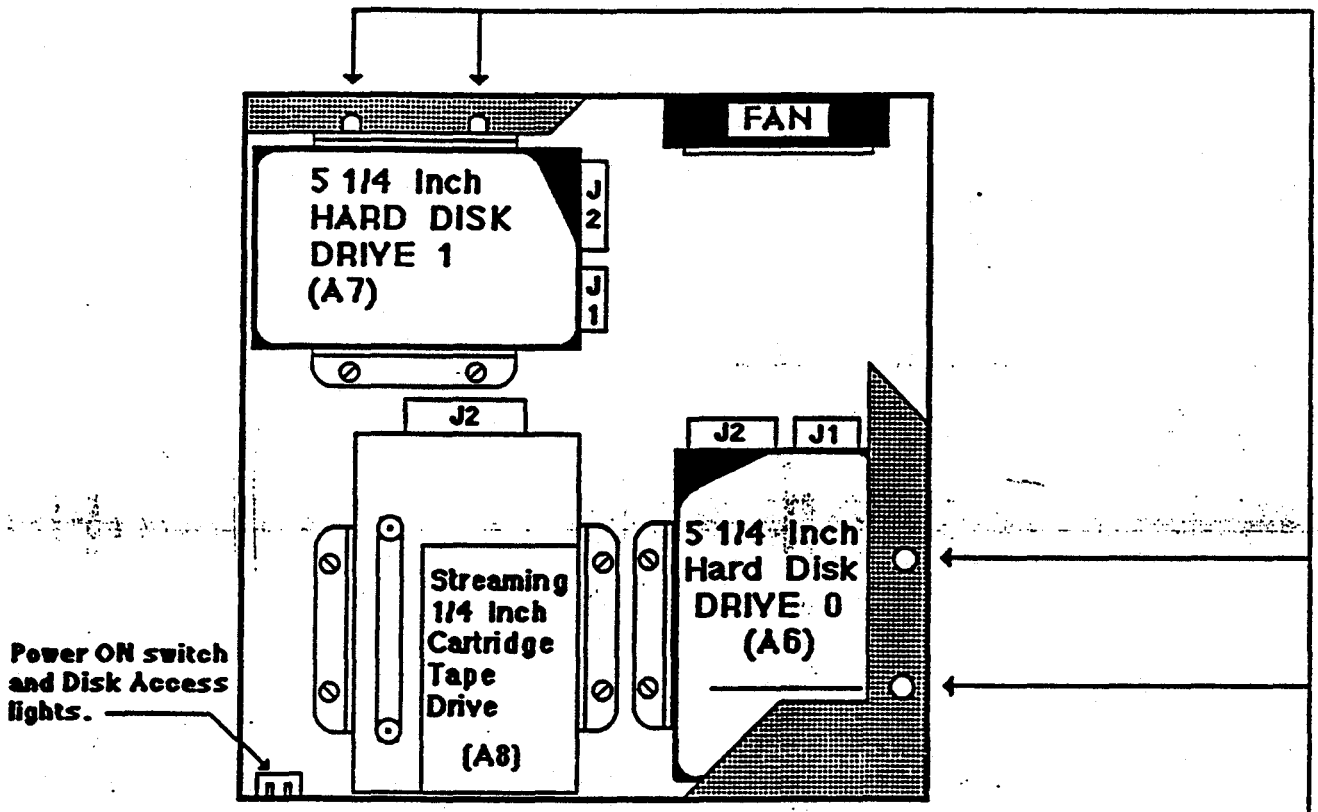


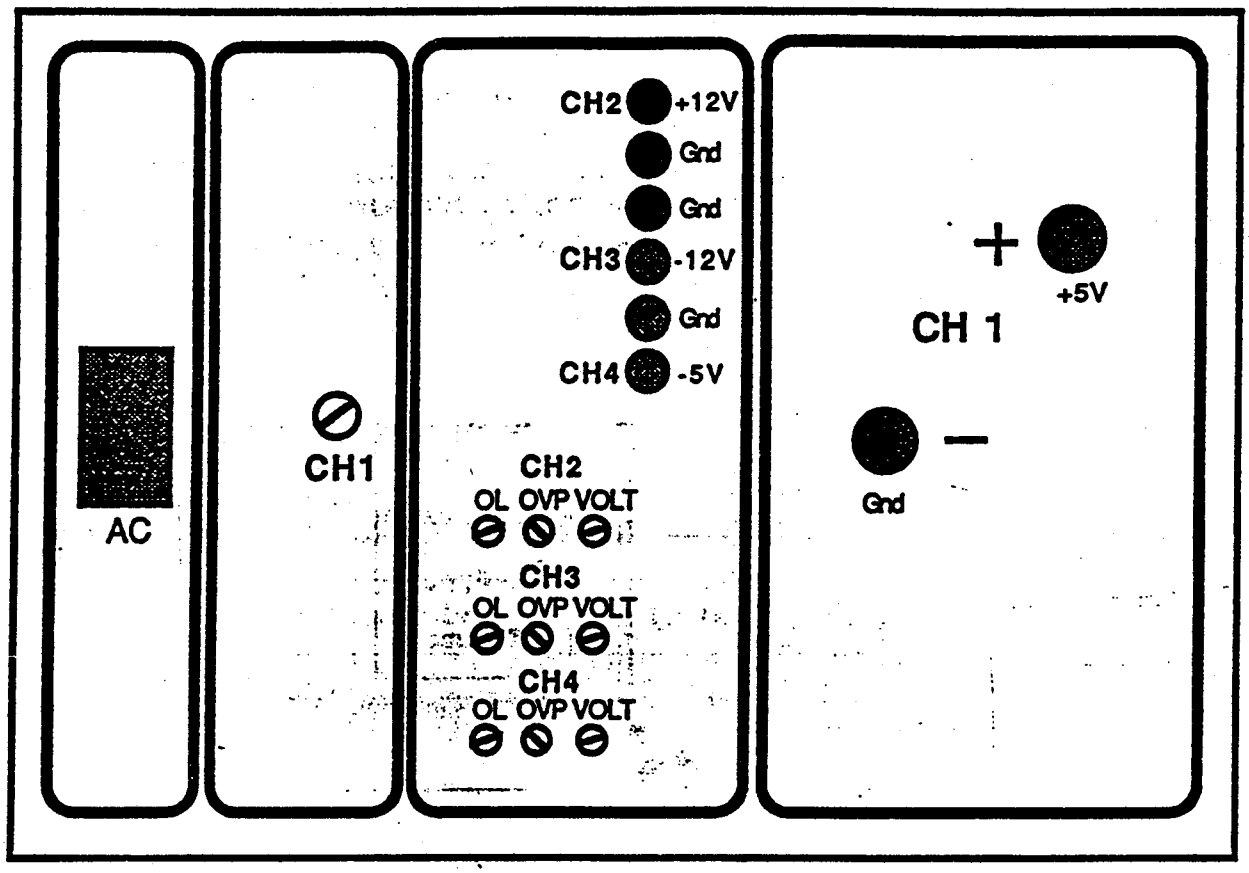
DIAGRAM 13: I/O compartment components location



NOTE:

When removing the Hard Disk Drives, you will need a screw driver with a long neck that will allow you access to the two screws that are obstructed by the flange that is part of the sheet metal that comprises the workstation housing. Part of the flange showing access holes for the disk fastening screws is shown and highlighted by the drawing.

DIAGRAM 14: Power compartment components location



notes:

1. OL (OverLimit protection) : output current of channel
2. OVP (Over Voltage Protection) : input voltage from AC

SECTION 3: Test Items

1. Where are the RGB video and sync lines terminated?

2. If your workstation contains 8 bit planes, show the state of the address switches for the bit plane in card slot 13.

3. What is the recommended state of all the switches in switch packs S1 and S2 located on the IP2 board?

4. What slot will the Ethernet board occupy for a 12 MB system with a Floating Point board?

5. Where does the cable from J1 of the Ethernet board connect to?

6. J3 of the UC4 board connects to what?

7. Where could you monitor the -12 voltage?

Contents

IRIX1 1-1

Introduction 1
Objectives 1
Sample Test Item 1
Resources 1
SECTION 1: Configuration 3
Boot Process/Exercise: General Information 3
Boot Process/Exercise: Micro Code Configuration 4
Boot Process/Exercise: Configuration Switches 5
Boot Process/Exercise: Kernel Boot Process 6
Boot Process/Exercise: Auto Configuration 7
Boot Process/Exercise: Init Process 9
Boot Process/Exercise: IRIS Init - Single User Mode 10
Boot Process/Exercise: IRIS Run Levels 11
Boot Process/Exercise: Multi Command 12
Boot Process/Exercise: Going to Multi User 13
Boot Process/Exercise: File rc.s0 14
Boot Process/Exercise: Fsck 20
Boot Process/Exercise: Mounting Filesystems 23
Boot Process/Exercise: Getty Processes 26
Boot Process/Exercise: Auto Boot to Multi Mode 27
SECTION 2: Boot Process Summary 28
Troubleshooting Hints 30
Shell Command Differences 31
Device File Differences 31
IRIS 3000-Series Disk Partitions 34
SECTION 3: Test Items 35

Introduction

You will learn, in this module, about the differences -some subtle and some not so subtle- in the IRIX implementation for the 3000-Series workstations. Focus will be on the configuration files as well as differences in the area of shell commands.

Objectives

At the completion of this lesson, you will be able to:

- Recognize the shell command differences differences between IRIX version 3.6 and IRIX System 5.3 version 3.1.
- Recognize system configuration file differences.
- Given a system environmental requirement, correctly set it to conform to that configuration.

Sample Test Item

Which configuration file reads the /etc/model entry to determine system type during the boot process?

Resources

This guide

The module contains three sections:

- **SECTION 1:**

  This section describes the configuration files utilized on the 3000-series workstations and summarizes the shell command variations that you will likely encounter. The "boot" sequence will be revisited to illustrate the system configuration process.

- **SECTION 2:**

  This section summarizes the process described in SECTION 1.

- **SECTION 3:**

  This section contains review questions that you should complete after studying this module's material.

SECTION 1: Configuration

Boot Process/Exercise: General Information

This portion of the lesson will explain by illustration the IRIS workstation **Boot Process.**

The boot process occurs in a sequential manner which this document follows.

Several notes must be made:

- The demonstration was produced on a 3030 or 3130 workstation using Revision 3.6 software.

- This discussion is generic and meant to help you understand what occurs during the boot process and look at the process as a **first line** diagnostic to be used to help isolate system failures.

  The boot process supplies vital diagnostic information to the person who understands what occurs during the process.

From power-on reset to **multi-user mode, several software** processes initialize the workstation for use:

- Microcode
- Kernel Boot Process and Auto-configuration.
- Init process

Boot Process/Exercise: Micro Code Configuration

- Power-on / Reset / Reboot

  These three functions cause the Prom Monitor (microcode) to perform a rudimentary system configuration.

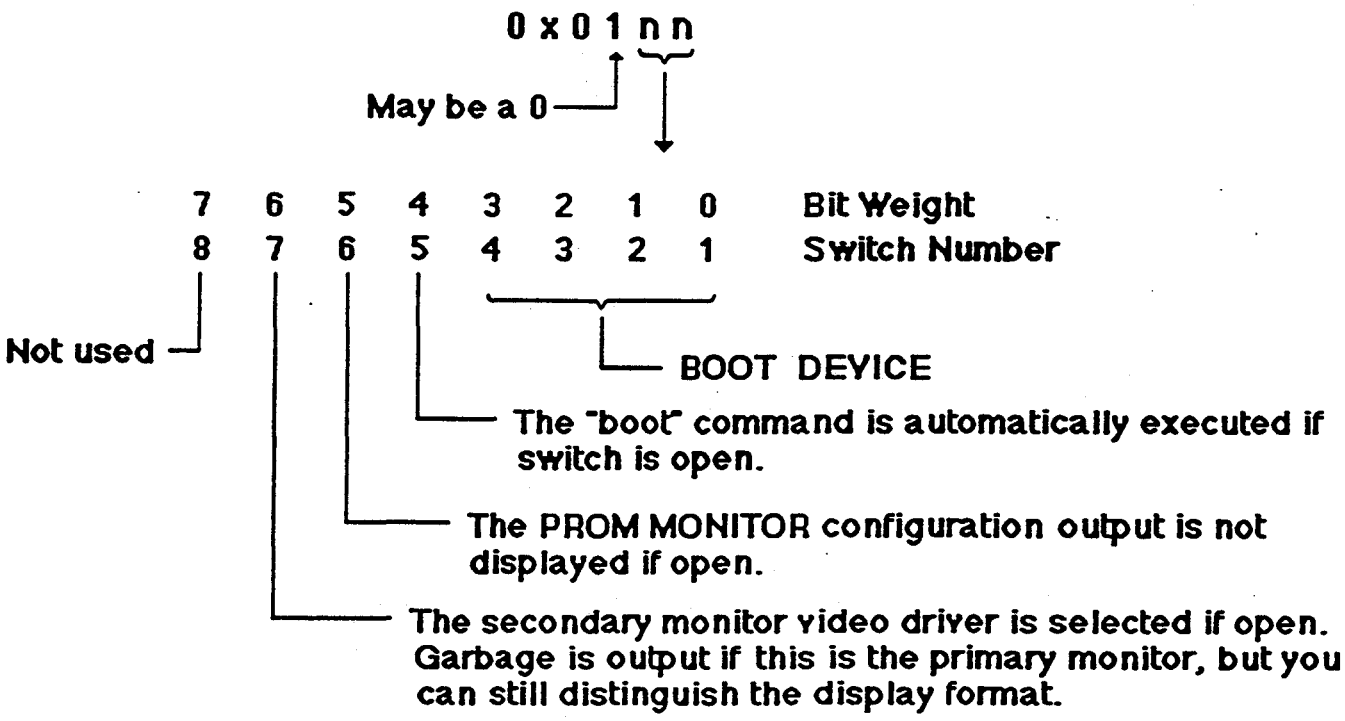
- **You should now execute the reboot** command

```
IRIS (IP2 - Revision B) Monitor Version 3.0.7 December 20, 1985
Memory Size  4mb (Physical Map (1mb/bit) 0x0000000f)
Configuration Switch: 0x0000
  Multibus Window (2mb) at Megabytes 0 and 1.
  Multibus accessible memory (1mb) begins
    at Physical memory page 300,
    at Virtual address 2000000.
```

- Microcode Diagnostics
  - The microcode reads the revision register of the IP2 board.
  - The microcode reports the amount of memory that it can access. The mask **(0x0000000f)** is bit significant per meg.
  - The microcode reports the settings of the rear panel switches.
  - The remainder of the information is determined by jumpers installed on IP2 at manufacture time.
  - If the PROM monitor **cannot** communicate with the primary console, it looks for an ASCII terminal on port 2.

Boot Process/Exercise: Configuration Switches

- See page 3-9 of the IRIS 3000 Owner's Guide.
- The microcode reports configuration switch settings per the following table (closed switches are reported as zeros).



Boot Process/Exercise: Kernel Boot Process

- When the PROM monitor boot command is executed, the microcode loads the Kernel into memory and then transfers program control to the Kernel by jumping to a specified location from which Kernel program execution begins.

```
SGI Extent Filesystem
Loading: si:0:defaultboot
  Text: 038318 bytes
  Data: 0113d8 bytes
  Bss:   024d7c bytes
Jumping to load program @ 20000400
```

- After the Kernel is loaded, the **Auto-configuration** process determines what devices are available and on-line.

  This process should be viewed as a diagnostic tool.

- **You should now execute the boot command b.**

  Configuration output is shown on the next page.

Boot Process/Exercise: Auto Configuration

```
SYSTEM 5 UNIX #0 [Wed May 7 04:49:59 PDT 1986]
(C) Copyright 1986 - Silicon Graphics Inc.
real = 4194304
kmem = 561152
user = 3633152
bufs = 819200 (max=16k)
dsd0 not installed
qic0 not installed
sii0 at mbio 0x07200 ipl 5
si0 (Hitachi 512-17 name: Hitachi 512-17) slave 0
si1 not installed
sf0 floppy (80/2/8) slave 2
siq0 at mbio 0x73fc ipl 5
sq0 (qic02 cartridge tape) slave 0
iph0 not installed
tmt0 not installed
ik0 not installed
nx0 (FW 2.5 HW 4.0) (0800.1400.3948) at mbio 0x7ffc ipl 2
fpa installed
lpen not installed
kernel debugger disabled.
root on si0a
swap on si0b. swplo=0 nswap=64000

          INIT: SINGLE USER MODE

#
```

- The auto-configuration process performs rudimentary diagnostics on all devices that it can communicate with.
  - If communication with the primary terminal is lost during the process, error messages may be output to the ASCII terminal on port 2.

- As a last step of the Kernel Boot process, a process (program) called **Init** is invoked.

Boot Process/Exercise: Init Process

- UNIX systems always run in one state or another. The actions that cause the various states to exist are under control of the **Init** process, which is the **first general process** created by the system at boot time. It reads the file **/etc/inittab**; which defines exactly which processes exist for which run level.

  NOTE:
  Run level does not specify a process priority as the name may imply.

- See **INIT(1M)** for generic description of *init.*

- See **INITTAB(4)** for generic description of *inittab.*

The following page describes how **init** is used on the IRIS.

Boot Process/Exercise: IRIS Init - Single User Mode

- Init scans the file **/etc/inittab** for the statement **initdefault** when it is initially invoked.

  If **initdefault** exists, the specified run level is set as the initial run level.

- The user is informed that the system is in single user mode.

INITTAB

```
is:s:initdefault:#default state
s0::sysinit:/etc/rc.s0 1>/dev/console 2>&1#fix console
bc::bootwait:/etc/brc </dev/console >/dev/console 2>&1#bootrun cmds
bl::bootwait:/etc/bcheckrc </dev/console >/dev/console 2>&1#bootlog
rc::bootwait:/etc/rc 1>/dev/console 2>&1#run com
pf::powerfail:/etc/powerfail 1>/dev/console 2>&1#power fail
co::respawn:/etc/getty console co_9600 none LDISC1#console
d1:23:respawn:/etc/getty ttyd1 dx_9600 none LDISC1#serial 1
d2:x3:respawn:/etc/getty ttym2 uu_1200 none LDISC0#serial 2
d3:x:respawn:/etc/getty ttyd3 dx_1200 none LDISC1#serial 3
c0:x:respawn:/etc/getty ttyd4 dx_9600 none LDISC1# serial option port 4
c1:x:respawn:/etc/getty ttyd5 dx_9600 none LDISC1# serial option port 5
c2:x:respawn:/etc/getty ttyd6 dx_9600 none LDISC1# serial option port 6
c3:x:respawn:/etc/getty ttyd7 dx_9600 none LDISC1# serial option port 7
c4:x:respawn:/etc/getty ttyd8 dx_9600 none LDISC1# serial option port 8
c5:x:respawn:/etc/getty ttyd9 dx_9600 none LDISC1# serial option port 9
c6:x:respawn:/etc/getty ttyd10 dx_9600 none LDISC1# serial option port 10
c7:x:respawn:/etc/getty ttyd11 dx_9600 none LDISC1# serial option port 11
```

Boot Process/Exercise: IRIS Run Levels

- The second field of each entry in **inittab (rstate)** defines the run level.
- IRIS 3000-Series workstations use only three run levels:
  - *s* = Single user mode.
  - *2* = Run level when user is running **XNS.**
  - *3* = Run level when user is using **TCP/IP.**
- The run level is changed using the **telinit** command.
  - **Telinit** is linked to *init.*
  - When **telinit** is executed, *init* again accesses *inittab* and executes all processes that match the specified run level (i.e. telinit 2).

    Processes with **rstate** fields that are empty, are always run.

Boot Process/Exercise: Multi Command

- IRIS default run level is **Single User.**
- The command **multi** will place the system into *multi user.*
- **You should now execute more /etc/multi.**

/etc/multi

```
#! /bin/sh
# bring system up from single-user mode to multi-user mode

#
# Use run-level 3 for tcp systems, and run-level 2 for other systems.
#
if test -x /etc/havetcp && /etc/havetcp; then

          telinit 3
else
          telinit 2
fi
# Prevent the shell from displaying a prompt.
sleep 5
```

- Multi contains two commands: *telinit 2* and *telinit 3.*
  - **Telinit 3** is executed if you are using *TCP/IP* Kernel.
  - **Telinit 2** is executed if you are using the *XNS* Kernel.
- Executing *telinit 2* is the same as executing *multi.* **You should now execute telinit 2.**

Boot Process/Exercise: Going to Multi User

- When **telinit** is executed, *init* re-reads */etc/inittab* and executes the command files **rc.s0, brc, bcheckrc** and **rc.**

  These files all have **rstate** fields that are empty.

INITTAB

```
is:s:initdefault:#default state
s0::sysinit:/etc/rc.s0 1>/dev/console 2>&1#fix console
bc::bootwait:/etc/brc </dev/console >/dev/console 2>&1#bootrun cmds
bl::bootwait:/etc/bcheckrc </dev/console >/dev/console 2>&1#bootlog
rc::bootwait:/etc/rc 1>/dev/console 2>&1#run com
pf::powerfail:/etc/powerfail 1>/dev/console 2>&1#power fail
co::respawn:/etc/getty console co_9600 none LDISC1#console
d1:23:respawn:/etc/getty ttyd1 dx_9600 none LDISC1#serial 1
d2:x3:respawn:/etc/getty ttym2 uu_1200 none LDISC0#serial 2
d3:x:respawn:/etc/getty ttyd3 dx_1200 none LDISC1#serial 3
c0:x:respawn:/etc/getty ttyd4 dx_9600 none LDISC1# serial option port 4
c1:x:respawn:/etc/getty ttyd5 dx_9600 none LDISC1# serial option port 5
c2:x:respawn:/etc/getty ttyd6 dx_9600 none LDISC1# serial option port 6
c3:x:respawn:/etc/getty ttyd7 dx_9600 none LDISC1# serial option port 7
c4:x:respawn:/etc/getty ttyd8 dx_9600 none LDISC1# serial option port 8
c5:x:respawn:/etc/getty ttyd9 dx_9600 none LDISC1# serial option port 9
c6:x:respawn:/etc/getty ttyd10 dx_9600 none LDISC1# serial option port 10
c7:x:respawn:/etc/getty ttyd11 dx_9600 none LDISC1# serial option port 11
```

Boot Process/Exercise: File rc.s0

- You should now login as "root" and change directories to /etc.
- You should now execute `more rc.s0`.

rc.s0

```
#! /bin/sh

# /etc/rc.s0
#   This script is run only during the read of inittab at boot time.
#   This script resets the system console to the actual physical
#   console as defined by /dev/systty

rm -f /dev/syscon
ln /dev/systty /dev/syscon

#
# Initialize the mount table /etc/mtab to reflect the state of
# mounted file systems in single user mode, i.e., that only
# the root file system is mounted.  This allows
# the /etc/mtab file to remain consistent with no further
# handling in the rc and brc scripts.  Manual mounts can be
# done in single user mode and /etc/mtab will remain consistent
# if you subsequently go multi.
#

/etc/devnm / | /etc/setmnt

#
# Display message required for legal reasons.
#

echo " "
echo " "
echo "                        RESTRICTED RIGHTS LEGEND"
echo " "
echo "               Use, duplication or disclosure by the"
echo "               Government is subject to restrictions"
echo "               as set forth in subdivision (c)(1)(ii) of"
echo "               the Rights in Technical Data and Computer"
```

```
echo "                Software clause at 52.227-7013"
echo " "
echo "                        SILICON GRAPHICS, INC."
echo "                          2011 Stierlin Road"
echo "                       Mountain View, CA 94039-7311"
echo " "
```

- *rc.s0* links */dev/systty* and */dev/syscon*. The *rstate* field *sysinit* (in inittab) insures that this occurs first.
  - /dev/systty: UNIX virtual console.
  - /dev/syscon: UNIX physical console.
  - /dev/console: IRIS physical console.
- On IRIS, all three console types are linked together.
- You should execute the following commands:
  - cat /etc/motd > /dev/console
  - cat /etc/motd > /dev/syscon
  - cat /etc/motd > /dev/systty

- *Init* reads the file */etc/brc*, which does the following:
  - Removes the file */etc/mtab* (mount table).
  - Checks for the system model number (/etc/model) and if it does not exist, prompts the user for the model number.

- Since the *rstate* field in *inittab* for this command specifies *bootwait*, this process will be started and the system will wait for it to terminate. When it dies, it will not be restarted.

- **You should now execute more brc.**

brc

```
#! /bin/sh
#
# This command file's function is to set up model-specific
# configuration miscellany.
#

trap "" 2
MODEL=/etc/model
FSTAB=/etc/fstab

if test ! -s $MODEL; then
                # Prompt user for model number and get the reply
                while :
                do
                echo "0etc/model does not exist."
                echo "Please enter the model number of your machine: \c"
                read reply
                case "$reply" in

                2300|2300T )
                model=$reply
                root=md0a
                usr=
                break
                ;;

                1400|2400|2400T )
```

```
model=$reply
root=md0a
usr=md0c
break
;;

1500|2500|2500T )
model=$reply
root=ip0a
usr=ip0c
break
;;

3010|3110 )
model=$reply
root=md0a
usr=
break
;;

3020|3115 )
model=$reply
root=md0a
usr=md0c
break
;;

3030|3120B|3130 )
model=$reply
root=si0a
usr=si0f
break
;;

* )
echo "Invalid model number. Please try again."
echo "Currently the only valid model numbers are:"
echo "1400, 15000
echo "2300, 2400, 2500, 2300T, 2400T, 2500T0
echo "3010, 3020, 30300
echo "3110, 3115, 3120B, 31300
continue
;;
esac
```

```
                done

                # If $FSTAB exists, leave it alone, otherwise build default list.
                if test ! -f $FSTAB; then
                (echo "$root /";
                 if test -n "$usr" ; then echo "$usr /usr"; fi) |
                setmnt -f $FSTAB
                chmod 644 $FSTAB
                fi
                # Now create $MODEL
                echo $model > $MODEL
                chmod 644 $MODEL
                #
                # Depending upon the model, we make generic device links.
                #
                case "$model" in

                1400|2300|2300T|2400|2400T|2500|2500T|3010|3020|3110|3115|3120B )
                rm -f /dev/floppy /dev/rfloppy
                ln /dev/mf0a /dev/floppy
                ln /dev/rmf0a /dev/rfloppy
                ;;

                3030|3130 )
                rm -f /dev/rmt1 /dev/rmt2
                ln /dev/sq0 /dev/rmt1
                ln /dev/nrsq0 /dev/rmt2
                rm -f /dev/floppy /dev/rfloppy
                ln /dev/sf0a /dev/floppy
                ln /dev/rsf0a /dev/rfloppy
                ;;
                esac
fi

#
# Depending on the model we decide if the sky floating point board
# should be downline loaded
#
model=`uname -t`
case "$model" in

1400 | 1500 | 2300 | 2400 | 2500 )
                /etc/fload
                ;;
```

```
esac
```

Boot Process/Exercise: Fsck

- *Init* reads the file */etc/bcheckrc*, which does the following:
  - Checks the date and asks the user if it is incorrect.
  - Runs Fsck.

  Fsck will access the file */etc/fstab* for the filesystems it must check.

- **You should now execute the following commands:**
  - **more bcheckrc**

bcheckrc

```
#! /bin/sh
#                @(#)bcheckrc.sh1.3
# ***** This file has those commands necessary to check the file
# system, date, and anything else that should be done before mounting
# the file systems.
trap "" 2
TZ=`cat /etc/TZ`
export TZ

trap 'echo ; echo Warning: verifying the date - interrupted! ; echo ; break' 2

if [ -x /etc/rc.getdate ] && date `/etc/rc.getdate`
then
                :
else
                while :
                do
                echo "0s the date `date` correct? (y or n) \c"
                read reply
                if [ "$reply" = y ]
                then
                break
                else
                echo "Enter the correct date (mmddhhmm[yy][.ss]):  \c"
                read reply
                date "$reply" > /dev/null
```

```
                fi
                done
fi
# **** Auto check, if necessary

: skipit

# For the 2300, 2300T and 3010 we always autocheck and throw away the output

case `/bin/uname -t` in

                2300 | 2300T | 3010)
                trap "" 2
                /etc/fsck -D -y > /dev/null 2>&1
                exit 0
                ;;
                *)
                ;;
esac

trap 'echo ; echo Warning: filesystem consistency checking - interrupted! ; echo ; e

while :
do
                echo "Do you want to check filesystem consistency? (y or n) \c"
                read reply
                case "$reply" in
                y )
                break
                ;;

                n* )
                exit 0
                ;;

                * )
                echo "Invalid input. Try again."
                continue
                ;;
                esac
done

                /etc/fsck -D
                trap "" 2
```

- **more fstab**

```
/dev/si0a       /               efs rw,raw=/dev/rsi0a  0 0
/dev/si0f       /usr            efs rw,raw=/dev/rsi0f  0 0
#/dev/si1g      /usr2           efs rw,raw=/dev/rsi1g  0 0
```

Boot Process/Exercise: Mounting Filesystems

- *Init* reads the file */etc/rc*, which does the following:
  - Creates a new mount table file */etc/mtab* using the entries in the file */etc/fstab*.
  - Mounts the filesystems per the file */etc/mtab*.
  - Starts the daemons: update, cron, xnsd, lpd and lpsched.
  - Starts any other processes that are specified in */etc/rc* (i.e. initialize a printer port).

- **You should now execute the following commands:**
  - **more rc**

```
#! /bin/sh
#
# System startup commands.  Do not edit this file, instead put local mods
# into /etc/rc.local
#
# $Source: /clone/sgi/usr/src/etc/RCS/rc,v $
# $Revision: 1.31 $
# $Date: 87/09/16 16:42:25 $
#

TZ=`cat /etc/TZ`
export TZ

# Initialization for terminals
case `/bin/uname -t` in
2300|2300T|3010)
                PATH=':/usr/bin:/bin:/etc'
                export PATH
                rm -rf /tmp/* /tmp/.??*;echo "Cleared /tmp"
                if test ! -d /tmp; then
                rm -rf /tmp; mkdir /tmp; chmod a+w /tmp
                fi
                if test -f /.mexrc; then
                su iris -c 'SHELL=/bin/tesh HOME=/ TERM=wsiris mex'
                fi
```

```
                if test -r /etc/sys_id; then
                hostname `cat /etc/sys_id`
                echo Hostname: `hostname`
                else
                echo No hostname
                fi
                echo "Standard daemons:\c"
                if test -f /etc/rc.tcp; then sh /etc/rc.tcp; fi
                if test -f /etc/rc.xns; then sh /etc/rc.xns; fi
                if test -f /etc/rc.488; then sh /etc/rc.488; fi
                sleep 3
                clear
                exit
                ;;
esac

# Initialization for workstations
PATH=':/usr/local/bin:/usr/bin:/bin:/etc'
export PATH
set `who -r`
if [ $7 = 2 -o $7 = 3 ]
then
                /etc/mount -avt efs
                /etc/mount -avt bell

                echo "Resetting locks and logs"
                rm -f /usr/spool/lpd/lock /usr/spool/lp/SCHEDLOCK
                rm -f /usr/adm/acct/nite/lock*
                (cd /usr/spool/uucp; rm -f LCK*; rm -f STST*; rm -f TM*);
                rm -f /usr/mail/*.lock

                hostname `cat /etc/sys_id`
                echo Hostname: `hostname`

# Start standard daemons
                echo "Standard daemons:\c"
                rm -f /dev/log /etc/syslog.pid
                if test -x /usr/etc/syslogd; then
                date '+/etc/rc starting syslogd at %D, %T' >> /usr/adm/SYSLOG
                /usr/etc/syslogd;echo " syslogd\c"
                fi
#  Uncomment the following 3 lines if you want to use Berkeley's lpd
#               if test -x /usr/lib/lpd ; then
#               /usr/lib/lpd;echo " lpd\c"
```

```
#               fi

                if test -x /usr/lib/lpsched; then
                /usr/lib/lpsched;echo " lpsched\c"
                fi
                echo "."

# Start special daemons
                if test -f /etc/rc.tcp; then sh /etc/rc.tcp; fi
                if test -f /etc/rc.xns; then sh /etc/rc.xns; fi
                if test -f /etc/rc.nfs; then sh /etc/rc.nfs; fi
                if test -f /etc/rc.488; then sh /etc/rc.488; fi
                if test -f /etc/rc.mail; then sh /etc/rc.mail; fi

                echo "More standard daemons:\c"
                date '+/etc/rc starting cron at %D, %T' >> /usr/adm/cronlog
                /etc/cron;echo " cron\c"
                /etc/update;echo " update\c"
                echo "."

# Run expreserve after starting sendmail, so that mail messages
# can be sent to the owners of vi work files
                /usr/lib/ex3.7preserve -;echo "Preserved editor files"
                rm -rf /tmp/* /tmp/.??*;echo "Cleared /tmp"
                if test ! -d /tmp; then
                rm -rf /tmp; mkdir /tmp; chmod a+w /tmp
                fi

                if test -f /etc/rc.local; then sh /etc/rc.local; fi
fi

chmod +rw /dev/ttyw*
```

- **more mtab**

```
/dev/si0a / efs rw,raw=/dev/rsi0a 0 0
/dev/si0f /usr efs rw,raw=/dev/rsi0f 0 0
```

Boot Process/Exercise: Getty Processes

- The last step *init* performs, is the starting of the *getty* processes against any attached ASCII terminals on ports 2-4.

Boot Process/Exercise: Auto Boot to Multi Mode

- As a last exercise, you will modify your workstation to boot into *multi user* from a *power-on/reset* condition.
  - Edit */etc/inittab*, and replace the s with a *2* in the second field of the first line.
  - Set rear panel configuration switch 5 to the *open* position.
  - Execute *reboot* and watch what happens.
- Restore configuration switch 5 to the *closed* position and edit */etc/inittab*, replacing the 2 with the s.

SECTION 2: Boot Process Summary

Sequence of events:

- **Reboot or reset:** The microcode reads IP2 revision level, sizes memory and reads rear panel configuration switches.

  If the microcode cannot communicate with the primary console, control is switched to the ASCII terminal on port 2.

- *If* configuration switch **5** is open, the microcode executes the boot command.

  Else, the PROM monitor is entered, from which the boot command must be executed manually.

- The microcode looks for the file **defaultboot** (or another boot file that was specified with the boot command) on the boot device and loads the boot program (Kernel) into main memory.

- The Kernel then performs the **auto-configuration** process.

  If specific types of errors are detected during the configuration process, or communication is lost with the primary console, error information is sent to the ASCII terminal on port 2 and the process is terminated.

- After the auto-configuration process completes and no errors have been detected, the Kernel invokes the **Init** process.

- **Init** scans the file */etc/inittab* for the statement **initdefault:**

  If **initdefault** is present, *init* places the system into the specified run level. (IRIS default is single user)

  Otherwise, the system prompts the user to enter a run level.

  If the default run level is **2** or **3, init** reads all configuration files in **/etc/inittab** that match the run level and executes all specified processes.

  Otherwise, the boot process halts with the system in **single user** mode. The user

must then execute *multi* to continue the bring up process.

Assuming **initdefault** specified run level 2 or 3, or the user executed **multi**, the process continues:

- Upon issuance of a run level change, **init** re-reads the file **/etc/inittab** and reads the files: *rc.s0, brc, bcheckrc, rc* and any other files that match the run level.

  Commands contained in these configuration files are executed in an order specified by the **action** field of each entry in **inittab**. (i.e. sysinit, bootwait, wait)

  - **rc.s0** links */dev/systty* and */dev/syscon.N*
  - **brc** removes the old */etc/mtab* file and checks for the system model number.

    If the model number is present in the file **/etc/model**, the process continues.

    Otherwise, the user is asked to enter the model number.
  - **bcheckrc** checks for the date and prompts user to answer if the date is correct and if **FSCK** is to be run.

    If **FSCK** is to be run, the file */etc/fstab* is read to determine what filesystems should be checked.
  - **rc** creates a new mount table file */etc/mtab*, mounts the specified filesystems, removes locks and logs, starts daemons and anything else the user specified in the **rc** file.

- **Init** starts a *getty* process for each ASCII terminal specified in the file **/etc/inittab**.

- The **login** prompt is displayed on all on-line terminals and the system is in **multi-user** mode.

Troubleshooting Hints

If your system does not boot and/or enter multi-user mode in a normal way, or your customer is complaining various terminals were not configured, or filesystems were not mounted, etc., before you start system troubleshooting you should always ask your customer if he/she changed any configuration file just prior to the failures.

i.e.

.login
.cshrc
.profile
bcheckrc
brc
checklist (In Rev. 3.4 and earlier)
crontab
fstab(Replaces rc.fs in Rev. 3.5)
gettydefs
group
inittab
mnttab (In Rev. 3.4 and earlier)
mtab(Replaces mnttab in Rev. 3.5)
model
multi
passwd
rc
rc.fs (In Rev. 3.4 and earlier)
rc.s0
rc.tcp
rc.xns
sys_id
termcap
ttytype

Remember, the key question is: Did you change any configuration files?

Shell Command Differences

Several differences exist between the 3000-Series and 4D-Series IRIX releases that you should be aware of. They have been compiled into the following tables:

3000-Series	4D-Series	Comments
ps	ps	Use option *-aux*
xx	n/a	XNS communications option command
xlogin	n/a	XNS communications option command
xcp	n/a	XNS communications option command
sgilabel	fx/label	Creates a disk drive label
n/a	chkconfig	Feature configuration switches

Device File Differences

Devices are special files located in */dev* that point to kernel resident device drivers. Those device drivers perform all of the I/O between a calling process and whatever physical device is being accessed. The advantage of such an architecture is that devices can be written to and read from just like any file.

All hardware devices supported or attached to the system must have a corresponding device. Normally, the device will be located in */dev* and will be given a name reflecting the device it supports. The following list represents the normal devices provided with the IRIS 3000 and IRIS-4D:

3000-Series	4D-Series	Comments
n/a	dsk/ips0d0s0-7	Controller 0, drive 0, partitions 0-7.
n/a	dsk/ips0d0vh	Controller 0, drive 0, partition 8 (volume header).
n/a	dsk/ips0d0s9	Controller 0, drive 0, partition 9.
n/a	dsk/ips0d0vol	Controller 0, drive 0, partition a (entire disk).
n/a	dsk/ips0d0s11-15	Controller 0, drive 0, partitions b-f.
n/a	dsk/ips0d1s0-7	Controller 0, drive 1, partitions 0-7.
n/a	dsk/ips0d1vh	Controller 0, drive 1, partition 8 (volume header).
n/a	dsk/ips0d1s9	Controller 0, drive 1, partition 9.
n/a	dsk/ips0d1vol	Controller 0, drive 1, partition a (entire disk).
n/a	dsk/ips0d1s11-15	Controller 0, drive 1, partitions b-f.

n/a	dsk/ips1d0s0-7	Controller 1, drive 0, partitions 0-7.
n/a	dsk/ips1d0vh	Controller 1, drive 0, partition 8 (volume header).
n/a	dsk/ips1d0s9	Controller 1, drive 0, partition 9.
n/a	dsk/ips1d0vol	Controller 1, drive 0, partition a (entire disk).
n/a	dsk/ips1d0s11-15	Controller 1, drive 0, partitions b-f.
n/a	dsk/ips1d1s0-7	Controller 1, drive 1, partitions 0-7.
n/a	dsk/ips1d1vh	Controller 1, drive 1, partition 8 (volume header).
n/a	dsk/ips1d1s9	Controller 1, drive 1, partition 9.
n/a	dsk/ips1d1vol	Controller 1, drive 1, partition a (entire disk).
n/a	dsk/ips1d1s11-15	Controller 1, drive 1, partitions b-f.
md0a	root	Root partition (md refers to ST506 controller).
si0a	root	Root partition (si refers to ESDI controller).
ip0a	root	Root partition (ip refers to SMD controller).
rsi0a	rroot	Raw disk root partition.
si0a-h	usr	User filesystem partition.
rsi0a-f	rusr	Raw filesystem user partition.
si0a-f	n/a	Disk partitions for ESDI-type controllers.*
si1a-h	n/a	2nd disk partitions for ESDI-type drives.*
rsi0a-f	n/a	Raw disk partitions for 1st drive.*
rsi1a-f	n/a	Raw disk partitions for 2nd drive.*
md0a-h	n/a	Disk partitions for ST506-type drives.*
md1a-h	n/a	2nd disk partitions for ST506-type drives.*
rmd0a-h	n/a	Raw disk partitions for ST506-type drives.*
rmd1a-h	n/a	2nd raw disk partitions for ST506-type drives.*
ip0a-h	n/a	Disk partitions for SMD-type drives.*
ip1a-h	n/a	2nd disk partitions for SMD-type drives.*
rip0a-h	n/a	Raw disk partitions for SMD-type drives.*
rip1a-h	n/a	2nd raw disk partitions for SMD-type drives.*
n/a	vh	Linked to /dev/dsk/ips0d0vh.
n/a	rvh	Linked to /dev/rdsk/ips0d0vh.
n/a	swap	Linked to /dev/dsk/ips0d0s1.
n/a	rswap	Linked to /dev/rdsk/ips0d0s1.
sq0	mt/ts0d0	1/4" tape with auto rewind.
nrsq0	mt/ts0d0nr	1/4" tape no auto rewind.
rmt1	mt/ts0d0	1/4" tape with auto rewind.
rmt2	mt/ts0d0	1/4" tape with no rewind.
n/a	mt/xmt0d0.800	1/2" tape with auto rewind, density = 800.
rmt3	mt/xmt0d0.1600	1/2" tape with auto rewind, density = 1600.
rmt3	mt/xmt0d0.3200	1/2" tape with auto rewind, density = 3200.

rmt3	mt/xmt0d0.6250	1/2" tape with auto rewind, density = 6250.
rmt4		1/2" no rewind
n/a	tape	Linked to /dev/mt/ts0d0.
n/a	nrtape	Linked to /dev/rmt/ts0d0nr.
n/a	/dev/SA	Special files used by System Administration.
n/a	/dev/rSA	Special files used by System Administration.
n/a	/dev/gro	Used for graphics output.
n/a	/dev/grin	Used for graphics input.
cent	cent	Ikon color printer controller.
console	console	IRIS-4D console.
	dials	Dial/button box.
drum	n/a	Refers to the system paging device.
grconc	grconc	Graphics console.
grcons	grcons	Graphics console.
n/a	keydb	Console keyboard.
mem	mem	Real memory.
kmem	kmem	Virtual memory.
n/a	mouse	Mouse.
null	null	Null file (bit bucket)
n/a	prf	Pseudo-device for system admin.
queue	queue	Queue for graphics input and output.
syscon	syscon	UNIX system console.
systty	systty	UNIX system terminal.
n/a	tablet	Digitizer pad.
tek	tek	Tektronics Color printer.
	ttyd1-12	Hardwired RS-232C ports.
	ttym1-12	Serial ports for modems.
ttyd0-31	n/a	Hardwired RS232C ports.
ttyn0-31	n/a	I/O conduits for use by the Ethernet controller.
ttyq0-31	ttyq1-99	Pseudo tty devices for *TCP rlogin.*
ttyw0-9	n/a	MEX window ports
ttyT0-7	n/a	TELNET/rlogin server devices.
ttyT8	n/a	uucp server device.
vers	vers	Color printer.
vp0	vp0	Ikon color printer controller.
hy0-3	hy0-3	HYPERnet port devices.

IRIS 3000-Series Disk Partitions

SECTION 3: Test Items

1. Which configuration file reads the /etc/model entry to determine system type during the boot process?

   _____

2. If communications with the graphics console are lost, control is switched to the ASCII port #_____.

3. By default, the *initdefault* entry in /etc/inittab is set to which run-level?

   _____

4. If your workstation is a 3020, the root partition is given what designation?

   _____

5. Which configuration file creates a new mount table file */etc/mtab* using the entries from the file */etc/fstab*?

   _____

Contents

BACK1 1-1
Introduction 1
Objectives 1
Criterion Test 1
Resources 1
SECTION 1: Instructional Procedure 3
Making a Bootable Tape 3
Creating a Backup Tape 7
SECTION 2: Procedures Summary 14
Bootable Tape Procedure 14
Backup Tape Procedure 15
SECTION 3: Test Items 17

Introduction

The "bootable" backup tape can be used to boot the workstation and rebuild the disk in case the file system on the disk is damaged beyond use. Make a bootable backup tape as soon as you install a new workstation. Make a new bootable backup tape whenever you install new software or install any new options.

Objectives

At the completion of this module, you will be able to:

- Recognize the function and organization of the "mkboot" tape.
- Create a "mkboot" backup of the "root" filesystem.
- Create "root" and "usr" filesystem backups using "cpio".

Criterion Test

Using the procedure in this module, correctly make "bootable" and "backup" tapes.

Resources

This guide

Three blank cartridge tapes

This lab project contains three sections:

Section 1: This section is used to teach you the procedure and explain to you what you are doing.

Section 2: This section lists just the procedure without all the instructional text.

Section 3: This is a number of review questions that you will complete after performing the lab project.

SECTION 1: Instructional Procedure

Making a Bootable Tape

STEP 1: Power on the workstation.

Power on your IRIS workstation using the front panel switch. You should see the initial PROM monitor message appear at your terminal:

```
IRIS (IP2 - Revision B) Monitor Version 3.0.7 December 20,
1985
Memory Size  4mb (Physical Map (1mb/bit) 0x0000000f)
Configuration Switch: 0x0000
  Multibus Window (2mb) at Megabytes 0 and 1.
  Multibus accessible memory (1mb) begins
    at Physical memory page 300,
    at Virtual address 2000000.
iris>
```

STEP 2: Boot the UNIX System.

```
iris > b

SGI Extent Filesystem
Loading: si:0a:defaultboot
  Text: 038318 bytes
  Data: 0113d8 bytes
  Bss:   024d7c bytes
Jumping to load program @ 20000400



SYSTEM 5 IRIX #0 [Wed May 7 04:49:59 PDT 1986]
(C) Copyright 1986 - Silicon Graphics Inc.
real = 4194304
kmem = 561152
user = 3633152
bufs = 819200 (max=16k)
dsd0 not installed
```

```
qic0 not installed
sii0 at mbio 0x07200 ipl 5
si0 (Hitachi 512-17 name: Hitachi 512-17) slave 0
si1 not installed
sf0 floppy (80/2/8) slave 2
siq0 at mbio 0x73fc ipl 5
sq0 (qic02 cartridge tape) slave 0
iph0 not installed
tmt0 not installed
ik0 not installed
ib0 not installed
ib1 not installed
px0 not installed
cd0 not installed
cd1 not installed
cd2 not installed
cd3 not installed
hy0 not installed
ex0 (FW 2.5 HW 4.0) (0800.1400.3948) at mbio 0x7ffc ipl 2
fpa installed
lpen not installed
kernel debugger disabled.
root on si0a
swap on si0b. swplo=0 nswap=64000

INIT: SINGLE USER MODE

#
```

STEP 3: Mount the USR filesystem.

Your next step will mount the user file system.....Remember, when you boot IRIX and the system is placed into single user mode, only the root file system is mounted by the system.

Once again return to the root directory.

 # cd /

Mount the user file system.

 # mount /dev/si0f /usr

The above command mounts the file system */dev/si0f* to the directory /usr.

NOTE:

If you had not returned to the root directory (cd /) before executing the mount command and you were still in the */usr* directory, you would get *mount: Device busy* message on your terminal when you attempted the mount.

Now if you executed the **df** command again, you will see a new line in the output indicating the mount.

```
                # df
Filesystem      Typekbytesusedavail%usedmounted
/dev/si0a       efs89254886389856%/
/dev/si0f       efs39865180862115546%/usr
```

We now have a second line in the table indicating the */usr* mount.

At this time install the blank tape into the workstation tape drive. (Make sure that it is not write protected).

The following table shows the storage capacity of the different tapes the 3000-Series workstations can handle:

Tape Type	Tape Length	Capacity
Quarter-inch	450 feet	~40 Mb
Quarter-inch	600 feet	~60 Mb
Half-inch	2400 feet	~45 Mb

Note the size (in kbytes used) of the user file system. If this value, when added to the size of the root file system (kbytes used), is greater than 35 megabytes when using a *450 foot* tape, then you will need to backup the */usr* filesystem on additional tapes. In this procedure, we will make our /usr filesystem using a separate cartridge tape.

You will now insure that the tape is at load point by executing a rewind command.

```
# mt  rew
```

The next command will create the bootable tape by placing the following on the tape:

File 1 = /stand	cpio format	(The utilities)
File 2 = /root	dd format	

```
#mkboot
```

Make sure the system is idle--no other user activity.

```
 /dev/si0a
  File System: root Volume: SGI

  **Phase 1 - Check Blocks and Sizes
  **Phase 2 - Check Pathnames
  **Phase 3 - Check Connectivity
  **Phase 4 - Check Reference Counts
  **Phase 5 - Check Free List
  403 files  4886 K blocks 3898 K free
mkboot: file 1 = cpio of /stand
mkboot: file 2 = dd of si0a
150+0 records in
150+0 records out

mkboot complete.
```

The process (when writing only */root*) will take about **5 minutes.**

You are now prepared for the worst case failure of your disk system; a filesystem that becomes so corrupted, that *fsck* can not repair it.

Creating a Backup Tape

Before proceeding, label one blank tape as ROOT BACKUP and another as USR BACKUP.

STEP 1: Boot the IRIX System into single-user state.

Make sure that *only* the root filesystem is mounted.

STEP 2: Check Filesystem Consistency

Before you backup a filesystem you should run *fsck* against the filesystem to check for any corrupted files.

```
# fsck

File system: root Volume: SGI

** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
nnn files nnnnn K blocks nnnn K free
```

STEP 3: Back Up the ROOT Filesystem.

Load a blank tape into the cartridge drive.

You are now ready to create the backup tape of the Root file system.

The next command will write the Root File System onto your tape using the **cpio** command. The (1) following the (h) option will cause a tape rewind before the write is started and after the write is complete.

```
# cpio -ovhl  /
```

NOTE:
If you had your tape write protected at this time, the following message would appear:

```
qic0: write protected
cpio: Can't open /dev/rmt1 for writing.
```

The write takes about 5 minutes and when it completes, the (#) prompt is displayed.

Now, remove the tape and install the tape labeled *USR BACKUP*.

STEP 4: Back up the USR filesystem.

Your next step will mount the user file system.....Remember, when you boot unix and the system is placed into *single-user mode*, only the root file system is mounted by the system.

```
# mount  /dev/si0f  /usr
```

The above command mounts the file system */dev/si0f* to the */usr* directory.

You will next change directories to the *usr* directory.

```
# cd  /usr
```

Now look at the size of the *usr* directory. We want to determine if its size (in megabytes) is too large for the tape. Use the **df** command for this task.

```
# df

File system    Type  kbytesuse   avail %usedmounted
/dev/si0a efs  8925  4886  3898  56%   /
/dev/si0f efs  39865 18086 21155 46%   /usr
```

The above output shows the size of the *usr* file system. (Remember, you may have duplicate lines because of a system bug reporting the disk free space). If the size is too large for the tape, then you must split the file system up into smaller parts and save them on additional tapes.

For this project you will only need one tape. Following the " cpio write " you will be shown how to split the file system if it was too large.

Now, write the file system to tape using the cpio command.

```
# cpio  -ovhl  .
```

The (.) following the option argument tells the system to use the current directory, which is, */usr*. (Remember, the [1] following the [h] option instructs that a rewind take place before and after the write operation.) The complete tree below *usr* is saved.

When the write is complete (about 20 minutes on your training workstations) the # prompt is displayed.

This part of the lab project is for general information. It gives examples of various *store-restore* operations.

QUESTION: What is the difference between the BACKUP tapes and the BOOTABLE tape?

ANSWER:
The Bootable tape is used when the file system is so corrupt that Unix cannot be booted. The utilities in the first file on the tape are used to rebuild the disk system. This totally distroyes any existing files on the disk and restores the system to a known good state.

From the point when the bootable tape is created the user should then create the backup tapes for the *root* and *usr* filesystems.

The backup tapes are used when only specific files or directories have been lost or corrupted; IRIX will boot up, the filesystems are consistent, but certain files need restoring. These files could be loaded from the backup tapes and placed into the directories they belong.

The user should create *incremental backups* per some fixed schedule to insure that all filesystems are backed up to a known good point.

QUESTION:
What if you had a backup tape (or any cpio format tape) and wanted to know what was on the tape?

ANSWER:
Mount the tape and execute the following command.

```
cpio  -ihvtl

  or

cpio  -ihvtl  >  list
```

The first command will rewind the tape and then read the files printing a list of the pathnames on the screen for each file on the tape. Using the t option causes only a read of the tape without directing the output to any directories on the disk. A rewind will be performed when the read is complete.

The second command will do the same thing, but the output will be re-directed into a file called *list*.

QUESTION:
You have somehow removed a file from your system. How do you restore the file?

ANSWER:
We will assume that the file you lost was */bin/time*. Mount the Root backup tape and perform the following command. (You may want to try this on the training workstation. Go remove the file and then execute the command)

```
cpio  -ivhdum1  /bin/time
```

The above command will restore *time* to the */bin* directory and print the pathname on the screen for you when it is done. A rewind will be performed after the restore.

If you wanted to restore several files, just list them as part of the argument to the cpio command. If you do not supply a file name, the default for the argument is the WILD CARD *metacharacter*, and the complete tape is restored.

QUESTION:
What if you wanted to move a complete directory from one workstation to another. You could use the following method.

Assume that the name of the directory is */emacs* and that it lives under */usr/lib*. You want to move it to the same relative position in the tree on the other workstation.

ANSWER:
At the workstation that has the directory, execute the following:

```
# cd  /usr/lib
# cpio  -ovh1  emacs  (this dumps complete tree below emacs)
```

At the other workstation, execute the following:

```
# cd  /usr/lib
# cpio  -ivhmud1    (without an argument, restore complete
                     tape)
```

QUESTION:
What if you had some special files that you wanted saved on a scratch tape. How would you do this?

ANSWER:
For illustration, lets say you had three files that you wanted saved on tape. We'll call them: Files A, B and C. Mount your tape and execute the following commands:

```
# cd  <your directory>
# mt  rew          (rewinds the tape)
# cpio  -ohv2  A   (writes file A with no rewind)
# cpio  -ohv2  B   (writes file B with no rewind)
# cpio  -ohv2  C   (writes file C with no rewind)
# mt  rew          (rewinds tape)
```

The *2* after the *5* option (of the cpio commands) indicates that a rewind is not to be performed before or after the operation.

You would now have a tape with three files on it : A, B and C.

QUESTION:
You want to restore only the third file (C) from your tape. How do you do this?

ANSWER:
Execute the following commands:

```
# cd  <your directory>
# mt  rew       (rewind tape)
# mt  fsf  2    (forward space two files)
# cpio  -ivh2   (read the third file, no rewind)
# mt  rew       (rewind tape)
```

NOTE:
If you wanted to know what was on the scratch tape, rewind the tape and execute **cpio -ivht2**. The name of the file will be printed on the screen.

If you issued the command again, then the name of the second file (if it exists) will be printed. This could be continued until you ran out of tape or files.

The operation just described could also be done like this:

```
# mt  fsf  2    (skip forward two files)
# cpio  -ivht2 (read the third file)
```

This completes the lab project. The next section gives only the generic procedure for creating the backup tapes.

When you are finished, complete the questionnaire in section three and turn it in to your instructor.

NOTE:
This lab project only mentioned and used the **cpio** command to create and restore the backup tapes. Another command, *tar*, could also have been used. This command is used only for tape drives, whereas, *cpio* can be used with tapes and disks to move files. The procedure is outlined in the IRIS OWNER'S GUIDE at the same place the above procedures for *cpio* are found.

SECTION 2: Procedures Summary

Bootable Tape Procedure

This section shows only the actual steps required for creating a bootable tape.

The process is outlined in the IRIS Series 3000 Owner's Guide, the section on Workstation System Administration.

1. Reboot the system into single-user mode.

```
reboot
b
fsck
```

2. Correct any errors reported by FSCK.

3. Change your working directory to the root directory.

```
cd  /
```

4. Mount the user file system:

For IRIS 3020:

```
mount  /dev/md0c  /usr
```

For IRIS 3030:

```
mount  /dev/si0f  /usr
```

5. Mount your blank tape.

6. Rewind the tape.

```
mt  rew
```

7. Run the " mkboot " program.

```
mkboot  /usr
```

If /usr is too large, back it up onto another tape and create a bootable tape with the root filesystem only.

```
mkboot
```

NOTE

You must be in *SINGLE-USER MODE* to create the backup tape of the Root file system.

Backup Tape Procedure

This section shows only the actual steps required for creating backup tapes.

The process is outlined in the IRIS OWNER'S GUIDE, section 4: Making Periodic Backups.

1. Reboot the system into single-user mode.

```
reboot
b
fsck
```

2. Correct any errors reported by FSCK.

3. Change your working directory to the root directory.

```
cd /
```

4. Install your blank tape.

5. Backup the Root file system.

```
cpio -ovhl /
```

6. Mount the user file system:

```
mount /dev/si0f /usr
```

7. Install your other blank tape.

8. Change directories to USER directory.

```
cd  /usr
```

9. Backup the USER files system.

```
cpio  -ovhl  .
```

SECTION 3: Test Items

1. Show the command syntax used to backup the *root* filesystem to tape.

2. You want to execute the *cpio* command, but do not want to cause a rewind before or after the operation. What character do you include as part of the options argument to insure this?

3. What command can you use to list the size of files or directories?

4. When would you use the Backup tapes?

5. What command would you use to print a listing of files contained on a tape written in *cpio* format?

6. You want to forward space your tape four files, show the command.

7. You want to place two directories on a single tape. Each directory to be a separate file on the tape. Show the commands to do this.

8. What mode must the workstation be in to perform the *mkboot* procedure?

9. What command would you use to inform any users on the network that you were going to reboot the system?

10. In what manuals could you find information concerning FSCK?

11. What disk partition is the */usr* filesystem stored in?

12. List the formats that the data is written in on the bootable tape for each file:

    File 1 =

    File 2 =

    File 3 =

13. How could you create a bootable tape with only the */stand* directory and the */root* file system on the tape?

14. What command is used to determine disk free space?

15. How can you really tell if a filesystem is mounted?

16. If you are in *multi-user* mode, how do you get to *super-user* mode?

17. Show the command for a *tape rewind* operation.

Contents

FEX10 1-1
Introduction 1
Objectives 1
Resources 1
SECTION 1: Instructional Procedure 3
SECTION 3: Test Items 10

FEX10

Introduction

The *fex* utilities allow you to perform a variety of low-level diagnostic and configuration functions on disk drives much like its' 4D-Series counterpart *fx*. As you will see, it is a little less-sophisticated and flexible, yet does the job adequately.

Objectives

At the completion of this lesson, you will be able to:

- Correctly load the appropriate "fex" program from a mkboot tape.
- Enter the "FE" portion of "fex".

Resources

This guide

Bootable cartridge tape

This module contains two sections:

- **SECTION 1:**

  This section contains execution instructions for *fex* programs.

- **SECTION 2:**

  This section contains test items that you should complete after studying this module's material.

SECTION 1: Instructional Procedure

When your workstation is experiencing disk drive filesystem errors and FSCK fails to correct the error, or you cannot boot UNIX due to disk errors, then its time to format the disk and restore the complete file system.

The following text will lead you through the execution of the commands necessary to *format the disk* and *restore the disk filesystems.*

STEP 1: Reboot the workstation.

```
# reboot

IRIS (IP2 - Revision B) Monitor Version 3.0.7 December 20,
1985
Memory Size  4mb (Physical Map (1mb/bit) 0x0000000f)
Configuration Switch: 0x0000
  Multibus Window (2mb) at Megabytes 0 and 1.
  Multibus accessible memory (1mb) begins
    at Physical memory page 300,
    at Virtual address 2000000.
iris>
```

STEP 2: Boot sifex from tape.

At this time install the Bootable tape created during Module Back1.

FEX is a Formatter and EXerciser (thus its name) for winchester disk drives on IRIS workstations. It will handle any one of the several types of disk drives that an IRIS may be using. Three *fex* programs exist; with the type of controller installed the determining factor as to what FEX program you will use.

- **MD FEX:**
  Used with small winchester drives. i.e Vertex, Priam, and Maxtor.
- **IP FEX:**
  Used when your disk drive is the 474 MB Fusitsu disk drive.

- **SI FEX:**
  Used when your disk drive is the 170 MB or 380 MB disk drives.

This exercise is written assuming the workstation contains a 170 MB disk. Therefore, this project will illustrate the use of SIFEX.

The method you use for SIFEX is the same for the other two programs. They are all *menu driven* and if you have a good understanding of SIFEX you will not experience any problems running the IPFEX and MDFEX programs.

Before we actually begin the process, let me give you some general information about the *FEX* programs; command syntax, responding to prompts, and special keyboard control keys.

- Most of your input to *FEX* is performed using a single character *without* using the *return key*. When you are entering a command, *FEX* will wait for you to enter a single character and then immediately execute the command. The exception to this is when *FEX* is requesting a "string" of characters or numbers from you.
- Strings may be edited using **backspace** to delete the previous character, and **ctrl- u** to delete all characters back to the last prompt.
- When *FEX* prompts for a value or a response, the default value or response is shown in parenthesis ( ).
- Using the **DEL** key will return you to the top level command prompt (`sifex 3.6>`) from anywhere in FEX.

You must now boot the desired fex program from your bootable tape. Remember, the first file on the bootable tape contains the stand-alone fex programs (some people call these utilities). Before you can boot the program you have to know what prom monitor "boot command" to use.

We will ask the PROM monitor to display the available commands for this prom monitor. We do this using the help command.

```
                iris > h

General Monitor Commands (All numeric values in hex):
  MEDIA is one of the following:
                hd- hard disk.(look for ip, sd, then md)
                ct- cartridge tape. (look for st, then mt)
                fd- floppy disk. (look for sf, then mf)
                ip- interphase disk. (474 MB drive)
                sd(or si)- storager disk.  (170 MB drive)
                md- midas disk.  (72 MB drive)
                st(or sq)- storager cartridge tape.  (1/2" tape)
                mt(or mq)- midas cartridge tape.  (1/4" tape)
                sf- storager floppy disk.
                mf- midas floppy disk.
                xns- network.  (ethernet)
                rom- EPROM board.
DEVSPEC is one of the following:
                host name- Name of the host. (MEDIA must be xns)
                unit- unit number of device (0, 1, ...).
                   (MEDIA must be a tape or disk device)
                <unit><fs>- unit number and filesystem (a-h)
                   (MEDIA must be a disk device.)
                address- multibus address.
                   (MEDIA must be a EPROM board.)

[MEDIA.DEVSPEC:][file]  load and begin execution of the named file
                                  file defaults to defaultboot
                                  SPECs are from switch settings
```

b [MEDIA.DEVSPEC:][file] same as above

n [DEVICE:][file] same a b with MEDIA = xns

ls [MEDIA.DEVSPEC:][file] list the files on the device

l [MEDIA.DEVSPEC:][file] load but don't begin execution of the file

g address [stack] start executing at specified address.
the stack address is a option.

Continue (y or n)?: **n**

We can see that the media mnemonic used to boot from tape is ct.

Before you boot *fex*, why not perform a short review of some operations covered about in module BOOT1.

First, list the contents of the *root* partition.

```
iris> ls  si0a:

SGI Extent Filesystem
 .              ..             lost+found     usr
 .cshrc         .login         .profile       Versions
 etc            kernels        lib            stand
 tmp            vmunix         vmunix1        ovmunix
```

This is a very basic command and if you could not read the disk you have serious problems with your disk subsystem. If you have a second disk, then you should try to list the contents of one of it's partitions, and if it fails start thinking disk controller problems.

To see what is contained on the first file on your Bootable tape, type:

```
iris > ls  ct0:

mdfex
ipfex
sifex
```

OK, these are the programs that were placed here by the **mkboot** program, so I guess we can now boot the one we want, sifex.

```
iris> b  ct0:sifex

SIFEX for (ESDI/ST412/506) Disk Drives and QIC-02 Tape
Interphase Storager Disk/Tape Controller Model 3030
** Version: 3.6
```

```
sifex 3.6>
```

STEP 3: Entering Field Engineer portion of sifex.

Two command sets exist for *sifex*; one for the user and one for the Field Engineer.

- **User Command Set**
  This is a limited set of commands that will allow the user access to *sifex* for the purpose of restoring disk files from tape.
- **Field Engineer Command Set**
  This is the complete set of commands that allow disk formatting, disk drive testing, and tape-to-disk copying.

In order to gain access to the full set of commands, one of eight passwords must be entered. To get *sifex* to prompt you for the password you must enter an *upper case Z* to the initial prompt.

```
sifex 3.6> Z

*** SECURITY PASSWORD ***
enter passWord:
```

You must now enter one of eight passwords in response to the above prompt. Each password corresponds to a letter in the prompt word, *passWord*; the letter that is set in upper case dictates what password you will use.

Following is the list of passwords and the corresponding *password* letter:

- If P use *carter*.
- If A use *ludwig*.
- If first S use *chase*.
- If second S use *darrah*.
- If W use *donl*.

- If O use *bradley*.
- If R use *ellis*.
- If D use *luttner*.

Since the above prompt has the W set in upper case, we used *donl* to gain access to the full set of *sifex* commands.

Now, enter your password.

```
                enter passWord: donl

accepted sifex 3.6>
```

We are now into *sifex*. Now, list the available commands using the **help** command.

```
                sifex 3.6> h

*** SIFEX -- COMMANDS ***
n     remote file copy routine
q     -quit and return to IRIS prom monitor.
s     -set miscellaneous variables.
t     -tape copy to disk utility.

b     -enter bad block edit mode.
c     -copy data.
e     -run drive read/write/seek tests.
f     -format the selected drive.
u     -update disk label.
```

Each of the above commands will take you into a different routine where specific functions are executed. During the next module, you will enter several of the routines and execute various commands within each routine in order to perform the complete task of rebuilding a disk drive.

NOTE: By pressing shift-3 (the # sign) an additional menu will be displayed showing a variety of additional commands available with SIFEX. However, discussion of these are beyond the scope of this course.
This program was initially created to allow manufacturing to prepare, test, and name disk drives. You will only learn the commands that you need to rebuild a damaged disk drive. Do not concern yourself with commands not covered by this procedure.

END OF PROCEDURE

SECTION 3: Test Items

1. Where do the FEX programs reside?

_____

2. Name the three FEX programs and list when you would use each one:

_____

_____

_____

3. List the prom monitor command that you would use to list the headers on the bootable tape:

_____

4. If you are some where in the FEX program and want to return to the main program (top level fex prompt), what key must you depress?

_____

Contents

FEX1 1-1
Introduction 1
Objectives 1
Resources 1
SECTION 1: Instructional Procedure 3
SECTION 2: Procedure Summary 14
SECTION 3: Test Items 17

FEX1

Introduction

In the last module, you learned how to load the *fex* programs and how to invoke the FE part of the program. In this module, you will learn how to use the disk utilities required to initialize it.

Objectives

At the completion of this lesson, you will be able to:

- Use "bad block" commands to list, modify, and add to the disk's bad block table.
- Format the disk drive.
- Exercise the disk drive.

Resources

This guide

Bootable cartridge tape

Disk drive bad block list

This module contains three sections:

- **SECTION 1:**

  This section is contains the instructional procedures and explanations for executing disk drive utility functions using the *fex* programs.

- **SECTION 2:**

  This section provides a summary of the procedures described in Section 1.

- **SECTION 3:**

  This section contains review questions that you should complete after studying the material in this module.

SECTION 1: Instructional Procedure

WARNING

The following exercises will destroy any existing data on the disk. Make sure you have made backup tapes of ALL file systems on the disk drive(s) on your workstation.

STEP 1: Selecting the Disk Drive.

Now that you have gained access to *sifex*, you must select the drive that you want to work on. To select a drive for formatting or testing you must enter the *set miscellaneous variables* routine using the *sifex* command **s**.

```
sifex 3.6> s

Set ?
```

Once into this routine, *sifex* asks you what variable you want to set. Now, list the available *set commands* using the **help** command.

```
Set ? h

*** Set commands ***
l     -change drive file system information
q     -quit and return to main routine
t     -set type of drive (? for list)
u     -set disk unit number

a     -display ALL settings
b     -bad blocks display
f     -set up the label
v     -set verbose
w     -write lock switch
```

To select a drive for testing or formatting use the **u** command.

```
Set ? u     Disk Unit (0) = 0
```

By entering the **u**, you told sifex that you wanted to select the unit. It shows you the default (0), and is waiting for you to accept the default or enter " 1 or f ". You will enter **0** for the training system. (0 = front drive, 1 = side drive)

At this moment all you needed to do is select the drive, which you have done, so **quit** this routine and return to the *sifex* control routine. You will return to this routine later.

```
Set ? q
```

```
Quit
```

STEP 2: Entering Bad Block Data.

Before you continue, it is recommended that you check your Bad Block Table and verify that it contains the entries that are listed on the sheet of paper attached to the top of the disk drive.

The Bad Block Table is used during formatting to define where bad spots are located on the disk surface. Knowing where the bad spots are, allows the track to be flagged as bad and the assignment of an alternate to be made.

NOTE: There is a chance that your Bad Block Table will not match the listing. This is due to the way SGI initially formatted some drives at the factory. If it does not match, then you must enter the data.

To change or display the Bad Block Table, you must enter the *bad block edit* routine using the **b** command.

```
sifex 3.6 >b  Bad Block edit, type h for help
bb >
```

Now, do as suggested and request help.

```
bb> h    *** Bad Block Commands ***
a-add bad blocks
c-clear bad block list
e-edit list
p-print list
q-quit
```

```
d-setup alternates
z-zap alternate assignments
```

First, you should to print the list to see if it matches the hard copy attached to the top of the drive.

```
bb >p  Print bad blocks:
```

You will see an output of the entries. They are in the format:

```
xxx/y

xxx = cylinder number
y   = head number
```

NOTE:
If your table does not match the hard copy listing, then perform the following NUMBERED steps, else skip the numbered steps and goto **STEP 3** after entering **q** to quit *bad block edit routine*. (The eight hundred numbers are the alternate tracks that the FEX program assigned to the existing bad blocks, ignore them)

1. Clear the bad block table by entering " **c** ".
2. Enter " y " to the " `confirm` " prompt.
3. Enter " a " to " `Add Bad Blocks` ".

The following prompt appears:

```
Add new entries.  Mode  cyl/hd, end with a blank line:
bb add:
```

NOTE:
Enter only the CYLINDER and HEAD information from the hard listing.

i.e.
31/0
921/6
135/3

When you are finished entering the bad block data, terminate the entry by depressing return without giving a " next " entry.

4. Return by itself to terminate the entries.....

Once you are satisfied that your Bad Block Table contains the correct entries, quit the Bad Block Edit mode.

```
bb > q

Quit
```

When adding a bad block the *add* command will take care of reformatting *reformatting* the bad track header with the new address of the track in the bad block area of the disk.

STEP 3: Formatting the Disk.

You will now instruct the program to format the disk. It will take about 8 minutes on the small disk.

```
sifex 3.6 >f

*** WARNING -- ALL DATA ON UNIT 0 WILL BE LOST!!!
Specific drive information goes here
.
.
.
```

```
Type 'go <return>' to start...
```

go <return>

The following output will accumulate on the screen...

```
Starting format....
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200
.....................................................................  ... 980
Formatting bad tracks....
Writing label....
Formatting complete.
```

NOTE:
The drive is now formatted and ready to receive your UNIX file systems, but the lab project will now show you how to test the drive for new bad spots. You would not format the disk followed by these tests in the field, since the tests destroy the format and you would then have to again run the format routine. This is backwards from the way you would normally work on the drive in the field.

STEP 4: Disk Exercise Tests.

You are now going to exercise the disk and see if any new bad spots are detected. If any new spots are detected you will re-enter the bad block edit mode and re-enter the new bad spot data.

The test takes about 8 minutes to complete one cycle. A thorough check would consume 12 - 24 hours, but run the test for the amount of time that you have.

You only need to run the test a couple of passes on your workstation. After the test is run, you will access an error table (maintained by sifex) to see if any new bad spots were detected.

You will now run the disk drive exercise tests....

sifex 3.6>

Now, again list the available *sifex* routines.

```
                sifex 3.6> h

*** SIFEX -- COMMANDS ***
n     -remote file copy routine
q     -quit and return to IRIS prom monitor.
s     -set miscellaneous variables.
t     -tape copy to disk utility.

b     -enter bad block edit mode.
c     -copy data.
e     -run drive read/write/seek tests.
f     -format the selected drive.
u     -update disk label.
```

Now, to select the disk exercise tests routine using the **e** command.

```
                sifex 3.6 > e

 Drive: Hitachi 512-17 Unit=0, (970=17/7/17 (512) ILV=1
Which exercise?
```

Since you have not been here before, list the available tests using the **l** command (stands for list).

```
                Which exercise? l    Exercise help -- Choose
from:

c-complete write/read multi pass/multi pattern.
d-disk read or write multiple.
e-error display/reset
m-multiple sector write/read repeated.
q-quit.
r-random reads.
Random number reset.
```

To select a test you must enter the *first letter* of the above listed tests. You are going to enter **c** to run the *complete write/read multi pass/pattern* test.

Notice that some of the tests begin with an *upper case* letter.

OK, lets run the test.

```
Which exercise? c    Complete Exercise -- track writes
and reads

Repeat how many times?
```

It is asking you for a number, if you depress **<return>**, the test will loop until you kill it by using the a key.

```
Repeat how many times? <return>

Alternate units?
```

Here the test is asking you if you want to alternate between the unit you have selected (unit 0) and the other disk drive. The default for this question is **no**, and thats what is selected if you depress **return**.

```
Alternate units? return
```

The following message is sequentially output and continues until you terminate the test using the a key. Let the test run two passes and then terminate it.

```
START LOOP 1      UNIT 0: Pattern  0xb1b6dbbd
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200
.....................................................................................
```

The next message will look the same except for the loop count.

After two passes, terminate the test using the a key.

```
a

Which exercise?
```

While the test was running, a log is maintained by the program to record any disk errors detected during the test. You must now display the log to check if any errors were detected.

NOTE:
The longer the test is allowed to run, the greater the chance of detecting new bad spots, if they exist.

Now, see what command you use to print the *error log*.

```
                Which exercise? l    Exercise help -- Choose from:

c-complete write/read multi pass/multi pattern.
d-disk read or write multiple.
e-error display/reset
m-multiple sector write/read repeated.
q-quit.
r-random reads.
Random number reset.
```

The command is **e**.

```
                Which exercise? e     Error display or Reset?
```

It is asking you if you want to *display* or *reset* the error log. You should now *display* it.

```
                Error display or Reset? d
```

If any new errors were detected, they would be listed here.

Now, quit the disk exercise routine.

```
                Which exercise? q

                sifex 3.6>
```

If you have new errors, then you must again enter the *bad block edit* routine and enter the *head* and *cylinder* information into the *bad block table*, goto **STEP 5.**

If no new errors were detected, then goto **STEP 6.**

STEP 5: Adding new Bad Blocks to Bad Block List.

NOTE:
The following steps just show the commands you enter and not the prompts or the screen output since you have already seen it.

1. Enter `b` (This enters Bad Block Edit routine)
2. Enter `a` (Add bad blocks -- Enter new bad spots as before)
3. Depress `<return>` (to terminate the edit process)
4. Enter `q` (quit edit routine)
5. Enter `e` (This enters exercise test routine)
6. Enter `e` (This enters error display or reset mode)
7. Enter `r` (This resets the error log)
8. Enter `q` (quit exercise test routine)

STEP 6: Re-formatting the disk drive.

You must now run the *format* program again, the reasons why:

- If you had to enter additional *bad block* information (**STEP 5**) into the *bad block table*, then, running *format* will assign alternate tracks for the newly added bad tracks, or,
- You entered this step from **STEP 4**, where you ran the complete set of tests against the drive, which destroyed all the format on the disk.

In either case, you must again format the disk.

The procedure would be the same as it was in **STEP 3**, except this time you **would not run** the exercise tests, you would jump to **STEP 7** after the format was complete.

At this point you have a disk drive that is formatted and ready to receive the operating system. You will now restore the ROOT file system using the **Tape Copy to Disk** utility portion of sifex.

STEP 7: Copy Root File System From Tape to Disk.

You must now enter the *tape copy to disk utility* routine.

```
sifex 3.6 > t   Tape to Disk Copy

Tape file (2) ?
```

Sifex is asking you if you wanted the second file on tape. You say yes by entering <return> and accepting the default of 2. The ROOT file system is the second file on the Bootable tape.

```
Tape file (2) ?<return>

Unit (0) =
```

Its now asking you what drive to restore the filesystem to; disk drive 0 being the default. Depressing **return** accepts the default.

```
Unit (0) =<return>

File system (a) ?
```

Now it wants to know what partition to install the filesystem into; the default is **a**, and that is where the ROOT file system will be written. Depressing **<return>** will accept the default.

```
File system (a) ?<return>

Copying 17850 blks in 960 chunks from tape file 2 to si0a
Type 'go <return>' to begin.....
```

go <return>

```
Started
Rewinding....
Copy started....
5 10 15 20 25 30 35
Rewinding......
Tape to Disk copy complete
```

```
sifex 3.6 >
```

You are now ready to quit *sifex* and boot IRIX. Once IRIX comes up, you will load the user file system into partition " c ".

```
sifex 3.6 >q    Quit -- confirm quit with 'y':
```

If you really want to quit *sifex*, then enter **y**.

```
Quit -- confirm quit with 'y': y
```

At this point the workstation is rebooted by *sifex* and you are placed into the PROM monitor.

SECTION 2: Procedure Summary

This section shows only the actual steps of the lab project.

The IRIS OWNER'S GUIDE, Series 3000, gives an outline of the Tape Restore portion of this procedure.

STEP 1: Reboot the workstation

STEP 2: BOOTING SIFEX FROM TAPE:

1. Insert the bootable tape.
2. Enter " h " for prom monitor command listing.
3. Enter the proper command to boot the FEX program from tape.

STEP 3: Entering Field Engineer portion of SIFEX

4. Enter an upper case Z.
5. Enter the correct password.

STEP 4: SELECTING THE DISK DRIVE

6. Type " s " to set miscellaneous variables.
7. Type " u " to select the drive.
8. Enter " 0 " or " 1 " to select drive to be restored.
9. Type "q" to quit set misc. variables.

STEP 5: ENTER BAD BLOCK DATA:

10. Type " b " to enter bad block edit mode.
11. Type " p " to print the current bad block table.

NOTE:
If the hard copy listing MATCHES the bad block table, execute "q" and go to STEP 6, otherwise execute the following and then go to STEP 6.

A. Type " c " to clear bad block table.
B. Type " y " to confirm clearing the table.
C. Type " a " to add bad block entries.
D. Now enter each entry from the hard copy listing.

Use the following format: Cylinder/head.

i.e. 31/0, 921/6.

E. Depress <return> without making an entry to terminate the entry process.

12. Type " q " to quit bad block edit mode.

STEP 6: FORMAT DISK:

13. Type " f " to format the disk.
14. Enter " go " to begin the process. (Takes about 8 minutes)

STEP 7: EXERCISE DISK:

19. Type " e " to enter exercise mode.
20. Type " c " to run the complete read/write test.
21. Depress <return> to loop the test.
22. Depress <return> to test only the selected drive.

NOTE:
One pass takes about 8 minutes. This test should be run at least 12 hours and preferably 24 hours for a thorough check of all the disk, but do what time allows you.

23. Depress " a " to terminate the test.
24. Type " e " to enter error display.
25. Type " d " to display any accumulated errors.
    Record any errors on a sheet of paper.
26. Type " q " to quit exercise mode.

NOTE:
If the exercise test DID NOT detect any new errors, then skip to STEP 9.

STEP 8: ADD NEW BAD BLOCKS TO BAD BLOCK TABLE:

A. Type " b " to enter bad block edit mode.
B. Type " a " to add bad blocks.
C. Enter each new bad block: format= cyl/head.
D. Depress <return> without an entry to terminate.
E. Type " q " to quit bad block edit mode.
F. Type " e " to enter exerciser routine.
G. Type " e " to enter error display or reset mode.

H. Type " r " to reset the error log.
I. Type " q " to quit exerciser routine.

STEP 9: Re-formatting the Disk Drive.

27. Type " f " to format the disk.
28. Enter " go " to start the format.

STEP 10: COPY ROOT FILE SYSTEM FROM TAPE TO DISK:

29. Type " t " for tape utility program.
30. Depress <return> to copy from tape file #2.
31. Depress <return> to copy to the selected drive.
32. Enter " a " to copy to partition " a ".
33. Enter " go " to begin copying.
34. Type " q " to quit FEX.
35. Type " y " to confirm quit.

STEP 11: COPY /USR FILE SYSTEM FROM TAPE TO DISK:

36. Boot UNIX from disk: Enter " b ".
37. Enter " mkfs /dev/rsi0f " to create user file partition.
38. Enter " mount /dev/si0f /usr " to mount user file system.
39. Enter " cd /usr " to change directory to /usr.
40. Enter " mt rew " to rewind tape.
41. Enter " mt fsf 2 " to skip tape forward two files.
    (PERFORM THIS ONLY IF THE MKBOOT TAPE CONTAINS /USR.
42. Enter " cpio -ivhmud2 " to copy in the user file system.

STEP 12:RESTORE DISK DRIVE LABELS:

43. Enter the " sgilabel " to get usage of disk label command:
    Usage: sgilabel [-n name] [-s serial#] dv#
    i.e. sgilabel -n "Release GL2-W3.5" -s 14596 si0
44. Enter " sync " to sync the disk drive.
45. Label the USER file system first:
    i.e. labelit /dev/si0f usr sgi
46. Label the ROOT file system:
    i.e. labelit /dev/si0a root sgi
47. Depress the RESET button to reboot the system.
48. Boot up UNIX and go to MULTI-USER to ensure proper operation.

SECTION 3: Test Items

1. What FEX command would you use to enter " Bad Block Edit Mode"?

2. If you entered SIFEX to add bad block data that you received from UNIX during normal operation, after entering the cyl/hd information, would you need to format the disk?

3. If you answered yes to question #5, then answer #6. Why do you need to format the disk if all you are doing is adding a new entry to the already existing table?

Contents

RSTR1 1-1
- Introduction 1
- Objectives 1
- Criterion Test 1
- Resources 1
- SECTION 1: Instructional Procedure 3
  - Disk Restoration Procedure 3
- SECTION 2: Procedure Summary 10
- SECTION 3: Test Items 14

Introduction

Now that you have formatted your disk with *fex*, you are ready to restore the file systems you backed up, using the *bootable* tape and backup of the *usr* file system.

Objectives

At the completion of this lesson, you will be able to:

- Restore the "root" filesystem using "fex".
- Restore the "usr" filesystem using "cpio".
- Label the disk device using the "sgilabel" command.
- Label each filesystem using the "labelit" command.

Criterion Test

Given a freshly formatted disk drive, correctly perform the procedures necessary to restore the "root" and "user" file systems and file system/disk labels.

Resources

This guide

IRIX Programmer's Manual - Volume 1A and 1B

Mkboot tape

/usr backup tape

This lab project contains three sections:

Section 1: This section is used to teach you the procedure and explain to you what you are doing.

Section 2: This section lists just the procedure without all the instructional text.

Section 3: This is a number of review questions that you will complete after performing the lab project.

SECTION 1: Instructional Procedure

Disk Restoration Procedure

The procedures for disk restoration involves several steps. They are:

- Loading the *fex* utility from tape (or disk from the */stand* directory).
- Restoring the "root" file system from the "Bootable" tape.
- Booting IRIX from the new root file system.
- Executing the *mkfs* utility, creating the empty *usr* file system.
- Restoring the */usr* file system from the *backup* tape.
- Labeling the disk and file systems.

STEP 1: Copy Root File System From Tape to Disk.

You must now enter the *tape copy to disk utility* routine.

```
     sifex 3.6 > t    Tape to Disk Copy

Tape file (2) ?
```

Sifex is asking you if you wanted the second file on tape. You say yes by entering <return> and accepting the default of 2. The ROOT file system is the second file on the Bootable tape.

```
     Tape file (2) ?<return>

Unit (0) =
```

Its now asking you what drive to restore the filesystem to; disk drive 0 being the default. Depressing **return** accepts the default.

```
     Unit (0) =<return>

File system (a) ?
```

Now it wants to know what partition to install the filesystem into; the default is *a*, and that is where you want ROOT to go. Depressing <return> will accept the default.

```
     File system (a) ?<return>

  Copying 17850 blks in 714 chunks from tape file 2 to si0a
  Type 'go <return>' to begin.....


     go <return>

  Started
  Rewinding....
  Copy started....
  5 10 15 20 25 30 35.......
  Rewinding......
  Tape to Disk copy complete
sifex 3.6 >
```

You are now ready to quit *sifex* and boot IRIX. Once IRIX comes up, you will load the user file system into partition " c ".

```
sifex 3.6 > q   Quit -- confirm quit with 'y':
```

If you really want to quit *sifex*, then enter y.

```
Quit -- confirm quit with 'y': y
```

At this point the workstation is rebooted by *sifex* and you are placed into the PROM monitor.

STEP 2: Copy User File System From Tape to Disk.

The next step is to boot the system using the ROOT file system that you just installed.

```
iris > b

SGI Extent Filesystem
Loading: hd:0:defaultboot
  Text: 038318 bytes
  Data: 0113d8 bytes
  Bss:   024d7c bytes
Jumping to load program @ 20000400


SYSTEM 5 IRIX #0 [Wed May 7 04:49:59 PDT 1986]
(C) Copyright 1986 - Silicon Graphics Inc.
real = 4194304
kmem = 561152
user = 3633152
bufs = 819200 (max=16k)
dsd0 not installed
qic0 not installed
sii0 at mbio 0x07200 ipl 5
si0 (Hitachi 512-17 name: Hitachi 512-17) slave 0
si1 not installed
sf0 floppy (80/2/8) slave 2
siq0 at mbio 0x73fc ipl 5
sq0 (qic02 cartridge tape) slave 0
```

```
iph0 not installed
tmt0 not installed
ik0 not installed
ib0 not installed
ib1 not installed
px0 not installed
cd0 not installed
cd1 not installed
cd2 not installed
cd3 not installed
hy0 not installed
ex0 (FW 2.5 HW 4.0) (0800.1400.3948) at mbio 0x7ffc ipl 2
fpa installed
lpen not installed
kernel debugger disabled.
root on si0a
swap on si0b. swplo=0 nswap=64000

INIT: SINGLE USER MODE

#
```

You are now in SINGLE-USER mode.

With the next step you will create the user partition that you will restore the *usr* filesystem into.

```
# mkfs  /dev/rsi0f

isize = 1920
filsys = /dev/rsi0f,  size = 79704: 7680,  unused 26
heads = 10,  Sectors = 32
cgsize = 3984:384,  firstcg = 24,  ncgs = 20
```

Next you will mount the *usr* filesystem.

```
# mount  /dev/si0f  /usr

WARNING!! -- mounting: < > as </usr>
```

The above warning occurs because the new *user* filesystem (/dev/si0f) does not have a label, you will label it later.

Now change directories to the *usr* directory.

```
# cd  /usr
```

Now, load the *cpio* backup tape of the *usr* file system into the cartridge tape drive.

Make sure it is rewound by typing:

```
# mt  rew
```

Restore the *usr* filesystem by typing:

```
# cpio  -ivhmud2
```

As each file is read, its pathname is displayed on the screen. The restore should take about 15 minutes (longer for larger systems).

NOTE: To verify that the file systems are now installed on the system, perform the following commands. It will show you the standard system is now present. It would also show you any optional products that are installed.

```
# cd  /

# cat  /Versions/*

upd       004-0144-045   GL2-W3.6 Update System, GL2-W3.6
man       004-0154-035   GL2-W3.6 Update Manual Pages, GL2-W3.6
demos     004-0174-035   GL2-W3.6 Update Demos, GL2-W3.6
gifts     004-0184-032   GL2-W3.6 Update Gifts, GL2-W3.6
mail      004-0194-032   GL2-W3.6 Update Mail, GL2-W3.6
troff     004-0304-032   GL2-W3.6 Laser Printer Opt, GL2-W3.6
emacs     004-0374-034   GL2-W3.6 Emacs Option
nfs       004-0384-032   GL2-W3.6 NFS Opt, GL2-W3.6
sys       004-0631-024   GL2-W3.5c Reconfigurable Kernel Option
trb       004-0911-012   GL2-W3.5 Sales Demos
root      004-0134-162   GL2-W3.6       Standard System (root)
```

You are almost done. Now you must restore the disk drive labels.

STEP 3: Restoring Disk Labels.

First, you will restore the disk label using the **sgilabel** command. The following shows the correct syntax:

```
Usage: sgilabel [- n name] [-s serial #]  dv#
```

Now, label the drive.

```
# sgilabel  -n "Release GL2-W3.6"  -s 20638  si0
```

In this example, the drive serial number was *20638*, the name used indicates the release of software installed (your name can be what you want), and the device was *si0*.

Now, look and see if you really did label the drive. To do this, again use the **sgilabel** command.

```
# sgilabel  /dev/si0

/dev/si0: Name: Hitachi 512-17, Serial: 10064
     drive: Hitachi 512-17, controller: Interphase 3030
     cylinders/heads/sectors(512 byte): 823/10/32
     alternate cylinder/# of alt cylinders: 806/17
     badtracks=26, interleave=1, trkskw=0, cylskw=5
               fs     base          size
                      sectors(cylinders)
               a:     320(   1),   18880(  59) Root
               b:   19200(  60),   32000( 100) Swap
               c:   51200( 160),   64000( 200)
               d:  115200( 360),  142720( 446)
               f:   51200( 160),  206720( 646)
               g:     320(   1),  257600( 805)
               h:       0(   0),  257920( 806)
```

Next, restore the filesystem labels using the *labelit* command.

Before you write the labels you must *sync* the disk drive.

```
# sync
```

Following is the correct syntax for the **labelit** command.

```
Usage:  labelit  /dev/r????  [ fsname  volume  [-n]]
```

The question marks designate the file system you want to label. The " -n " option skips the label check before labeling. (I will not use the *-n* option)

Label the *usr* filesystem first.

```
#labelit  /dev/rsi0f  usr  sgi

Current fsname: , Current volname: , Blocks: 79704, Inodes: 7680
FS Units: 512b,  Date last mounted: Sat  June 21  18:00  1986
New fsname = usr,  New volname = sgi
#
```

Next, label the *root* filesystem.

```
#labelit  /dev/rsi0a  root  sgi

Current fsname: , Current volname: , Blocks: 17848, Inodes: 7660
FS Units: 512b,  Date last mounted: Sat  June 21  18:00  1986
New fsname = usr,  New volname = sgi
#
```

You must now reboot the system by *depressing the reset button.* This will insure that the labels get updated.

When the PROM Monitor is entered, boot the system using the normal method.

Your customer would now be up and running. The restoring of additional files would be a customer task. You got the system back to the basic system, the customer would then use the backup tapes to restore the full system.

SECTION 2: Procedure Summary

This section shows the steps taken to restore the disk from beginning to end.

STEP 1: Reboot the workstation

STEP 2: Boot SIFEX from tape:

1. Insert the bootable tape.
2. Enter " h " for prom monitor command listing.
3. Enter the proper command to boot the FEX program from tape.

STEP 3: Enter the Field Engineer portion of SIFEX:

4. Enter an upper case Z.
5. Enter the correct password.

STEP 4: Select the disk drive:

6. Type " s " to set miscellaneous variables.
7. Type " u " to select the drive.
8. Enter " 0 " or " 1 " to select drive to be restored.
9. Type "q" to quit set misc. variables.

STEP 5: Enter bad block data:

10. Type " b " to enter bad block edit mode.
11. Type " p " to print the current bad block table.

NOTE:
If the hard copy listing MATCHES the bad block table, execute "q" and goto STEP 6, else execute the following and then goto STEP 6.

A. Type " c " to clear bad block table.
B. Type " y " to confirm clearing the table.
C. Type " a " to add bad block entries.
D. Now enter each entry from the hard copy listing.
   Use the following format: Cylinder/head.
   i.e. 31/0, 921/6.

E. Depress <return> without making an entry to terminate the entry process.

12. Type " q " to quit bad block edit mode.

STEP 6: Format the disk drive:

13. Type " f " to format the disk.
14. Enter " go " to begin the process. (Takes about 8 minutes)

STEP 7: Exercise the disk drive:

15. Type " e " to enter exercise mode.
16. Type " c " to run the complete read/write test.
17. Depress <return> to loop the test.
18. Depress <return> to test only the selected drive.

NOTE:
One pass takes about 8 minutes. This test should be run at least 12 hours and preferably 24 hours for a thorough check of all the disk, but do what time allows you.

19. Depress " a " to terminate the test.
20. Type " e " to enter error display.
21. Type " d " to display any accumulated errors.
    Record any errors on a sheet of paper.
22. Type " q " to quit exercise mode.

NOTE:
If the exercise test DID NOT detect any new errors, then skip to STEP 9.

STEP 8: Add new bad blocks to the bad block table:

A. Type " b " to enter bad block edit mode.
B. Type " a " to add bad blocks.
C. Enter each new bad block: format= cyl/head.
D. Depress <return> without an entry to terminate.

E. Type " q " to quit bad block edit mode.
F. Type " e " to enter exerciser routine.
G. Type " e " to enter error display or reset mode.
H. Type " r " to reset the error log.
I. Type " q " to quit exerciser routine.

STEP 9: Re-format the disk drive:

23. Type " f " to format the disk.
24. Enter " go " to start the format.

STEP 10: Copy the root file system from tape to disk:

25. Type " t " for tape utility program.
26. Depress <return> to copy from tape file #2.
27. Depress <return> to copy to the selected drive.
28. Enter " a " to copy to partition " a ".
29. Enter " go " to begin copying.
30. Type " q " to quit FEX.
31. Type " y " to confirm quit.

STEP 11: Copy the usr file system from tape to disk:

32. Boot IRIX from disk: Enter " b ".
33. Enter " mkfs /dev/rsi0f " to create user file partition.
34. Enter " mount /dev/si0f /usr " to mount user file system.
35. Enter " cd /usr " to change directory to /usr.
36. Enter " mt rew " to rewind tape.
37. Enter " mt fsf 2 " to skip tape forward two files.
38. Enter " cpio -ivhmud2 " to copy in the user file system.

STEP 12: Restore the disk labels:

39. Enter the " sgilabel " to get usage of disk label command:
    Usage: sgilabel [-n name] [-s serial#] dv#
    i.e. sgilabel -n "Release GL2-W3.6" -s 14596 si0
40. Enter " sync " to sync the disk drive.
41. Label the USER file system first:
    i.e. labelit /dev/si0f usr sgi
42. Label the ROOT file system:
    i.e. labelit /dev/si0a root sgi
43. Depress the RESET button to reboot the system.

44. Boot up IRIX and go to MULTI-USER to ensure proper operation.

SECTION 3: Test Items

1. Where do the FEX programs reside?

_____

2. Name the three FEX programs and list when you would use each one:

_____

_____

_____

3. List the PROM Monitor command that you would use to list the headers on the bootable tape:

_____

4. What FEX command would you use to enter " Bad Block Edit Mode"?

_____

5. If you entered SIFEX to add bad block data that you received from IRIX during normal operation, after entering the cyl/hd information, would you need to format the disk?

_____

6. If you answered yes to question #5, then answer #6. Why do you need to format the disk if all you are doing is adding a new entry to the already existing table?

7. Once IRIX is up, what command do you use to create a new disk partition?

8. What command is used to label a disk drive?

9. What command is used to label a file system?

10. If you are some where in the FEX program and want to return to the main program (top level fex prompt), what key must you depress?

Title

How To Use Special Boot Strap Tape

Models Affected	Software Release
3xxx or 2400t with Storager disk controllers	N/A

Parts / Documentation Affected	Part Number	Rev	DA
Special bootstrap tape	800-1020-001		
Instructions	080-0024-001		

Reported By	Date Reported
Terry Drasny	09/13/88

Assigned To	Date Resolved
Paul Bell	01/04/89

Effectivity	Reference
Immediate	Attached - Special Boot Tape Instructions

Description

The Special Boot Strap tape is needed when the customer mkboot tape was made from a smaller disk drive than the replacement disk. This will happen when the customer buys a 380 megabyte disk upgrade and replaces the 170mb system disk (0) with 380mb disk.

The Special Boot Strap tape has a copy of the sifex program and a special Unix mini-root operating system. The Tape to Disk Utility is used from the sifex program to write the Mini-root to the swap partition (b) of disk 0 and the sifex program is quit. Then the Special Unix operating system is booted from the (b) partition and restores the root partition from the customers mkboot tape. The Special unix operating system will automatically login as root and will execute a .profile shell script. The profile prompts the user to install customer mkboot tape into the tape drive then starts the dd read for tape and write to root partition of disk 0.

Note: When the dd program finds the EOF mark (end of file) of the tape it flags an error, this is expected. This Special Boot Strap tape is only valid for the 3000/2400T systems with Storager disk controllers and will not work on other systems with DSD controllers.

Action Required

None...For your information only!

Resolution/Recommendation

The following procedure explains step by step HOW TO USE Special Boot Strap tape.

Follow these steps if system is 3xxx/2400T and Storager controller:

(1) Boot the sifex program from Special boot Strap tape if the 170MB is being replaced by 380MB and the prompt will reflect the sifex program loaded.
Example: sifex>

Use the following command to boot sifex from Special Tape:
iris> ct0:sifex <return>

Type "h" <return>
"This will invoke the help command and list all valid commands"

Type "t" <return>
"This invokes Tape to Disk copy"

Tape file (2)? <return>
"This takes the default Tape file number 2"

Unit (0)= <return>
"This selects the default for the Unit address 0"

File System (a)? Type "b" <return>
"This selects the swap partition (b)"

The system will then print:
copying 600 blks in 500 chunks from tape file 2 to si0b
Type 'go<return>' to begin...

To start the copying
Type "go" <return> or <return> to abort.

The system will then print:
spacing forward 1 files....
copy started....
1 2 3 4 5 6 7 8 9 10 11 12
rewinding....
Tape to Disk Copy complete

After copy has completed exit sifex
Type q <return>
Type y <return> --confirm quit with "y"
"This quits sifex"

(2) The Prom Monitor prompt will be displayed and the next step is to boot the special vmunix program loaded on the swap partition "b".

iris> b si0b:vmunix <return>

(3) Once the special vmunix program is booted the system will halt at single user mode and automatically start a shell script to prompt user to restore mkboot to new disk drive.
The system will print the following messages and prompt user:

INIT: SINGLE USER MODE

make new system (y/n) ?
"Type y <return>"

Load the system (customer) mkboot tape into tape drive

Is the tape loaded (y/n) ?

The profile script is now copying the root file system from tape with a DD command too root partition si0a. When the dd command reaches the END OF FILE MARK the system will print out an error:
dd read error : I/O error
The most important thing is that the number of records transferred (in) match the number (out)..
Example:

235+1 records in
235+1 records out

reboot ? "Type n <return>"

Before rebooting the system make sure that the system will boot into single user mode and not multi-user mode. Make sure the /etc/inittab file reflects this mode request.
Example of /etc/inittab set for single usr mode:
is:s:initdefault: #default state
Now the system can be rebooted with the "reboot" command.

(4) Now that the root (/) file system has been restored the system can be booted to SINGLE USER MODE and the usr file system can be restored. The customer may wish to increase the swap partition before doing mkfs and restoring usr. This is the best time to make a new "mkboot" tape, then restore the usr file system.

Reference Owner Guide section 6.2 Iris Workstation Disk Configuration to change swap partition size and make usr file system.
The size of any partition in megabytes can be Calculated by the "number of sectors" multiplied by number of data bytes per sector (512).
Example for Micropolis 380:

sgilabel si0: (command to print out disk label)
The size of B partition is 36750. (36750 * 512 = 18.816 mb)

Title

Modifing 85 Mb Toshiba disk label.

Models Affected

2xxx/3xxx

Software Release

GL2-W3.5 or GL2-W2.4

Parts / Documentation Affected

Disk Drive, 85MB Toshiba, ST506

Part Number 9410017

Rev

DA

Reported By

John Mckinley

Date Reported

12/09/87

Assigned To

Paul Bell

Date Resolved

04/20/88

Effectivity

Immediate

Reference

None

Description

This is a procedure for modifing the disk label for the 85 meg. Toshiba when the Iris sytem is not at software release GI2-W3.6. The usr disk partitions overlap the bad tracks area, and when the disk is almost completely full the usr data can be lost. This problem can be corrected by manually modifing the disk label, zapping the bad block table, formatting the disk and then recovering all data from mkboot and usr backup tapes.

A halt in shipments has been ordered to keep the Toshiba drives from going out to both repair and upgrade customer. When GL2-W3.6 software is released, the drives will be permitted ship (early January 88).

In special cases we will ship a special tape to field engineers to properly support the limited number of Toshibas now in the field.

Action Required

None...For your information only!

Resolution/Recommendation

** Warning: This procedure will distroy all data written on the disk drive. Before doing this procedure, the system should be completely backed-up with a fresh mkboot tape of root and a complete copy of /usr.

1) Check disk drive unit address switches located bottem of drive before installing new disk drive.

switches 1 2 3 4 5 6 7 8
drive 0 1 0 0 0 0 0 1 0
drive 1 0 1 0 0 0 0 1 0

2) Boot Fex from mkboot tape.

3130 b ct0:sifex
2500 b mt0:ipfex
2400 b mt0:mdfex

3) Select unit for testing. Note: Prompt will change to SET?>

a) Type h for help to list commands.
b) Type s for set misc functions.
c) Type h for help to list commands.
d) Type u for unit select for testing.
e) Type "1" or "0" as appropriate.
f) Type q for quit set.

4) Get into protected Fex.

a) Type Z to access protected fex.
b) Enter Password "donl" note: if you blow it, start again at Step 2.
c) Prompt = xxfex> xx = md, si, or ip
d) Type h for help to list commands.

5) Select bad block edit mode. Note: Prompt will change to bb>

a) Type b for edit bad block.
b) Type h for help to list commands.
c) Type p for print list of bad blocks.
d) Copy down the list of bad blocks cylinder/heads only, not alternate track assignments.

Example:
bad blocks cyl/hd: cyl/hd alternate
example: 171/1: 810/1

This is the bad block 171/1.
This is the alternate track assignment 810/1.

e) Type z for zap alternate assignments.
f) Type q for quit bad block edit.

6) Change disk label alternate cylinder information from 810/20 to 820/10 and check all other values. For 2XXX Iris only, Change the B & G disk partition values and make G the Boot device.
Note: Prompt will change to Set?>
a) Type s for set misc functions.
b) Type h for list of commands.
c) Type u for select unit for testing.
d) Type c for change drive label information.
1) Label: #7
2) Name: MK56FB
3) Serial#: Drive Dependent.
4) Controller: (Qualogy 5217) <return>
5) Cylinders: (830) <return>
6) Heads: (10) <return>
7) Sectors: (17) <return>
8) Alternates: (810) Change to (820) <return>
9) # of Alternates: (20) Change to (10) <return>
10) Interleave: (1) <return>
11) Track Skew: (0) <return>
12) Cylinder Skew: (11) <return>
13) File systems info: lba or cylinder entry? <return>
14) Read label below for Base & Size for (CYLINDER NUMBERS).
e) Type q for quit.

The following is the new 85 meg Toshiba disk label to be used.

```
Name: Toshiba MK56FB, Serial: xxxx
drive: mk56fb type (36), controller: Qualogy 5217
cylinders/heads/sectors(512 byte): 830/10/17
alternate cylinder/# of alt cylinders: 820/10  *Was 810/20*
badtracks=xx, interleave=1, trkskw=0, cylskw=11
fs      base          size
        sectors(cylinders)
 a:     170(   1),  18700( 110) Root
*b:   18870( 111),  16830(  99) Swap
 c:   35700( 210), 102680( 604)
 d:   35700( 210),  51340( 302)
 e:   87040( 510),  51340( 302)
 f:     170(   1), 139230( 819)
*g:       0(   0),      0(   0)
 h:       0(   0), 139400( 820)
*Root: (a)  Swap: (b)  Boot: (a)
```

*Note: Change the following partitions for 2000 systems only.

```
b:  18870( 111),  16320(  96) Swap
g:  35190( 207),    510(   3)
Root: (a)  Swap: (b)  Boot: (g)
```

7) Update disk label and format.
   a) Type u for update disk label.
   b) Type y for YES to update disk label.
   c) Type f for format disk.
   d) Type go to start format.

8) Restore Root & Usr disk partitions with backups made earlier, and follow the Crash Recovery in SGI's Owners Guide.

Contents

ARCH1 1-1
- Introduction 1
- Objectives 1
- Sample Criterion Test Item 1
- Resources 1
- SECTION 1: Graphics Subsystem Architecture 3
  - Introduction 3
  - 3000-Series Bitplanes 4
  - BIT PLANE BOARD: BP3 6
  - BIT PLANE DATA 7
  - COLOR MAP 8
  - BP3 PHYSICAL ORGANIZATION 10
  - The Geometry System 25
  - The GF2 Board 27
  - The Update Controller Board 29
  - The Display Controller 31
- SECTION 3: Test Items 32

ARCH1

Introduction

In this module on the graphics subsystem and its architecture, you will see alot of similarities to the 4D-Series layout. A block-level description of the major component functions is provided and the appropriate configuration information for each board.

Objectives

At the completion of this module, you will be able to:

- Describe the physical organization of BP3 Bit Plane boards.
- Match the color programming mode to its correct function.
- Map the system bitplane identifiers to the correct BP3 board.
- List the function performed at each stage of the transformation process.
- Identify the logical functions of the GF2 board.
- Identify the logical functions of the UC4 board.
- Identify the logical functions of the DC3 board.

Sample Criterion Test Item

On an unlabeled graphics pipeline block diagram, label each Geometry Engine with its correct function.

Resources

This guide

This module contains two sections:

- **SECTION 1:**

  This section is contains the instructional procedures and explanations describing the architecture of the graphics subsystem.

- **SECTION 2:**

  This section contains review questions that you should complete after studying the material in this module.

SECTION 1: Graphics Subsystem Architecture

Introduction

Consider a "generic" computer graphics system such as the one shown in the Exhibit.It consists of the following components:

- graphics processor
- frame buffer controller
- update controller
- display controller

These components are part of most computer graphics system and responsible for the following functions:

- The **graphics processor** is responsible for *transforming* raw graphics data from the system CPU into X and Y coordinate data. On SGI work stations, the *geometry pipeline* performs this function.
- The **frame buffer controller** takes transformed graphics data and performs additional calculations. Among them are:
  - generating drawing commands to the update controller.
  - performing *Z-buffer arithmetic* for hidden surface removal.
  - calculating shade range values for depth-cued lines.
  - calculating shade ranges for Gouraud shaded objects.
- The **update controller** does the following:
  - decides how lines will be drawn by determining which pixels to turn on between line endpoints.
  - generates fill patterns for polygons.
  - draws characters (from font memory on the UC4 board).
  - writes data to the *frame buffer* (bitplane memory).

- The display control performs the final *scan conversion* of the bitplane data by:
  - synchronizing the reading of bitplane data with operation of the update controller.
  - converting the digital picture data to analog values.
  - generating the analog video signals to drive the raster scan system pf the display monitor.

3000-Series Bitplanes

QUESTION:

What is a **Bit Plane?**

Answer:

It is a Random Access Memory (RAM).

Question:

How large is a single BP3 Bit Plane?

Answer:

Each IRIS bit plane is **1024 x 1024 x 1.**

Question:

What is the bit plane used for?

Answer:

Storage of a single bit of digital coded color data for each one of the pixels on the face of the CRT.

- The bit stored for each pixel carries the same relative bit weight within the 8-bit color code, in other words - the bits in the plane represent one of the bit weights (1, 2, 4, 8, 16, 32, 64 or 128) of the 8-bit color code.

- Only 1024 x 768 locations are used of the possible 1024 x 1024.

Question:

Why are only **1024 x 768** locations used?

Answer:

The maximum monitor resolution is **1024 x 768.**

- The bit plane is an exact copy of the pixel arrangement on the face of the CRT.
  - **1024 pixels wide.**
  - 1024 pixels high (lines), of which, **only 768 are used.**

BIT PLANE BOARD: BP3

Question:

A Bit Plane board (BP3) is comprised of how many bit planes?

Answer:

4 bit planes.

Question:

Are the bit planes identical?

Answer: Yes.

- The same address lines drive all four bit planes on BP3, and any other BP3 boards that may be installed.
- The same relative location in each bit plane stores a single bit of the 8-bit color code for the same relative pixel on the CRT screen.

Question:

Are the bit weights (for the color code) of each plane the same?

Answer:

NO - Each bit plane will store a different bit weight of the color code. (see diagram)

BIT PLANE DATA

Question:

Do the bit planes store more than one type of data?

Answer:

YES - Three types of data can be stored in the bit planes depending on the programming mode: **RGB MODE, COLOR MAP MODE,**

Z-BUFFER MODE.

- If in **RGB MODE**, the bit planes hold digital color codes for each of the primary colors.

  RGB Mode requires a minimum of 6 BP3 boards.

  - Each color is defined by an 8-bit code.
  - 3 x 8 = 24 bits.
  - Each BP3 will hold 4 bits, therefore,

    24 divide by 4 = 6 BP3's (more later)

- If in **COLOR MAP MODE**, the bit planes will hold an address

  (index) into another RAM, called the **COLOR MAP.** (more later)

- If in **Z-BUFFER MODE**, some of the bit planes will hold

  Z-coordinate (depth information) values for **Hidden Surface Removal** functions. (more later)

  While in Z-Buffer Mode, 12 Bits planes will also hold Color Map Mode data.

COLOR MAP

- The **Color Map** is located on the **Display Controller** (DC4).
- It is a **4096 x 24** bit RAM.
  - Addressed with **12** bits which produces 4096 locations.
  - The 24 data bits store three 8-bit color codes - one for each primary.
  - At any point in time, the Color Map can define (map) **4096** colors out of the possible 16.7 million.
- The color map is addressed from two places: **Bit Planes & Multibus.**
  - **Bit Planes**

    When in **Color Map Mode**, the bit planes supply the **12-bit address**, which is an index into a specific location. The three 8-bit (24 bits) color codes are output to the **DAC's** - This is a table-look-up function.
  - **Multibus**

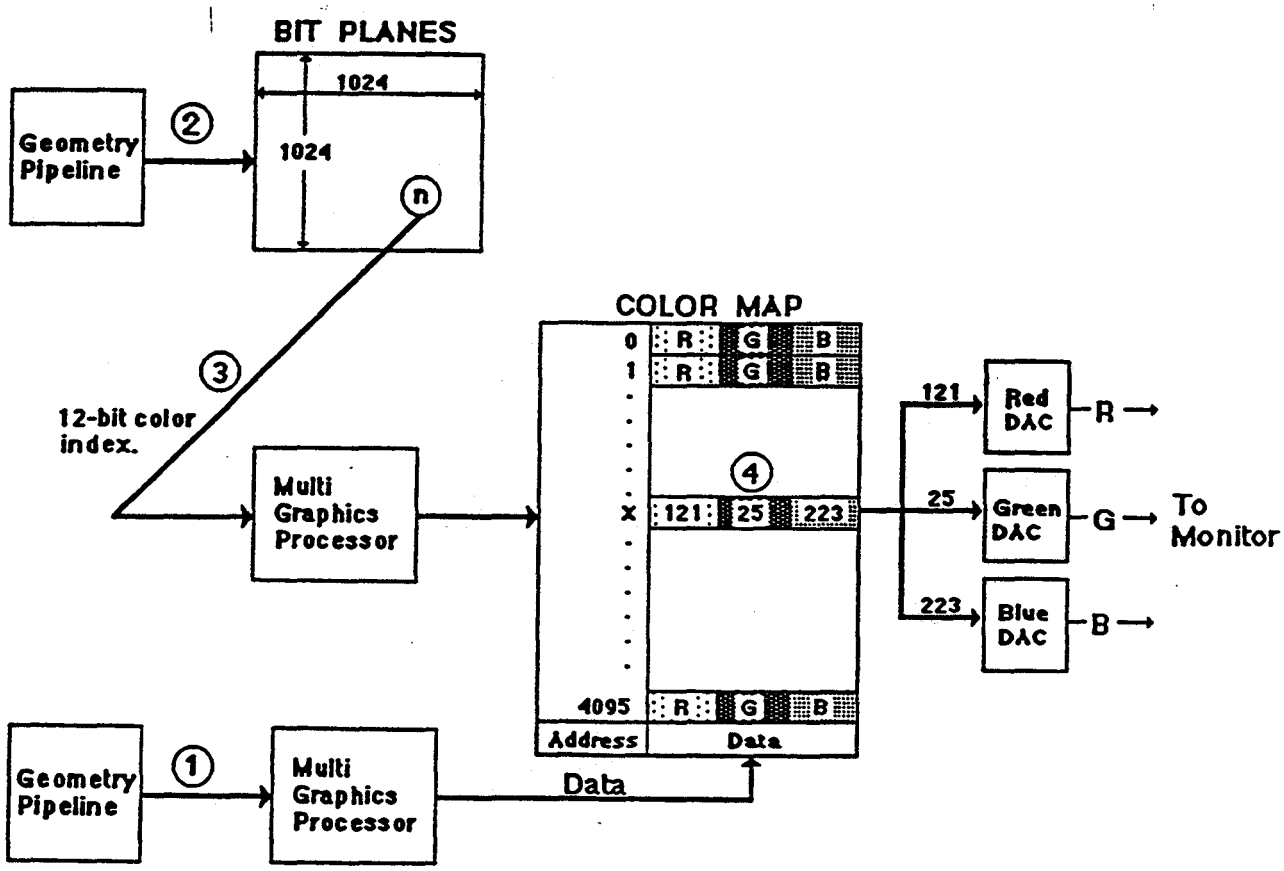
    When the programmer wants to change the color codes in the color map, the system will load (map some new colors) the color map with new color codes.

    See next page for Block Diagram of the Color Map.

The diagram below has four (4) numbers that are circled. Each number corresponds to one of the following paragraphs.

Exhibit 1. FRAME BUFFER STRUCTURE: COLOR MAP MODE

1. The color map is loaded with the digital values of the desired colors.
2. The Update Controller (UC4) loads pixel data into the Bit Planes.
3. The Display Controller (DC4) accesses the Bit Planes, at a rate that is synchronized to the monitor raster scanning, fetching pixel data sequentially - the data becoming index addresses into the color map. Point three freezes the scan at a point (a location) called n for illustration.
4. N addresses location n in the color map, from which 24 bits of color code are fetched. The three color codes are feed to the DAC's, and pixel n on the screen is painted the defined color.



BP3 PHYSICAL ORGANIZATION

The Bit Plane boards are comprised of four (4) identical bit planes. The planes are given names and the board is divided into two halves: A & B.

The names are generic and the actual configuration of the BP3 boards is illustrated by several of the diagrams that follow this diagram.

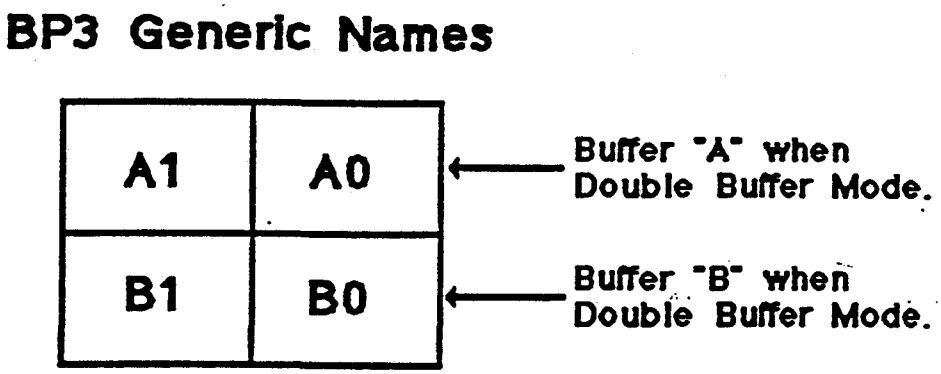


Exhibit 2. Bitplane Mnemonics

Remember, the bit planes hold an address index when in Color Map Mode. The generic address bit weights are as follows:

- Single Buffer Mode

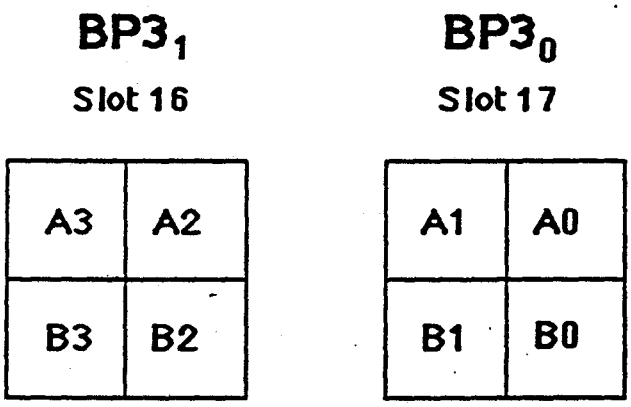
BP3 bit	B1	A1	B0	A0	
Bit weight	8	4	2	1	= 16

- Double Buffer Mode

BP3	A1/B1	A0/B0	
Bit weight	2	1	= 4

Exhibit 3. COLOR MAP MODE: Single Buffer 2 Bit Planes

2 Boards (8-bit deep pixel)



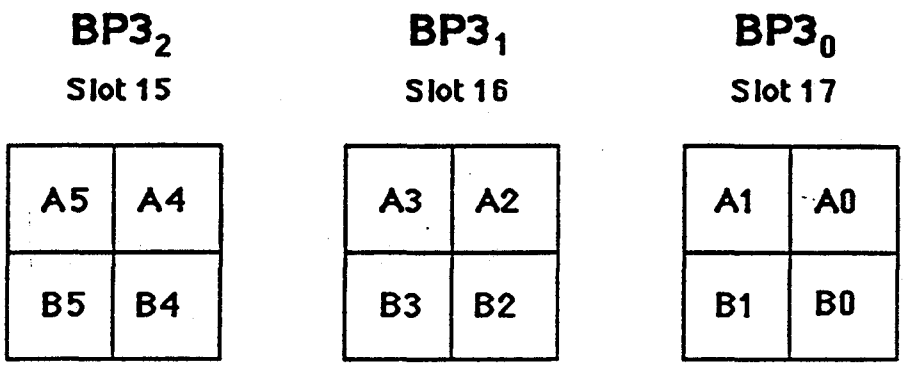
DC4 Color Map Address Bit	7	6	5	4	3	2	1	0
BP3 Bits	B3	A3	B2	A2	B1	A1	B0	A0
Color Map Address Weight	128	64	32	16	8	4	2	1

Using an 8-bit deep pixel,
256 colors could be mapped.

Address bits 8-11 are set = 0.

Exhibit 4. COLOR MAP MODE: Single Buffer 3 Bit Planes

3 Boards (12-bit deep pixel)

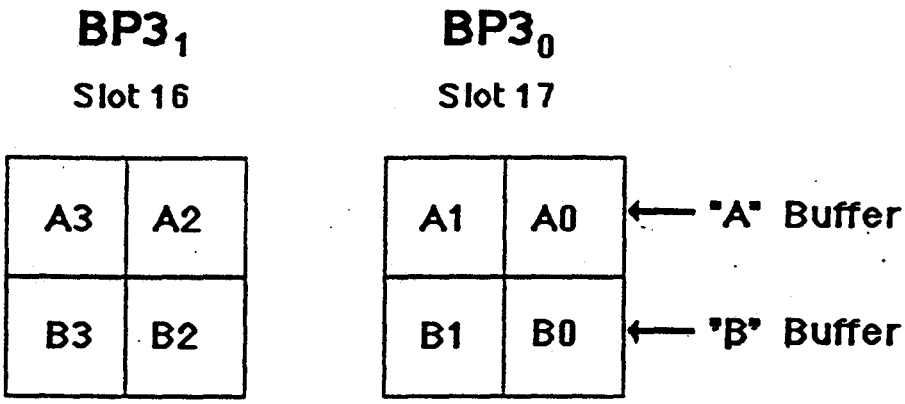


DC4 Color Map Address Bit	11	10	9	8	7	6	5	4	3	2	1	0
BP3 Bits	B5	A5	B4	A4	B3	A3	B2	A2	B1	A1	B0	A0
Color Map Address Weight	2048	1024	512	256	128	64	32	16	8	4	2	1

Using a 12-bit deep pixel, 4096 colors could be mapped.

Exhibit 5. COLOR MAP MODE: Double Buffer 2 Bit Planes

2 Boards (4-bit deep pixel)



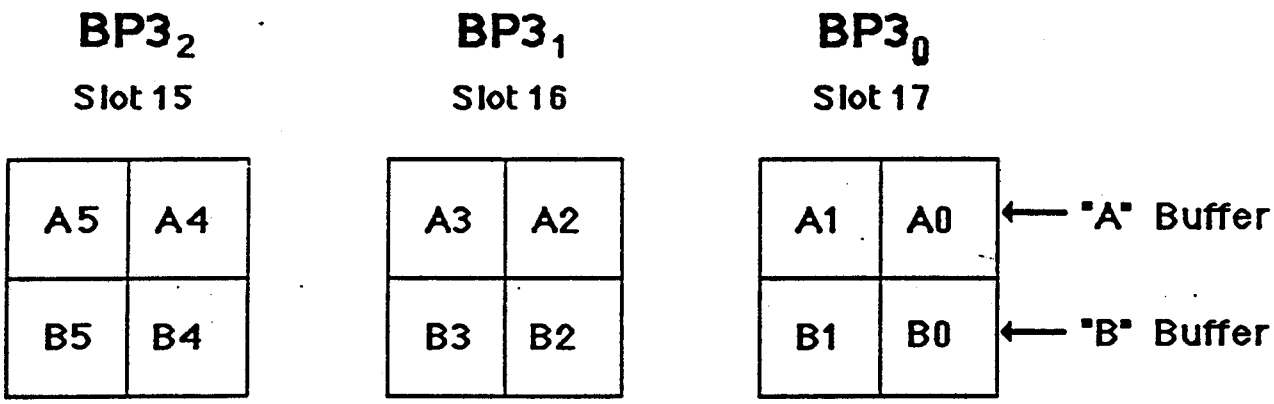
DC4 Color Map Address Bit	3	2	1	0
BP3 Bits (A Buffer)	A3	A2	A1	A0
BP3 Bits (B Buffer)	B3	B2	B1	B0
Color Map Address Weight	8	4	2	1

Using a 4-bit deep pixel in each buffer, 16 colors could be mapped.

Address bits 4-11 are set = 0.

Exhibit 6. COLOR MAP MODE: Double Buffer 3 Bit Planes

3 Boards (6-bit deep pixel)



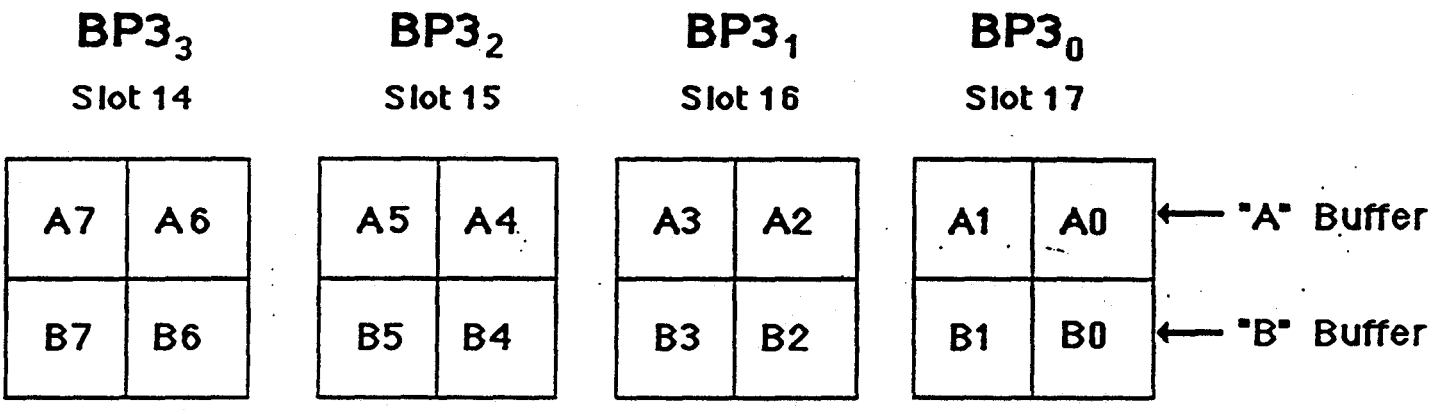
DC4 Color Map Address Bit	5	4	3	2	1	0
BP3 Bits (A Buffer)	A5	A4	A3	A2	A1	A0
BP3 Bits (B Buffer)	B5	B4	B3	B2	B1	B0
Color Map Address Weight	32	16	8	4	2	1

Using a 6-bit deep pixel in each buffer,
64 colors could be mapped.

Address bits 6-11 are set = 0

Exhibit 7. COLOR MAP MODE: Double Buffer 4 Bit Planes

4 Boards (8-bit deep pixel)



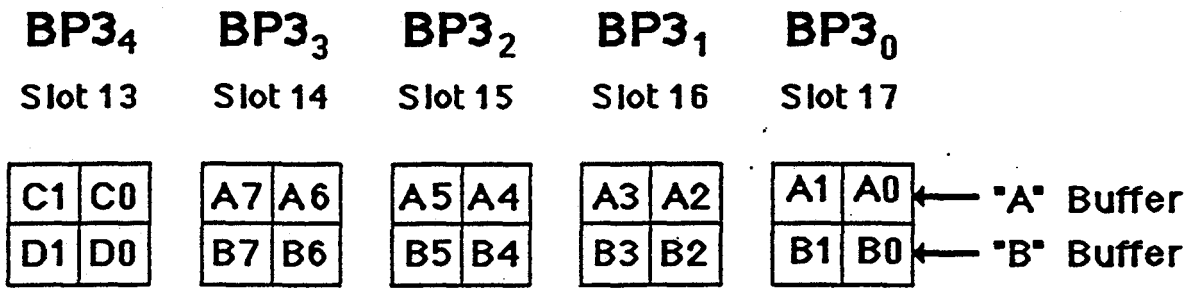
DC4 Color Map Address Bit	7	6	5	4	3	2	1	0
BP3 Bits (A Buffer)	A7	A6	A5	A4	A3	A2	A1	A0
BP3 Bits (B Buffer)	B7	B6	B5	B4	B3	B2	B1	B0
Color Map Address Weight	128	64	32	16	8	4	2	1

Using an 8-bit deep pixel in each buffer,
256 colors could be mapped.

Address bits 8-11 are set = 0

Exhibit 8. COLOR MAP MODE: Double Buffer 5 Bit Planes

5 Boards (10-bit deep pixel)



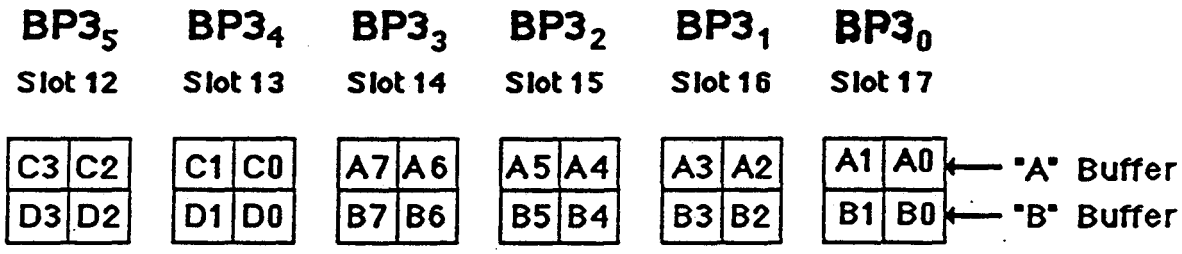
DC4 Color Map Address Bit	9	8	7	6	5	4	3	2	1	0
BP3 Bits (A Buffer)	C1	C0	A7	A6	A5	A4	A3	A2	A1	A0
BP3 Bits (B Buffer)	D1	D0	B7	B6	B5	B4	B3	B2	B1	B0
Color Map Address Weight	512	256	128	64	32	16	8	4	2	1

Using a 10-bit deep pixel in each buffer, 1024 colors could be mapped.

Address bits 10-11 are set to zeros.

Exhibit 9. COLOR MAP MODE: Double Buffer 6 Bit Planes

6 Boards (12-bit deep pixel)



DC4 Color Map Address Bit	11	10	9	8	7	6	5	4	3	2	1	0
BP3 Bits (A Buffer)	C3	C2	C1	C0	A7	A6	A5	A4	A3	A2	A1	A0
BP3 Bits (B Buffer)	D3	D2	D1	D0	B7	B6	B5	B4	B3	B2	B1	B0
Color Map Address Weight	2048	1024	512	256	128	64	32	16	8	4	2	1

Using a 12-bit deep pixel in each buffer,
4096 colors could be mapped.

Exhibit 10. RGB MODE

Must have 6 boards (24 bits)

$BP3_5$	$BP3_4$	$BP3_3$	$BP3_2$	$BP3_1$	$BP3_0$
Slot 12	Slot 13	Slot 14	Slot 15	Slot 16	Slot 17
C3 C2 / D3 D2	C1 C0 / D1 D0	A7 A6 / B7 B6	A5 A4 / B5 B4	A3 A2 / B3 B2	A1 A0 / B1 B0

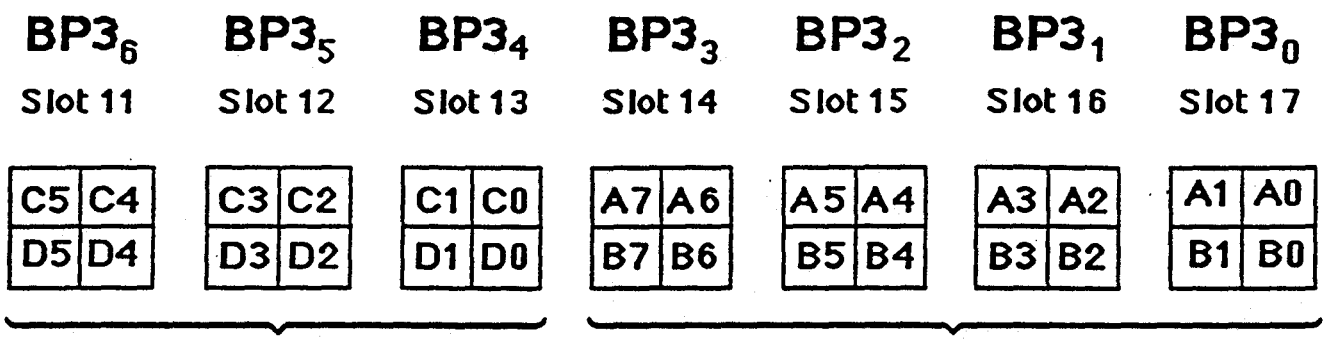
	Bit Plane bits to DAC input bits							
	7	6	5	4	3	2	1	0
Red	A7	A6	A5	A4	A3	A2	A1	A0
Green	B7	B6	B5	B4	B3	B2	B1	B0
Blue	D3	C3	D2	C2	D1	C1	D0	C0
	128	64	32	16	8	4	2	1
	Color Weight							

Red = 256 shades (intensity)
Green = 256 shades (intensity)
Blue = 256 shades (intensity)

256 x 256 = 65,536 x 256 = 16,777,216

In RGB mode, 16,777,216 colors are possible.

Exhibit 11. Z-BUFFERING: 7 Bit Planes



Z - Buffer Data
- 12 planes

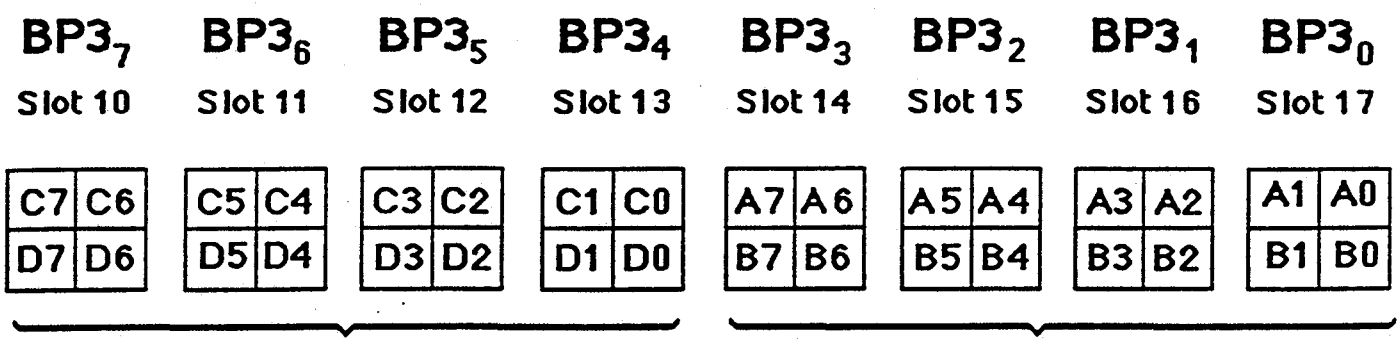
Pixel Color Data
- Single buffer mode only
- 12 bit-planes store pixel color code to produce 4096 possible mapped colors.
- 4 bit planes un-used.

- Z-Buffer data is not used to drive the DAC's to produce video. It is used for **Hidden Surface Removal.**

  Hidden surface removal is used when two or more objects occupy the same pixel. The Z coordinate for each pixel on the screen is stored in the bitplanes. When a pixel is to be drawn, it's new Z value is compared to the existing Z value. If the new Z value is less than the existing Z value (i.e., closer to the viewer), the new color value for that pixel is written into the bitplanes along with the new Z value. Otherwise, the color and Z value remain unchanged. The result is only the parts of the image visible to the viewer are displayed on the screen.

- Z-Buffering could be accomplished by programming if the customer does not purchase the optional bit planes, but it would take more time to perform the operation.

Exhibit 12. Z-BUFFERING: 8 Bit Planes



QUESTION:
Why are 4 bit planes lost?

ANSWER:
Because of the method used to write data into the bit planes. The bit planes are accessed on a word boundary - a word is comprised of 16 bits.

Bit Plane Write Access

Single Buffer Mode:	Write to A & B.
Double Buffer Mode:	Write to A & C or B & D.
Z-Buffer Mode:	Write to A & B for color.
	C & D for depth (Z-coordinate).

Since 12 bits is the maximum color pixel depth, 4 bit planes in the A & B longword are unused.

This diagram and the next illustrate how the Bit Planes are cabled to the Display Controller (DC4).

Exhibit 13. BIT PLANE to DC4 CABLE CONNECTIONS

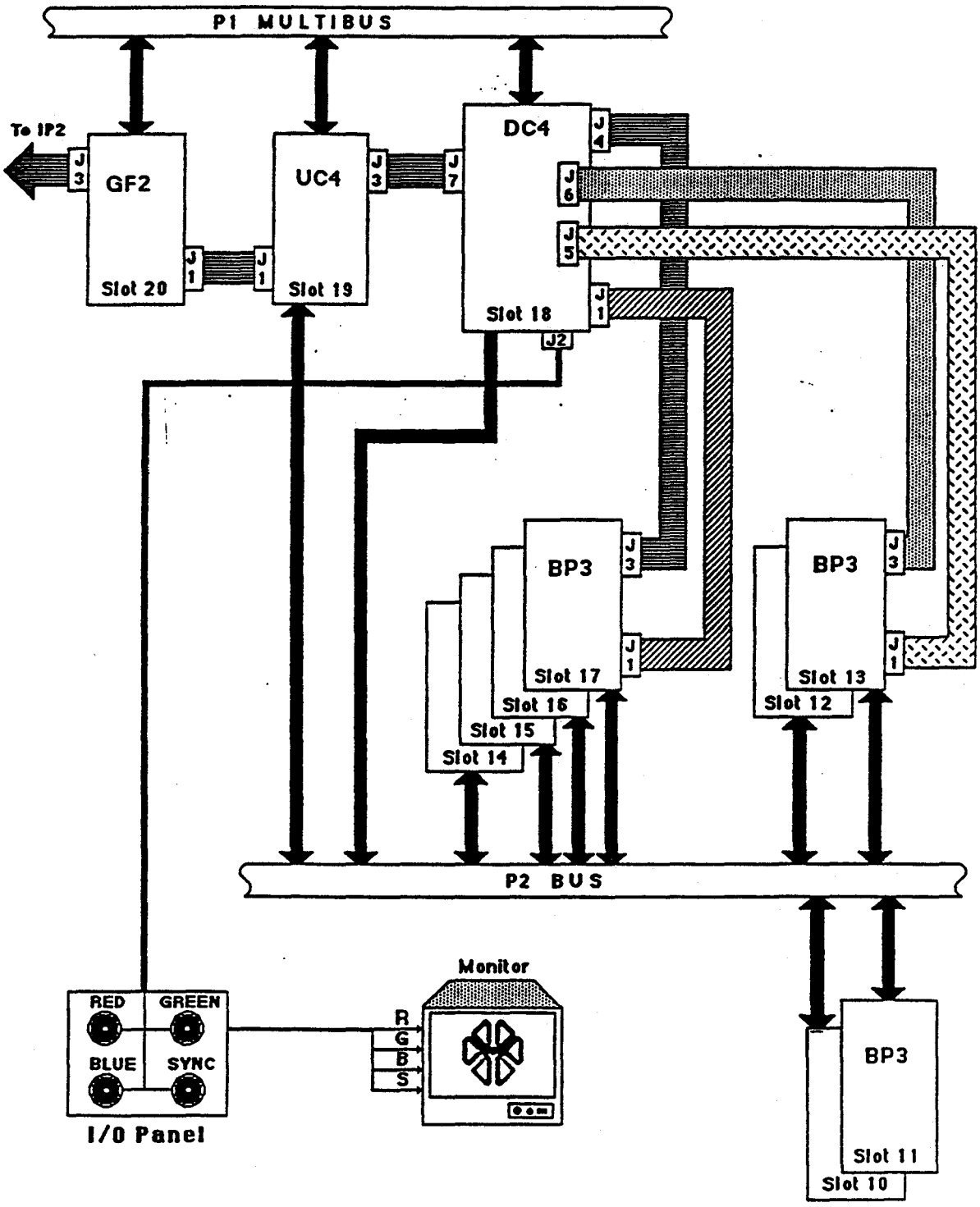


Exhibit 14. BIT PLANE MAPPING and CABLE CONNECTION TABLES

Mapping System Bitplane Names to BP3's							
Bitplane		Bitplane		Bitplane Names for each BP3 Slot Number of BP3's			
number	name	number	name	17/13	16/12	15/11	14/10
0	A0	10	C0	A0			
1	A1	11	C1	A1			
2	A2	12	C2		A0		
3	A3	13	C3		A1		
4	A4	14	C4			A0	
5	A5	15	C5			A1	
6	A6	16	C6				A0
7	A7	17	C7				A1
8	B0	18	D0	B0			
9	B1	19	D1	B1			
A	B2	1A	D2		B0		
B	B3	1B	D3		B1		
C	B4	1C	D4			B0	
D	B5	1D	D5			B1	
E	B6	1E	D6				B0
F	B7	1F	D7				B1

BP3 to DC4 Cable Connections					
Bd #	Bitplanes	BP3	DC4	BP3	DC4 Non-Int
0	A0,A1,B0,B1	J1	J1	J3	J4
1	A2,A3,B2,B3	J1	J1	J3	J4
2	A4,A5,B4,B5	J1	J1	J3	J4
3	A6,A7,B6,B7	J1	J1	J3	J4
0	C0,C1,D0,D1	J1	J5	J3	J6
1	C2,C3,D2,D3	J1	J5	J3	J6
2	C4,C5,D4,D5	-	-	-	-
3	C6,C7,D6,D7	-	-	-	-

Note: In non-interlaced mode, use both DC4 and DC4 Non-Int cable connections.

Exhibit 15. The Geometry Subsystem

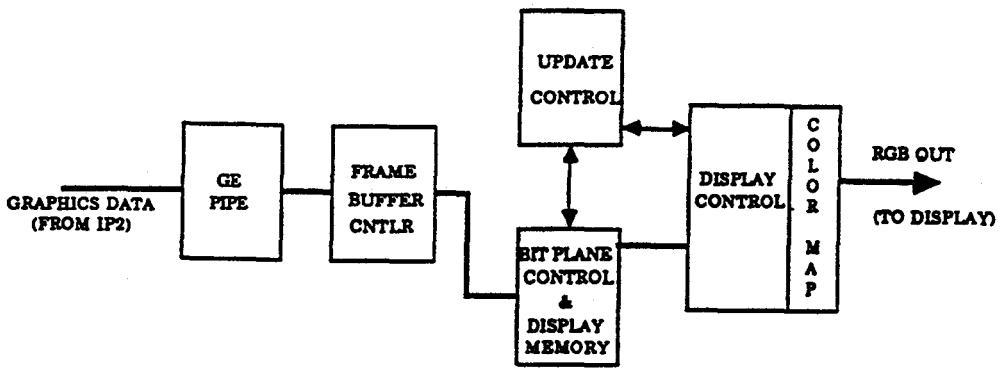
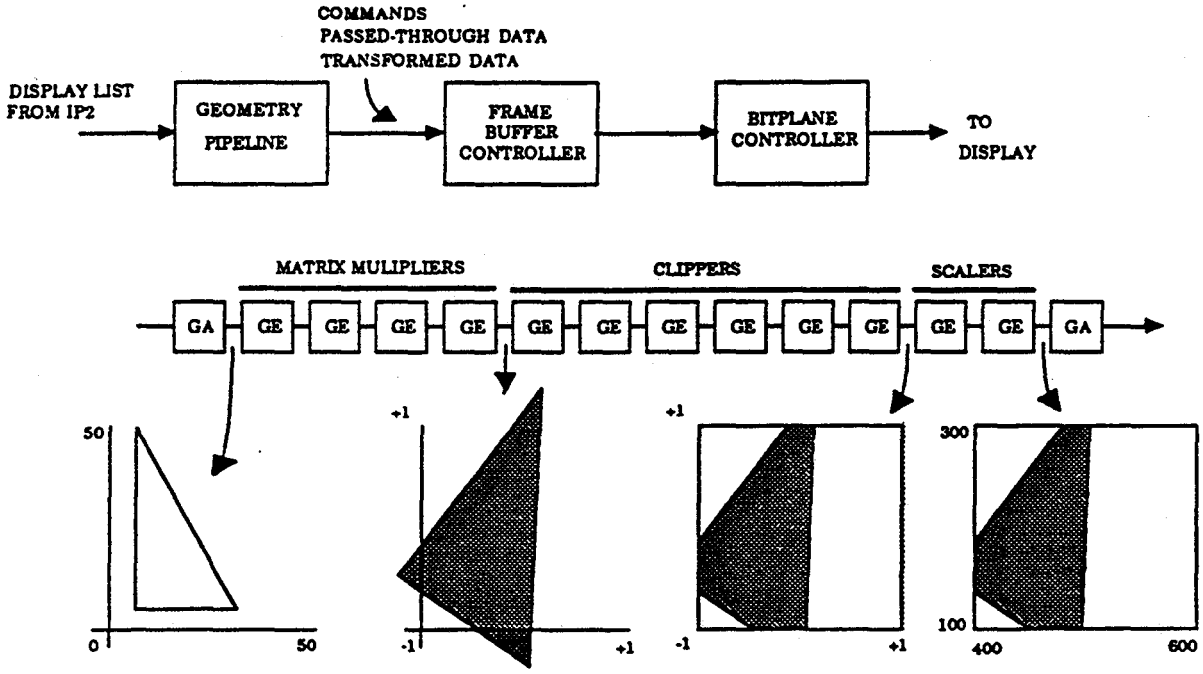


Exhibit 16. The Geometry Pipeline Operation



The Geometry System

The Geometry System comprises the core of the IRIS graphics subsystem. It consists of a *pipeline* of Geometry Engines (GEs) designed specifically for high-performance floating-point geometric computation. The pipeline consists of three distinct subsystems. Although each Geometry Engine is identical, it is its' physical position in the pipeline that determines its' function.

The three subsystems are:

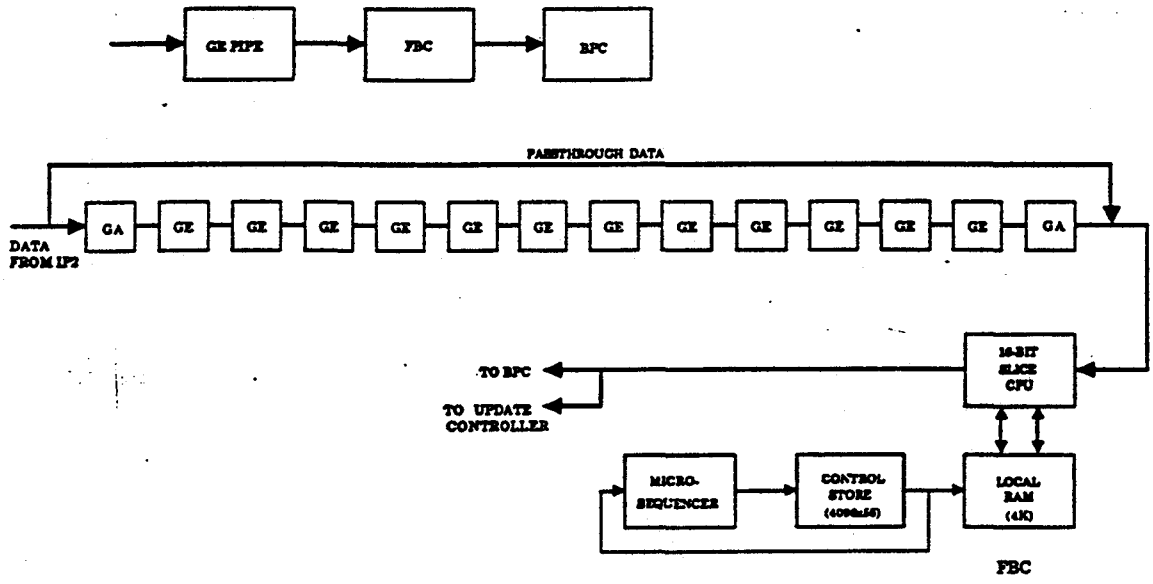
- Matrix Subsystem
- Clipping Subsystem
- Scaling Subsystem

The first four GEs comprise the **Matrix Subsystem.** Their purpose is to perform the matrix multiplication to transformation of graphical data.

The following six GEs comprise the **Clipping Subsystem.** Each is responsible for enforcing the boundaries for a single plane in the viewing volume. The diagram below shows a typical viewing volume and its' associated planes. Note that GE 5 and 10 are optional since they perform Z-clipping (near and far plane), an option on the IRIS 3000-Series workstations.

The last two GEs comprise the **Scaling Subsystem.** They convert the output of the Clipping Subsystem GEs to the appropriate coordinate system of the display or output device.

Exhibit 17. GF2 Block Diagram



The GF2 Board

The GF2 board consists of two major areas of real estate. They are the:

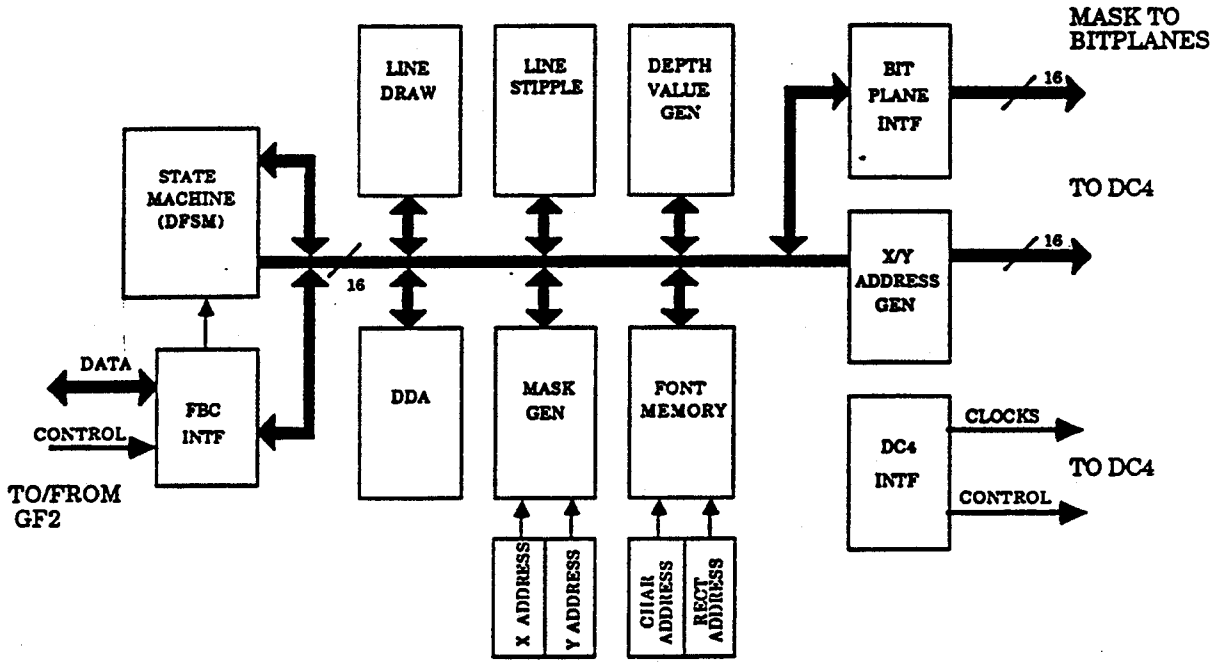
- Geometry Pipeline
- Frame Buffer Controller (FBC)

The **Geometry Pipeline**, in addition to the 14 (12 w/o the Z-clipping option) contains two additional GE chips called Geometry Accelerators (GAs). These are positioned at the head and tail of the Pipeline. The head GA converts incoming floating point data from an IEEE format to a special format that the GEs can utilize. The tail GA performs the reconversion back to the IEEE format. Each GA has a FIFO buffer to minimize delay of data through the pipeline and uses a high and low-watermark interrupt scheme to control data flow.

Both the GAs and GE chips are identical. However, their physical position in relationship to one another, determines their specific function. Within each chip is a configuration register that is loaded during initialization of the Geometry Subsystem to set the proper mode for each chip.

The **Frame Buffer Controller** is a specialized processor which uses four AMD2903 bit-slice CPUs to form a 16-bit processor. Operation of the processor is controlled by µcode contained in the Control Store RAM. A µcode sequencer controls addressing and sequence of execution of the µcode routines on behalf of the processor.

Exhibit 18. UC4 Block Diagram

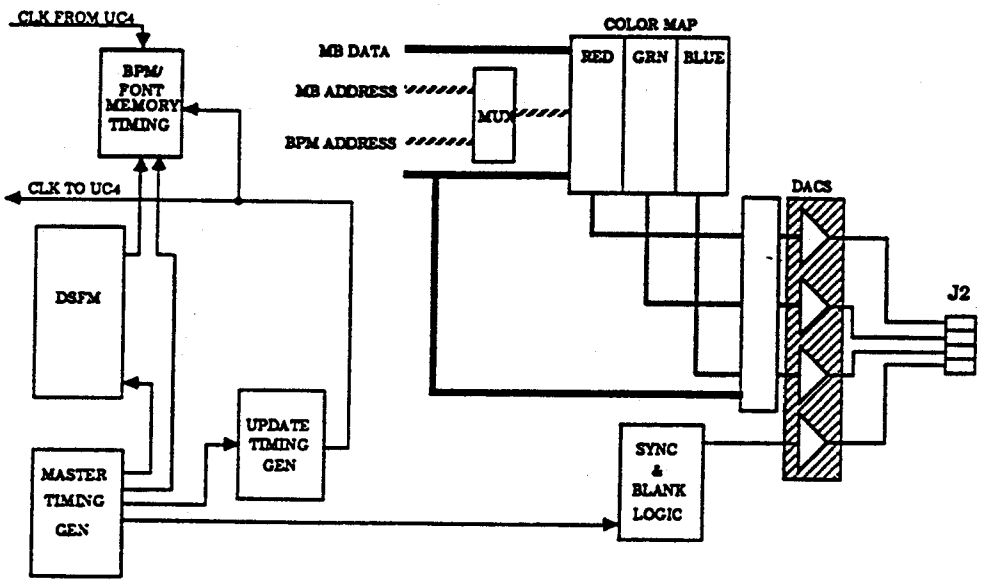


The Update Controller Board

The primary function of the Update Controller (UC4) is to draw lines, characters, and rectangles to the Bit Planes. Some other functions performed by the UC4 are:

- providing an interface between the *Raster Subsystem* (also referred to as the BPC) and the geometry pipeline (in the FBC).
- Confining Bit Plane memory modifications to a screen-aligned area (called the BPC viewport. Points, lines, characters, and filled rectangles are masked, rather than clipped, to these softly defined boundaries.
- Maintaining a board-local memory space for storage of character and rectangle masks, colorcodes and write enable codes for the Bit Planes.
- Maintains a 16-bit value, called a *line stipple pattern* that is used during line drawing.
- Generates Bit Plane memory refresh addresses.

Exhibit 19. DC4 Block Diagram



The Display Controller

The **Display Controller (DC4)** comprises the raster or bitplane control subset of the graphics subsystem. The raster subsystem manages the 24-bitx1024x1024 raster memory through the following three processes:

- Update
- Display
- Refresh

The **Update** process includes any operation that involves moving data to or from the bitplane memory. These operations include drawing of points, lines, characters, rectangle patterns, and filled trapezoids.

The **Display** process refers to that of *orderly* retrieval of data from bitplane memory to the display device.

The **Refresh** process involves the periodic access of all bitplane memory locations to prevent data loss within the DRAMs.

The DC4 generates the following:

- Display addresses for monitor screen refresh and control for bitplane memory addressing.
- Synchronization and blanking signals used by the color monitor.
- Clocking controls for display updating.
- Maps output of the bitplanes to three 8-bit colors or provides color mapping.
- Converts three 8-bit color codes to three analog color signals (RGB).

SECTION 3: Test Items

1. Which graphics subsystem comonent is responsible for determining how lines will be drawn?

_____

2. On which PC board is the Color Map memory located? address switches for the bit plane in card slot 13.

_____

3. On which PC board is bitplane B1 located?

_____

4. Final scan conversion is performed by which graphics function?

_____

Contents

ARCH10 1-1
- Introduction 1
- Objectives 1
- Criterion Test 1
- Resources 2
- SECTION 1: CPU Subsystem Architecture 4
  - IP2 Features 5
  - IP2 Board 5
  - Memory Organization 6
  - Memory Address Map 6
  - Local Memory 8
  - Mouse Port 9
  - Graphics Pipeline Port 10
  - Bus Errors 10
- SECTION 2: I/O Subsystem Architecture 12
  - Disk/Tape Controllers 12
  - Interphase Storager II Disk/Tape Controller 12
  - DSD 5217 Disk/Tape/Floppy Controller 23
  - Interphase SMD 2190 Disk Controller 29
  - Ciprico Tapemaster 1000 Tape Controller 35
  - Tape Controllers 42
  - Ethernet Controllers 42
- SECTION 3: Test Items 48

ARCH10

Introduction

In this module you will get a better idea of the how the CPU and I/O subsystems work. A description of each board, at the major functional block level, is given. Also, diagrams showing the location of configuration switches and jumpers are included.

Objectives

At the completion of this lesson, you will be able to:

- Describe the physical organization of the CPU and I/O subsystems.
- Identify the major functional blocks of the CPU and I/O subsystems.
- Match the major functional blocks of the CPU and I/O subsystems with their correct descriptions.
- Locate and verify the correct setting of each configuration switch/jumper on the IP2, FP1, and IM1 boards.
- Match the disk drive type to its' correct controller board.
- Locate and verify the correct setting of each configuration switch/jumper on the disk and tape controller boards.
- Match controller error indications with their correct description.

Criterion Test

Complete a block diagram by adding the missing block identifications and adding any missing interblock connections. You may use any available documentation.

Resources

This guide

This module contains three sections:

- **SECTION 1:**

  This section is contains the instructional procedures and explanations describing the architecture of the CPU subsystem.

- **SECTION 2:**

  This section is contains the instructional procedures and explanations describing the architecture of the I/O subsystem.

- **SECTION 3:**

  This section contains review questions that you should complete after studying the material in this module.

SECTION 1: CPU Subsystem Architecture

The CPU subsystem is comprised of the following components:

- IP2 board
- FP2 board
- IM1 boards (up to four)

The IP2 board is a standard 12" by 12" multibus board including, as the system main CPU, a Motorola M68010 running at 16Mhz.

> 2000-series workstations used the IP1 CPU board running a M68010 CPU chip at 12.5Mhz. The "original" SGI board called the "SUN" (not to be confused by the company of the same name) was designed at Stanford University. The SUN(aka PM1) board used a 68000 CPU. SGI licensed the SUN board, modified it to interface with the graphics and mouse ports, and added some PROM space.
>
> The first SGI-designed CPU board, used in many of the 1000-series workstations and also 68000-based, was (and most certainly still is) called the PM2 board. Also a 68000-based board, it permitted multibus access to main memory unlike the PM1 board.
>
> The PM2 board was replaced by the IP1 CPU for reliability and is software compatible with the older processor.

The FP1 board provides the IRIS with a dedicated coprocessor specifically designed to handle all floating-point calculations required to render 3D graphics data. The FPA (Floating Point Accelerator), as it is sometimes called, uses a Weitek Arithmetic Logic Unit and Multiplier Unit which give the IRIS the capability of performing high-speed floating point calculations.

The IRIS 3000-Series workstations can be configured with a maximum of 16Mbytes of main memory using a combination of IM1 boards. Each board is configured with 4Mbytes of DRAM.

IP2 Features

The IP2 board provides the following features and functions:

- Four RS232C serial ports for terminals and printers.
- a private port to the graphics pipeline.
- 64 Mbytes of virtual address space available for each process.
- Support for up to 16 Mbytes of physical memory.
- Fault detection circuits.
- Real-time clock with non-volatile memory and battery backup.
- A 32-bit dedicated interface to an IEEE floating-point processor board.

Features not supported on the IP2 that were on older models are:

- Centronics parallel I/O port found on PM2 boards.
- DMA processor. On the IP2, the CPU controls data transfers to/from memory.
- Memory caching. The IP2 does not utilize a cache memory.

IP2 Board

The system CPU (IP2 board) contains the following functional components:

- M68020 32-bit microprocessor
- main memory address map
- local static RAM memory
- local PROM
- status and configuration registers
- real-time clock
- mouse/keyboard port

- RS232C DUARTS(2)
- Graphics port
- Floating Point Accelerator port
- multibus interface

Memory Organization

The M68020's virtual memory address space occupies 4 Gigabytes. Simply translated, its' 32-bit address bus can address any one of the 4 Gigabytes of memory locations. In the IRIS, this memory space is grouped into 16 groups called segments. Each segment occupies 256 Megabytes of address space. Each address space has a specific use. The following diagram shows the organization. Note that some segments are not used.

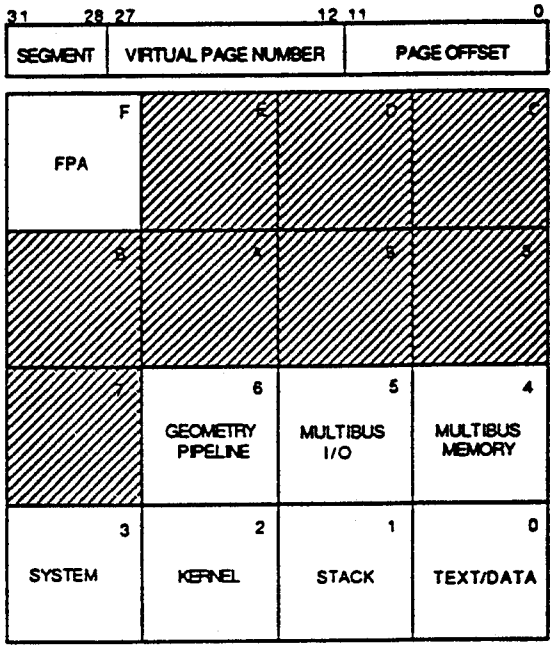


Exhibit 1. Virtual Memory Segmentation

Memory Address Map

The purpose of the Memory Address map is to translate *virtual* memory addresses to *physical* memory addresses. Virtual addresses are those output by the CPU. Since the M68020 has an addressing space that is 4 Gigabytes wide (32-bit address lines) and the

IRIS can only accommodate up to 16 Megabytes of physical memory, association of a virtual address to a physical one must occur for each memory access.

The memory map is nothing more than a local memory containing a list of pointers to physical locations in physical memory. These locations are referred to as *pages*. Each page is 16Kb long. Thus, each entry in the memory map points to the beginning address of a page of physical memory.

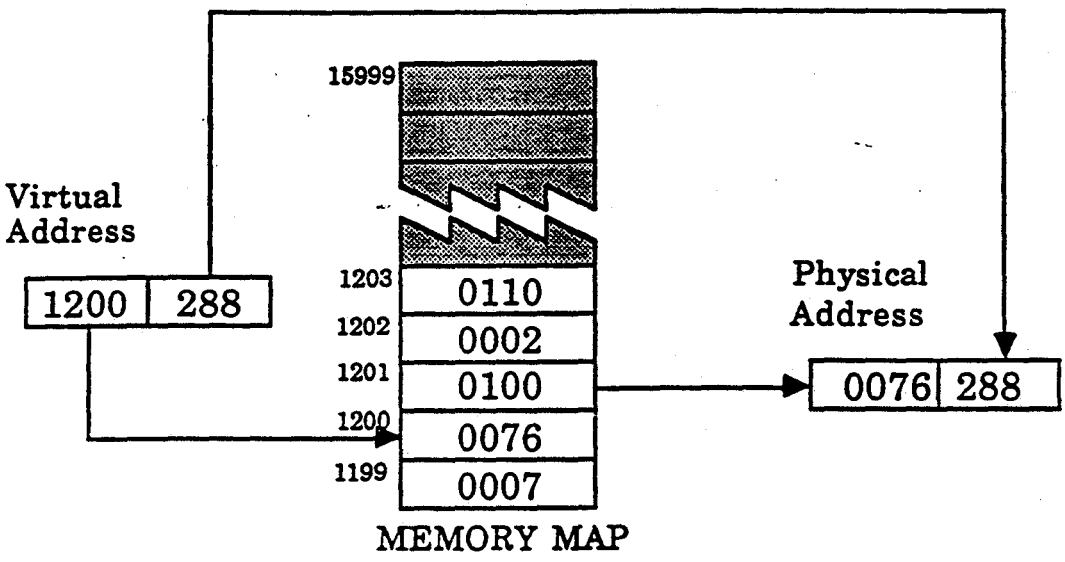


Exhibit 2. Memory Map Data Flow

The address translation process is one of using a number of virtual address bits output by the CPU as an index to a single entry in the map memory. That entry contains an address to the a physical location for the respective page. The location of the data in the page, called the *offset*, is provided by concatenating the low-order virtual address to the translated page address. A simplified diagram of the process is shown below:

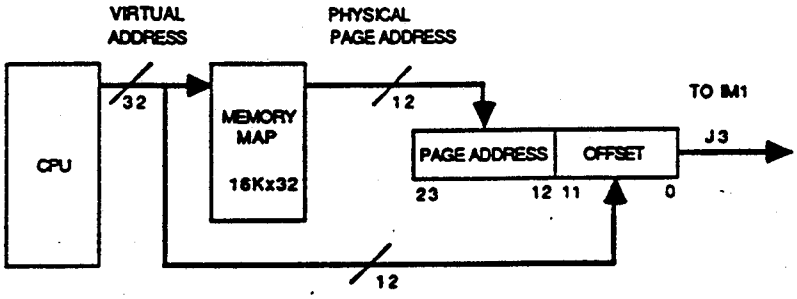


Exhibit 3. Virtual to Physical Address Translation

Memory mapping of virtual addresses to physical ones allows programs to utilize the full address space that the CPU architecture provides regardless of physical main memory size.

Local Memory

The IP2 board contains 16 kilobytes of static RAM for use by the CPU and PROM Monitor routines. The PROM Monitor contains the IRIX bootstrap program and memory access routines.

Mouse Port

Mouse movement is constantly monitored by comparing mouse control signals with a register value which represents the current state of the mouse control signals.

Any time a mouse movement is detected, a CPU interrupt is asserted. The register value is compared with the current value of the mouse control signals to determine the direction of movement. The interrupt can then be reset and the new value stored so that the change may be detected.

The mouse generates four signals:

X Fire	indicating that the x-axis position has changed.
X Direction	indicating that the movement was to the right.
Y Fire	indicating that the y-axis position has changed.
Y Direction	indicating that the movement was upward.

The following diagram shows the how the different combinations can detect movement in any direction.

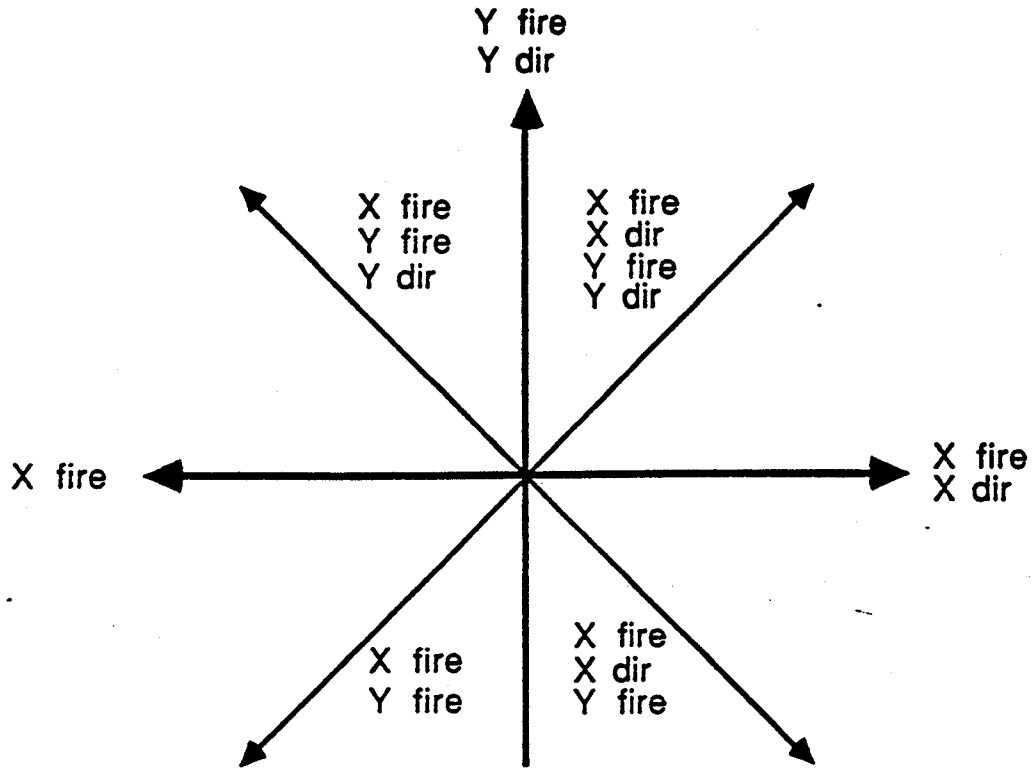


Exhibit 4. Mouse Movement Detection

Graphics Pipeline Port

The IP2 is directly connected to the input of the GF2 boards through the graphics port connector J5. The graphics port interface provides a fast access path for graphics data to the pipeline in 16-bit chunks. The state of the graphic pipeline input is constantly monitored. As long as data is being accepted it can be transferred with no wait states.

Bus Errors

The M68020 uses an input signal called *BERR* (bus error) which, when asserted causes the CPU to:

- abort the current CPU cycle and undo any internal side effects of the cycle.
- save the current state of the cycle (e.g., instruction counter, status registers, etc.) so that the cycle may be retried later.
- enter system (supervisor) state to attempt to clean up the condition(s) that caused the bus error.

The following conditions cause a bus error:

- an access attempt to an invalid memory segment.
- an access by a user process to a memory segment that it does not have permission to access.
- an access by a user process of a multibus memory segment that it does not have permission to access.
- a memory timeout. This occurs when a memory access is requested and no acknowledgement is received within a given period.
- access to a protected memory page.
- when the FPA asserts its' bus error output signal.

The bus error is considered a *hard* system crash and will usually cause the CPU to enter the *halt* state.

SECTION 2: I/O Subsystem Architecture

The I/O subsystem consists of:

- Disk/tape controller
- Ethernet node controller
- RS232C expansion controller

Disk/Tape Controllers

There are three disk controllers used in the 3000-series workstations:

CONTROLLER	DISK	TAPE	INTERFACE	PART#
Interphase Storager II	170Mb	Cipher 1/4"Cartridge	ESDI	940008
DSD 5217	72/85Mb	Cipher 1/4" Cartridge	ST506	940001
Interphase 2190	474Mb	none	SMD	940004

Interphase Storager II Disk/Tape Controller

Introduction

The Interphase Storager II Disk/Tape Controller is an intelligent multifunction controller for the Multibus and is based on a 68000 µprocessor. It can support up to two Winchester disk drives and four QIC02 or Archive interfaces as well as up to two 3 1/2, 5 1/4, or 8-inch floppy disk drives.

General Architecture

The General architecture of the controller is shown in Exhibit 5. The on-board µprocessor controls all disk and tape operations and makes it possible to relieve the workstation's CPU from having to monitor I/O operations and data transfers. These I/O operations tend to be *really* slow in relation to the speed at which the workstation's CPU operates. And remember that the main goal of the workstation CPU is to execute system and user programs. Having to deal with I/O operations would severely impact throughput.

To permit simultaneous disk and tape operations, two independent state machines are utilized. These state machines perform device-level command sequences for read and write operations between the storage media and Virtual Buffers.

Virtual buffers are employed on the board to hold read/write data. A pool of buffers is

available to be allocated to a device on demand and released when not needed anymore. Enough buffers are assigned to the pool so that it appears to each attached device that it has a seeming endless number of buffers. This is referred to as virtual buffering. These buffers are contained in the on-board memory.

The Storager II also utilizes caching of data to increase access to disk or tape data. The cache consists of high-speed RAM. When a requested sector or block of data is read into a buffer, subsequent sectors or blocks will also be read until its available buffers are filled. Thus, if subsequent requests from the operating system are for sectors logically contiguous with those previously requested, then since they are already in the buffer, no physical disk read is required.

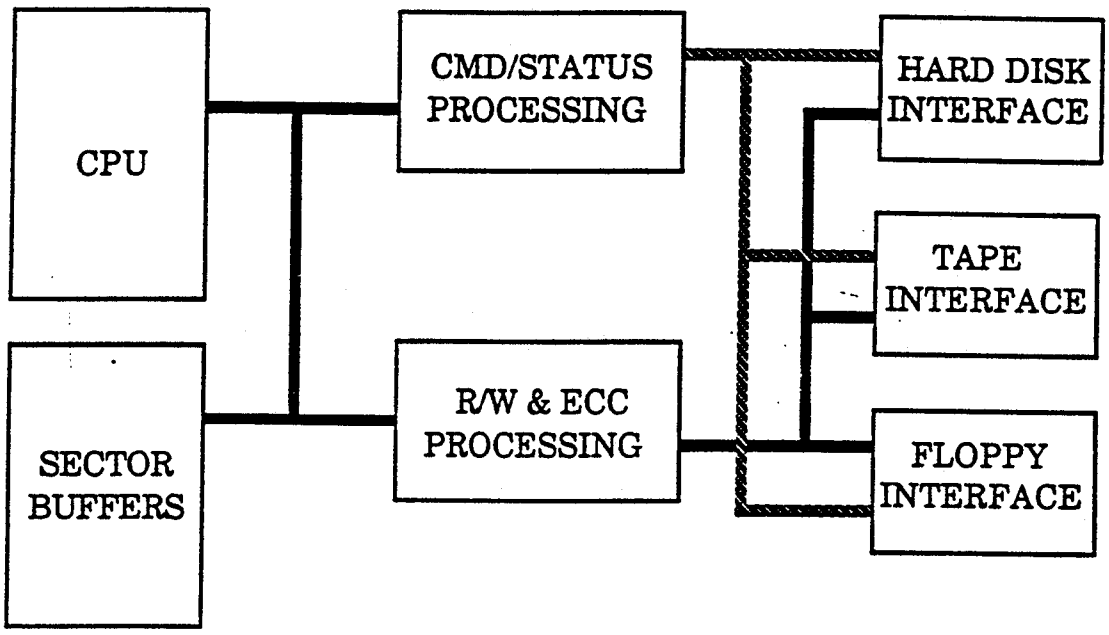


Exhibit 5. Storager II Block Diagram

Controller Interconnect

Exhibit 6 shows the physical interconnections between the controller and disk/tape drives.

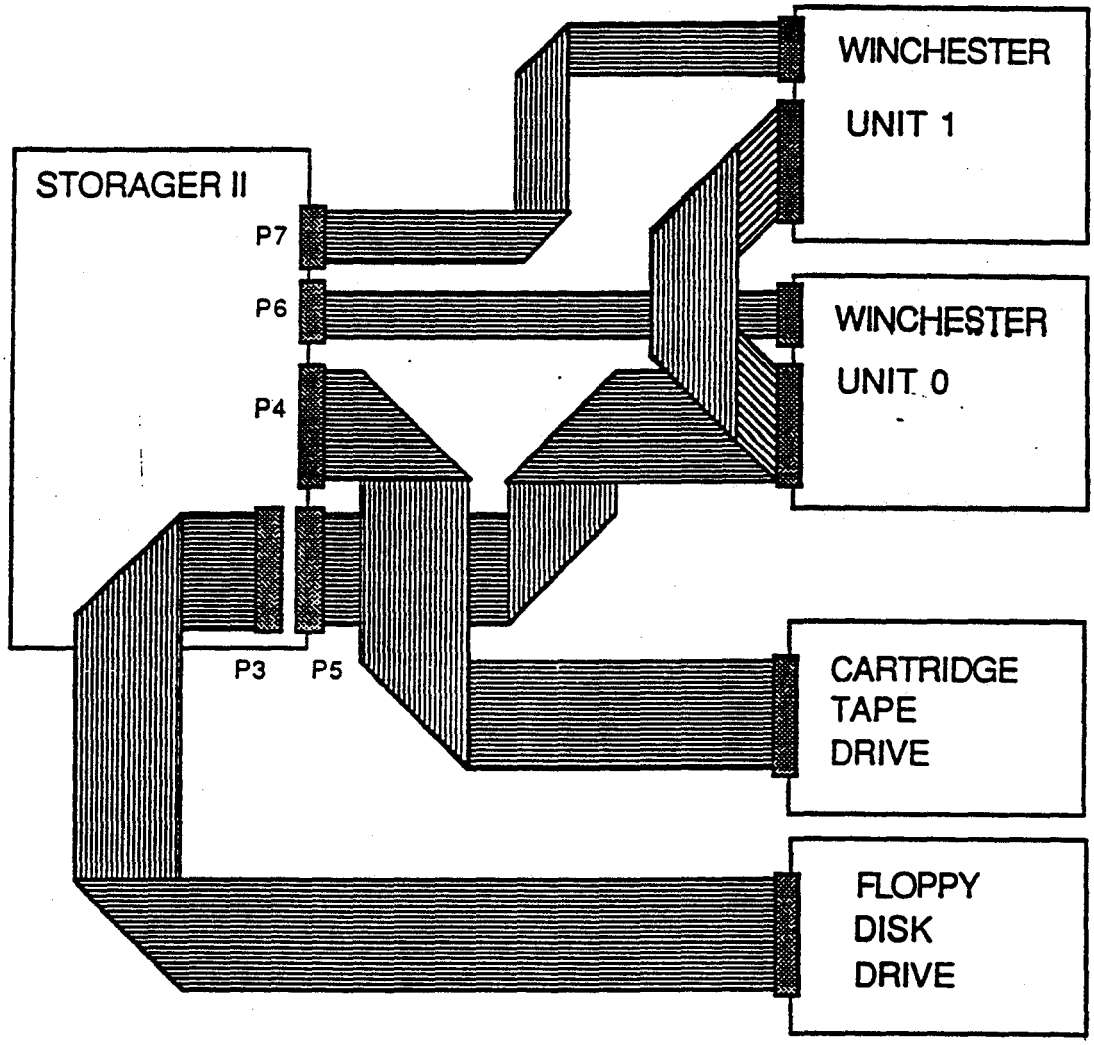


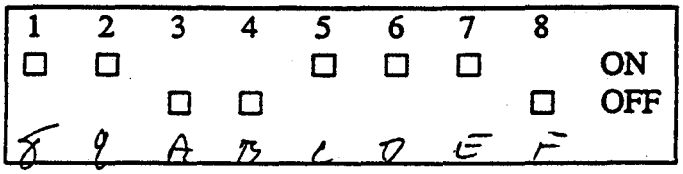
Exhibit 6. Controller and Disk/Tape Interconnection

Configuration

Hardware controller configuration is performed at the factory and may not necessarily be required before installation in a workstation. *However, always check the board for proper strapping.*

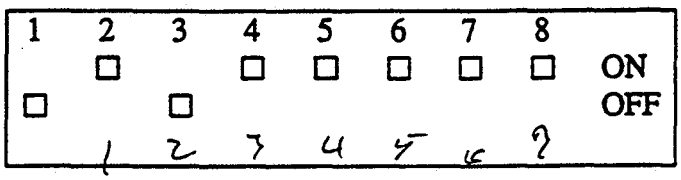
The following jumpers and switchpaks must be set prior to installation into a workstation. Refer to Exhibit for correct settings.

Storager Option Straps		
Strap	Default	Description
E20-E21-E22	E20-E21	Multibus BPRN mode select
E18-E19	Installed	Multibus BPRO mode select
E4-E5	Installed	Data separator clock speed select
E42-E43	Removed	Four or eight I/O register select
JP12-J13 JP14-JP15	Installed Installed	This jumper field is used for mixing ESDI and ST506 type drives
JP1-JP2-JP3 JP6-JP5-JP4 E16-E17 JP9-JP10-JP11	JP2-JP3 JP4-JP5 Installed JP9-JP10	Queue mode select; 8/16 bit I/O addressing. 1
E24 E26 E28 E30 E32 E34 E36 E38	Installed	Multibus interrupt priority level 7 Multibus interrupt priority level 6 Multibus interrupt priority level 5 Multibus interrupt priority level 4 Multibus interrupt priority level 3 Multibus interrupt priority level 2 Multibus interrupt priority level 1 Multibus interrupt priority level 0



S1 Rocker Functions	
Switch Number	Function
1	Address bit 8
2	Address bit 9
3	Address bit a
4	Address bit b
5	Address bit c
6	Address bit d
7	Address bit e
8	Address bit f

Switch S1 Rocker Assignments and Settings



S2 Rocker Functions	
Switch Number	Function
1	Reserved
2	Address bit 1
3	Address bit 2
4	Address bit 3
5	Address bit 4
6	Address bit 5
7	Address bit 6
8	Address bit 7

Switch S2 Rocker Assignments and Settings

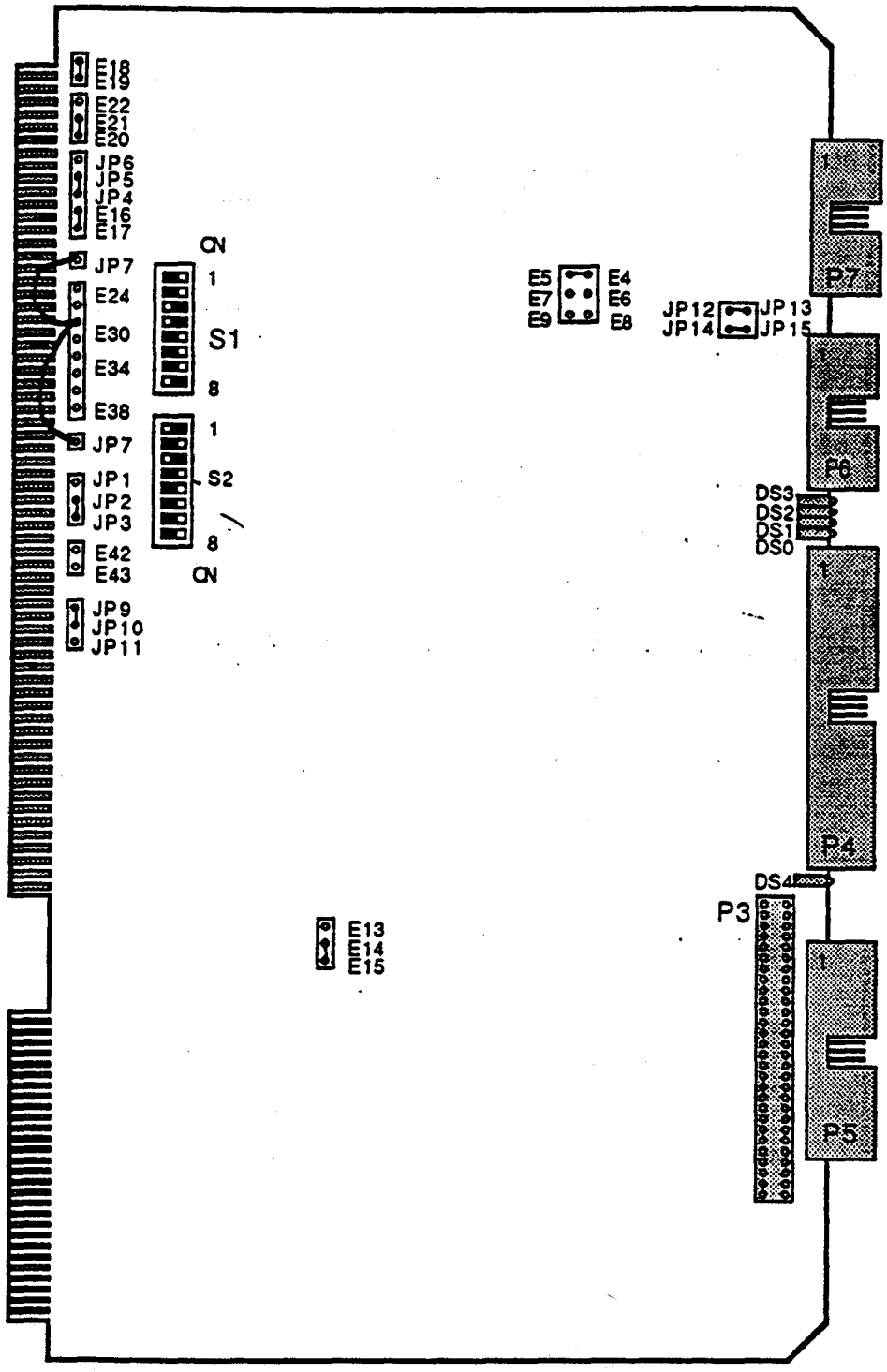


Exhibit 7. Storager II Switch/Jumper Locations

Diagnostics

When an error is detected during execution of the power-on diagnostics, or during operation on any command, the LED block DS3 (consisting of LEDs 0 through 3) is used to display the error code. The same error code that is place in the IOPB (command block in memory) is displayed in a sequence and repeated until another command is issued. Decoding the LED pattern may be necessary when the error code is not displayed with the error message. The error code is displayed as follows:

- F (all bits on)
- Error code (low-order digit)
- Error code (high-order digit)
- F (all bits on)
- Error code (low-order digit)
- Error code (high-order digit)
- Repeat

For example, the sequence *F 1 8 F 1 8 F....* can be interpreted as error code 81. Each digit is displayed for approximately 1½ seconds.

Error Codes

The Storager reports an error during a transaction by placing an error code into its command block (IOPB) in memory. The error codes are grouped as follows:

10-7f Controller or disk drive errors.

80-9f Tape drive errors.

00-0f Soft errors. Exception conditions which circumvented a transaction while in progress.

CODE	DESCRIPTION
01	End of tape encountered during disk to tape or tape to disk copy.
02	Filemark detected.
10	Disk not ready.
11	Invalid disk unit address detected (Address > 3).
12	Seek error.
13	ECC error in the data field persisted even after read retry.
14	Invalid command code.
15	Invalid cylinder address.
16	Invalid sector number.
18	Bus timeout error caused by an attempt to access nonexistent multibus memory space.
1a	Disk write protected.
1b	Disk not selected.
1c	No address mark detected in header field.
1d	No address mark detected in data field.
1e	Drive fault detected.
20	Disk surface overrun. Attempted to seek to a track greater than maximum specified for the drive.
22	Unable to reader header field.
23	Uncorrectable data error.
27	Format timeout. Usually a problem with the drive's R/W Logic.
28	No Index pulse detected. See probable cause above.
29	Sector not found.
2A	ID field error-wrong head.
2D	Seek timeout. Unable to complete seek within allowed time period.
30	Restore timeout. Unable to complete a reseek to track 0 within allowed time period.
40	Unit not initialized.
42	Gap specification error. Sync gap read problem.
50	Sectors/track specification error.
51	Bytes/sector specification error.
52	Interleave factor specification error.
53	Invalid head number specified in IOPB.

CODE	DESCRIPTION
60	Protection timeout. Internal controller problem.
61	Max Cylinder number specification error.
62	Number of heads specification error.
63	Step pulse specification error.
64	Reserved byte specification error.
65	Controller RAM failure.
66	Controller RAM failure.
67	Event RAM failure.
69	Invalid controller state.
6a	Invalid sector number.
6b	Timer failure. Controller malfunction.
6c	ROM failure.
6d	ROM failure.
80	Tape drive not selected.
81	Tape drive not ready.
82	Tape drive not online.
83	Cartridge not in place.
84	Unexpected BOT.
85	Unexpected EOT.
87	Unexpected Filemark encountered.
88	Block in error not located.
89	No data detected. Blank tape or tape positioned past data.
8a	Tape cartridge write protected.
8b	Illegal command.
8c	Command sequence timeout. Drive not responding to handshake.
8d	Status sequence timeout. See above.
8e	Data block transfer timeout.
8f	Filemark search timeout. Unable to find filemark.
90	Unexpected exception.
91	Invalid tape drive address.
92	Ready timeout.
93	Tape timeout specification error.
94	Invalid block count.

DSD 5217 Disk/Tape/Floppy Controller

Introduction

The DSD 5217 Controller provides, on a single board, the capability to control winchester and floppy disks, and cartridge tape drives. The controller can handle two of each device. Typical configurations contain 1-2 Winchester disks and either one cartridge tape or floppy disk drive. The controller, like others used in SGI workstations are µprocessor controlled and therefore handle all control of attached I/O devices locally and independent of the workstation's main processor.

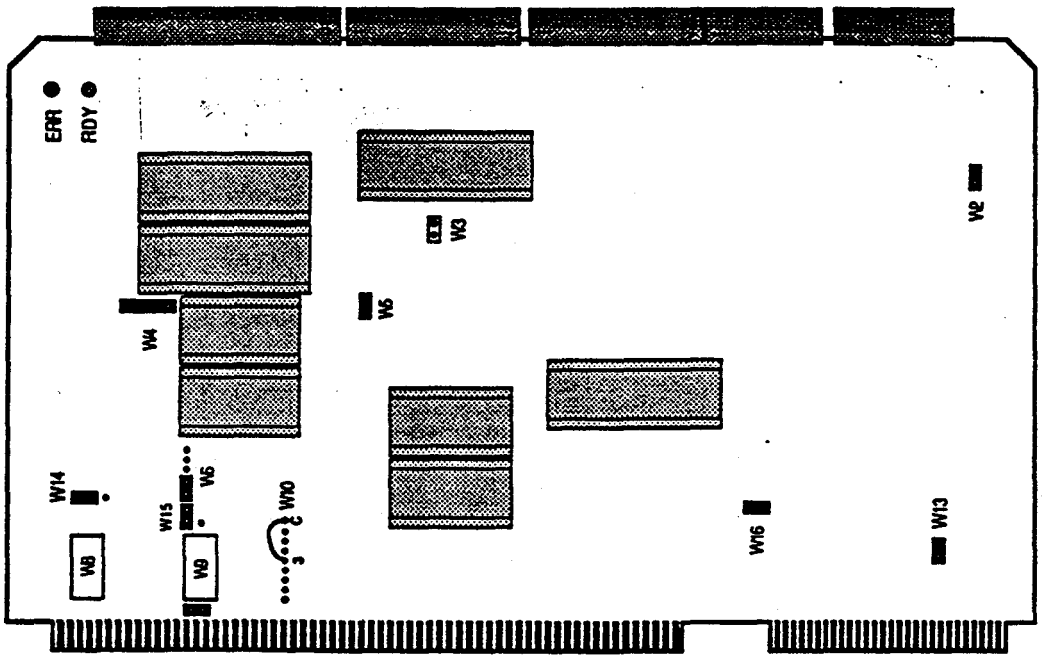


Exhibit 8. Controller Layout

Controller Interconnect

The exhibit below shows the signal interconnections between the controller and attached devices.

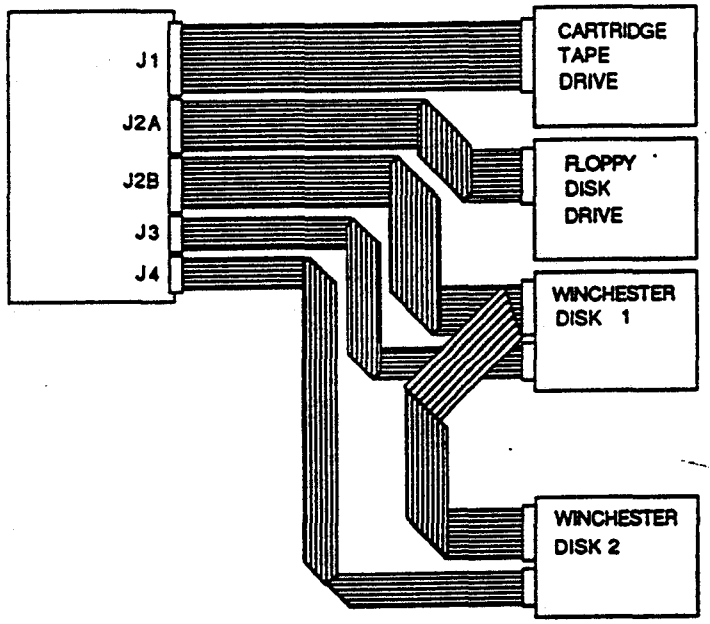


Exhibit 9. Controller/Drive Signal Interconnections

DSD 5217 Signal Interconnections			
Conn	Type	Description	Maximum Length
J1	50-pin flat ribbon	QIC02 tape interface	32 ft.
J2A	34-pin flat ribbon	SA460 floppy disk interface	20 ft.
J2B	34-pin flat ribbon	ST506 Hard disk interface	10 ft.
J3	20-pin flat ribbon	Hard disk read/write data	20 ft.
J4	20-pin flat ribbon	Hard disk read/write data	20 ft.

NOTE

Connectors J3 and J4 are interchangeable between drives.

Configuration

DSD 5217 Jumper Configuration					
Jumper	Function	Pin #	In	Out	Options
W2	Factory set	1-2	X		None
W3	Data Transfer Rate	1-2	X		16-bit data bus select.
W4	Factory Set	1-2	X		None.
		3-4	X		
W5	Bus Arbitration	1-2	X		Yield to higher priority
	Mode Select	2-3		X	bus request
		2-4			X
		5-6			X
W8	Serial or Parallel	1-2	X		Serial multibus priority
	Bus Priority				Scheme selected
W10	Multibus Interrupt	C-0		X	Interrupt Level 0
	Level Select				(Highest)
		C-1		X	Interrupt Level 1
		C-2		X	Interrupt Level 2
		C-3	X		Interrupt Level 3
		C-4		X	Interrupt Level 4
		C-5		X	Interrupt Level 5
		C-6		X	Interrupt Level 6
		C-7		X	Interrupt Level 7
W13	Factory Set	1-2	X		None
W14	I/O Address Size	1-2		X	
	Select	2-3	X		16-bit Address Select
W15	Factory Set	A	X		None
		B		X	
W16	Factory Set	1-2	X		None

Jumper blocks W7 and W9 are shown in the following table. These two blocks define the wake-up address for the controller. The wake-up address is used by the workstation's main processor to initiate communications between the controller and itself.

Wake-up Address Configuration				
Jumper	Function	Pins	1	0
W9	Wake-up address (Least significant byte)	0		X
		1		X
		2	X	
		3		X
		4		X
		5		X
		6		X
		7		X
W7	Wake-up address (Most significant byte)	8		X
		9		X
		10		X
		11		X
		12		X
		13		X
		14		X
		15		X

Diagnostics

Some diagnostic tools to aid in isolating a faulty controller are included with the board. Available are:

- PROM-based selftest programs
- Activity LED indicators

The selftest programs executes during power-up or system reset and verifies that the board is functional. If a failure occurs, the Error LED (CR1) and Activity LED (CR2) are used to indicate the completion state. Also, during normal system operation they are used to indicate the state of the controller. The normal LED sequence is shown in the table below:

LED Sequence				
Sequence	Power	CR1	CR2	State
During power-up or system reset	On	On X	On X	Functional Not functional
During selftest	On	Off On	On On	Functional Failed selftest
While running	On	Off Off On Blinking	On Off Off Off	Ready Busy Not valid Error code

When the selftest is complete, the CR2 LED should be lit and CR1 should be off indicating that the controller is functional. If CR1 ia blinking, then a board failure occurred.

Error Codes

CODE	DESCRIPTION
11-13	Reserved
14	RAM Error.
15	ROM error.
16	Seek in progress.
17	Illegal format type.
18	End of media.
21	Illegal sector size.
22	Diagnostic fault.
23	No index detected.
24	Invalid command.
25	Sector not found.
26	Invalid address.
27	Selected unit not ready.
28	Write protect.
31-33	Reserved.
34	Data ECC error.
35	ID ECC error.
36	Drive fault.
37	Cylinder address miscompare.
38	Seek error.
41	Data field not found.
42	Wrong type of data field.
43	Index too early.
44	Index too late.
45	Read/write controller error.
46	Bus timeout error.
47	No drive exists.
51	Tape cartridge not in place.
52	Tape cartridge write protected.
53	Tape drive not online.
54	Tape unrecoverable data error.
55	No data on tape.
56	Data miscompare during diagnostic.
57	Miscellaneous (cause unknown) tape error.

Interphase SMD 2190 Disk Controller

Introduction

The Interphase 2190 Disk Controller is used in workstations using the 474 megabyte Fujitsu disk drive. It is required to handle the increased data transfer rate of the drive. The controller provides Direct Memory Access transfers of disk data across the Multibus, prefetching of disk data through an onboard cache memory, and data error detection and correction.

Controller Interconnect

Exhibit 10 shows the signal interconnection between the controller and drive.

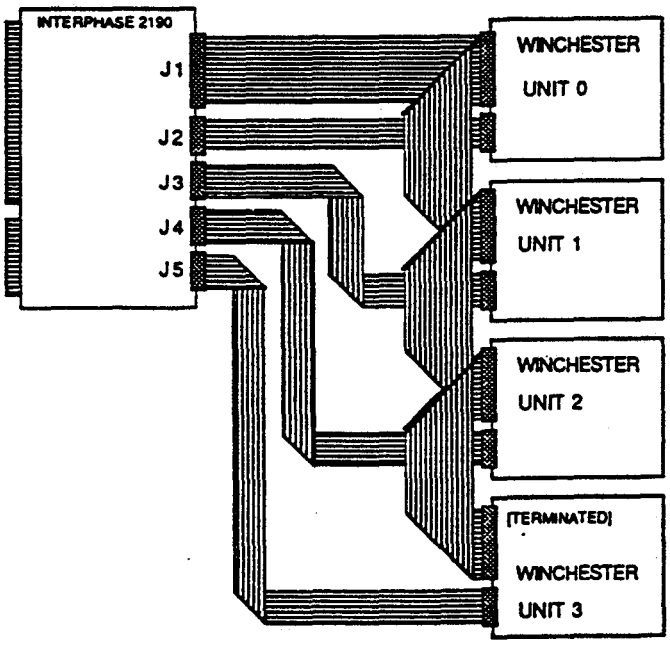


Exhibit 10. Interconnections

Configuration

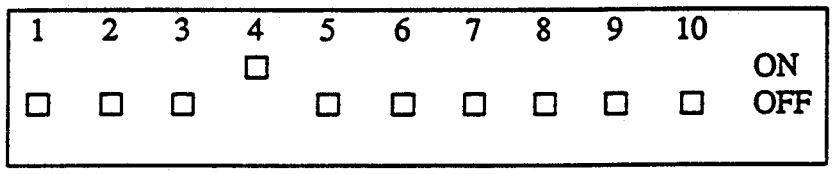


Exhibit 11. Switch S1 Configuration

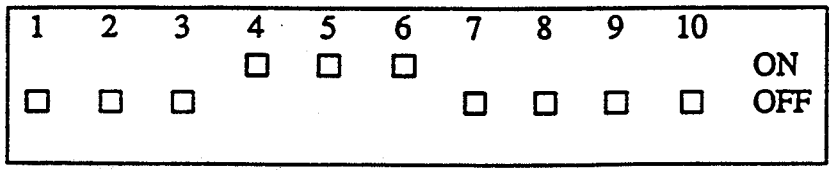


Exhibit 12. Switch S2 Configuration

The following table shows the jumper strap configuration for the various jumpers on the Interphase 2190 Disk Controller.

Interphase 2190 Disk Controller Strapping					
Jumper	Function	IN	OUT	Strap	
W1	CBRQ	1-2	X		
W2	Unit Selection	1-C-2			C-2
W3	Unit Selection	1-C-2			C-2
W4	Sector Flag	1-C-2			C-1
W5	Sector Flag	1-C-2			C-1
W6	-5vdc Reg	1-C-2		X	
W7	PROM Size	1-C-2			C-1
W8	RAM Size	1-2	X		1-2
W9	Buffer Size	1-2	X		1-2
W10	Buffer Size	1-C-2			C-2
W11	8/16 Bit I/O	1-2	X		1-2
W12	I/O Decode	1-C-2			C-1
W13	I/O Decode	1-C-2			C-1
W14	Priority	1-C-2			C-1
W15	BCLK	1-2		X	
W16	Priority	1-C-2			C-1
W17	Interrupt	C-0...7			C-5
W18	-5vdc Source	1-C-2			C-1

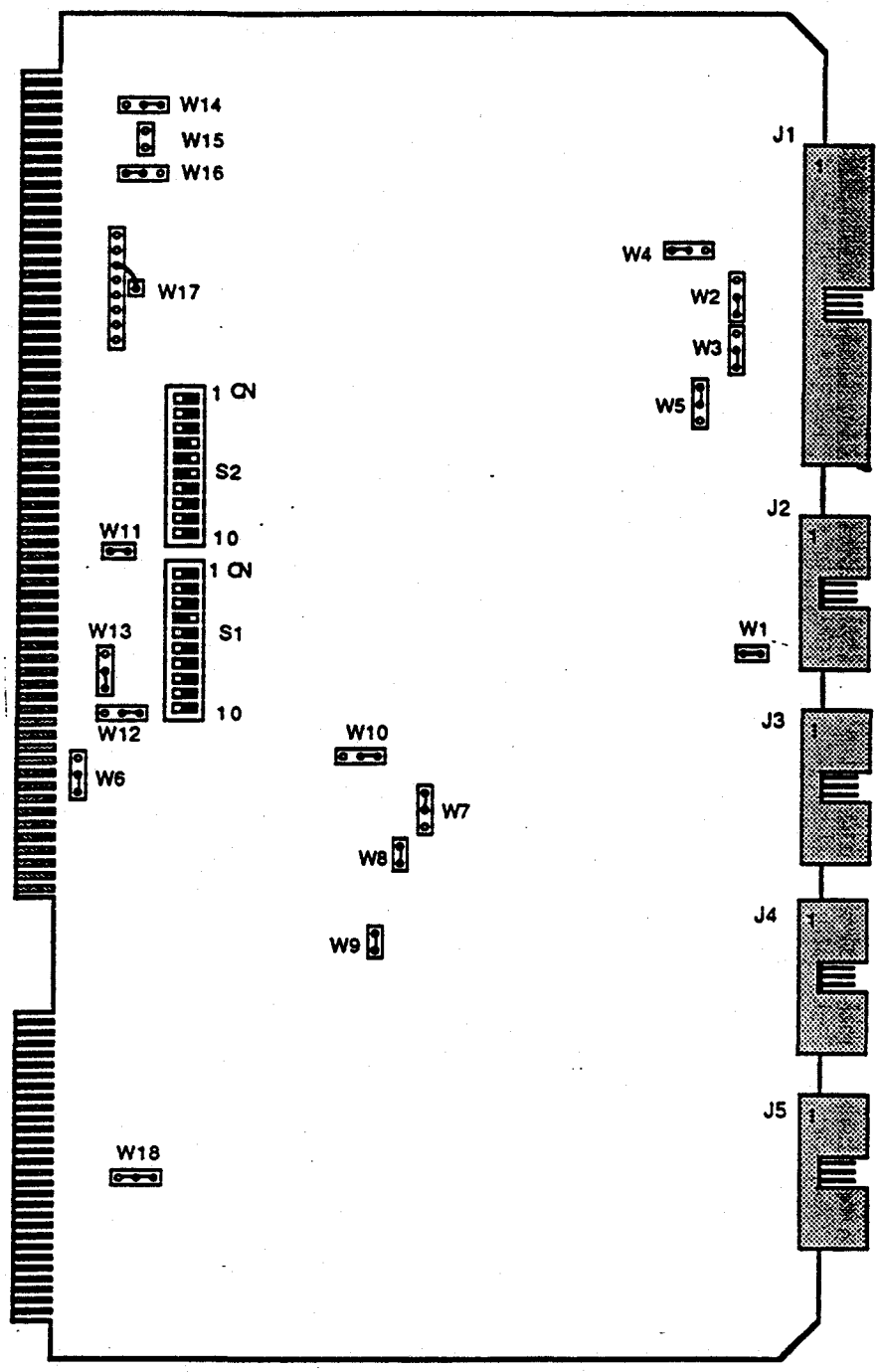


Exhibit 13. Interphase 2190 Disk Controller Board Layout

Error Codes

Interphase 2190 Controller Error Codes		
Code	Error	Description
10	Not Ready	Drive READY signal not asserted.
11	Invalid Address	Invalid drive physical address
12	Seek Error	Unable to locate a desired sector
13	ECC Error	Uncorrectable read error in the data field
14	Invalid Cmd	The IOPB command code is not valid
15	Reserved	
16	Invalid Sector	The IOPB desired sector value is invalid
17	Reserved	
18	Bus Timeout	Multibus acquisition did not occur within three msec. of a request
19	Reserved	
1a	Wrt Prt	An attempt tp write to a write protected drive occurred
1b	Not Selected	Controller attempted to select a drive with no response
1c	No AM	No sync information is found in the header of the target sector
1d	Reserved	
1e	Drv Fault	A fault condition exists in the selected drive
1f	Reserved	
20	Reserved	
21	Reserved	
22	Reserved	
23	Uncorr ECC	An uncorrectable ECC error occurred in the data field of a sector
24	Reserved	
25	Reserved	
26	No Sector	A sector pulse from the selected unit is missing
27	Overrun	During a data transfer, a missing xmit or recv clock was detected
28	No Index	An index pulse was not detected within 65 msec.
2a	Hd Sel Error	The head number read from the ID field did not compare with the actual head selected.
2b	Invalid Sync	Did not correctly read the Sync character
2d	Sk Timeout	A seek operation was not completed within 500 msec
2e	Bsy Timeout	Busy active for more than 500 msec
2f	Off Cyl	Drive was off cylinder for more than 3 seconds after being selected

Interphase 2190 Controller Error Codes (Continued)		
Code	**Error**	**Description**
30	RTZ Timeout	A restore command did not complete within 3 sec
31	Wrt underrun	Controller could not supply write data to the drive fast enough for the drive to continue
40	No Init	A command was attempted on an uninitialized drive
42	Gap Spec	The defined value for gap 1 or 2 is too small
4b	Sk Error	a seek error was reported by the drive
4c	Mapped Hdr	No sector pulse detected on a track to be mapped
50	Sctr/Trk Err	The defined sector per track number is zero
51	Bytes/Sctr Err	The defined bytes per sector count is too large
52	Interleave	The defined interleave factor is either zero or greater than the number of sectors per track
53	Inv Hd Adr	The target head in the IOPB was greater than the drive's actual number of heads
5d	Inv Burst Cnt	The specified DMA burst count causes the controller to attempt to transfer an odd number of bytes

Ciprico Tapemaster 1000 Tape Controller

Introduction

The Ciprico tapemaster 1000 controller is designed to control up to eight 1/2-inch magnetic tape drives. It controls start/stop and/or streaming tape drives conforming to the Pertec interface specification. Bus transfer rates to 4MB/sec can be achieved, while tape transfer rates of up to 1.5MB/sec can be attained.

The Tapemaster is controlled through addressable read/write registers. Tape commands are are initiated by the system processor by means of I/O parameter blocks which reside in system main memory. Actual tape operations are performed by the Tapemaster itself, thereby relieving the system processor from tape I/O responsibility.

Controller Interconnect

Exhibit 14 shows the signal interconnection between the tape controller and drive(s).

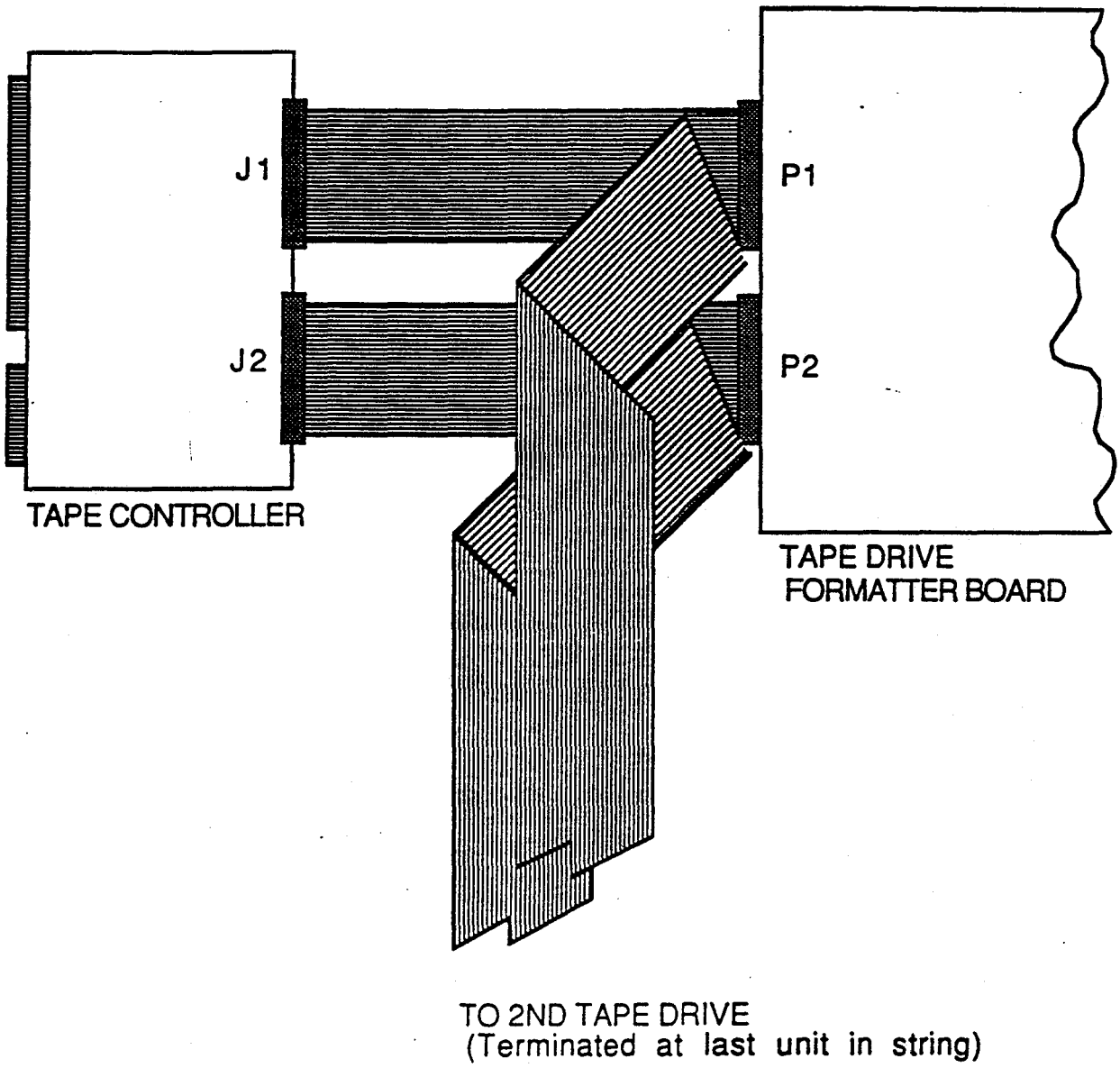


Exhibit 14. Tape Controller/Tape Drive Interconnection

Configuration

The following table shows the controller board jumpers and switches.

Tapemaster 1000 Switch/Jumper Settings		
Strap	Setting	Description
E1	1-2	Vendor installed
E2	2-3	Multibus diagnostic disable
E3	None	Remote diagnostics adaptor enable
E4	1-2	Multibus CBRQ/ signal enable
E5	1-2	Multibus serial priority select
E6	1-2	Vendor installed
E7	I5	Multibus interrupt level 5 select
S1	7	I/O address (MSB)
S2	0	I/O address
S3	9	I/O address (LSB)

Refer to Exhibit 15 for switch and jumper locations.

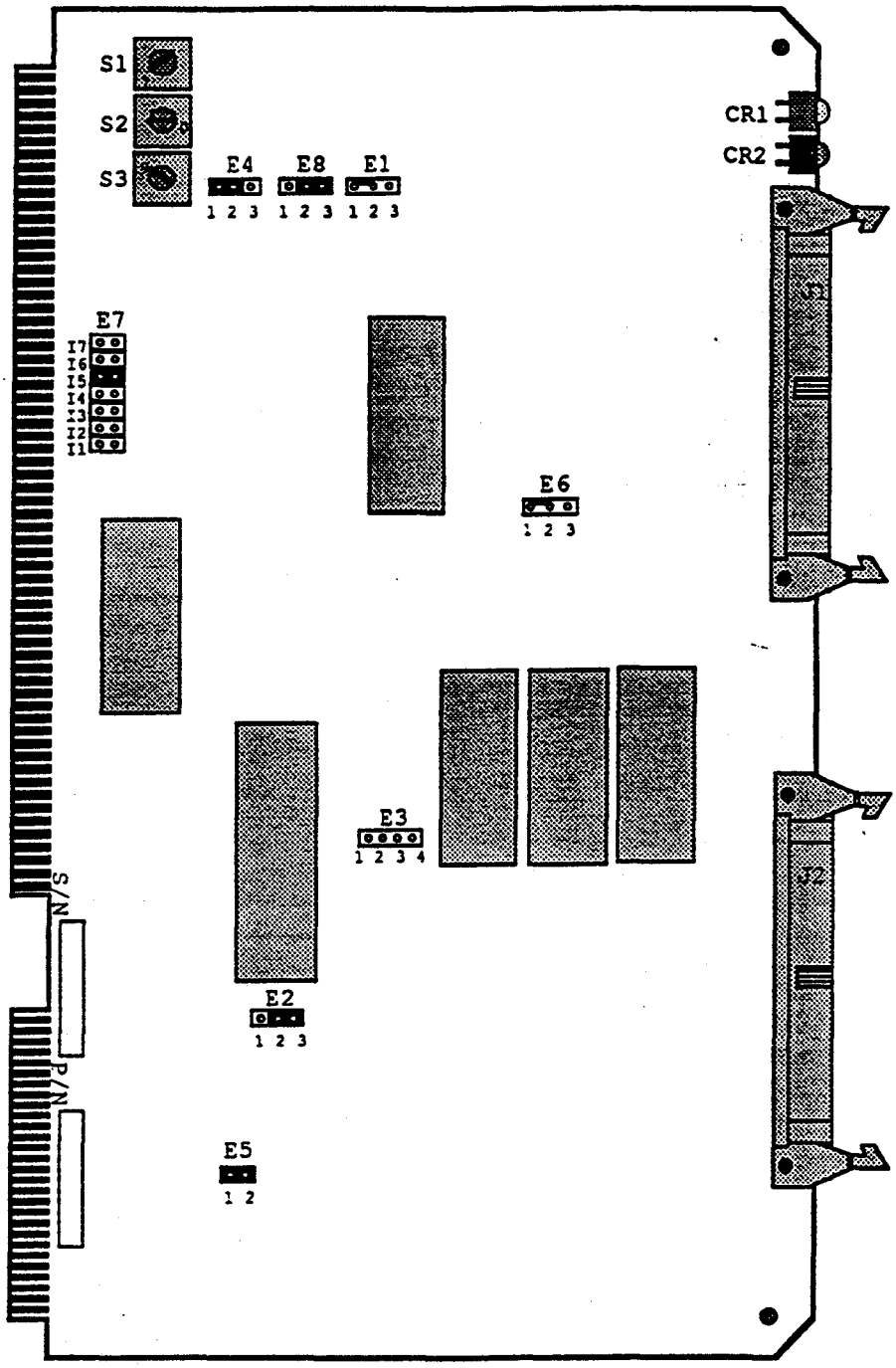


Exhibit 15. Ciprico Tapemaster 1000 Board Layout

On-board Tests

Local PROM-based diagnostics are invoked upon system reset or power-on. They perform a functional go/nogo board functionality test. The red LED (Error) is lit if an error occurs. Successful completion should occur within one second with the green LED (ready) lit and the red LED extinguished.

System-level Tests

At the operating system level, several tools are available to test the availability of the controller and the attached drive(s). They are the:

- "tar" command
- "cpio" command
- "mt" command

Error Codes

CODE	DESCRIPTION
01	Bus timeout error.
02	Cannot issue reverse command at BOT.
03	End of tape expected.
04	Unexpected filemark detected.
05	Formatter busy.
06	Hard tape error.
07	Invalid address to start swap.
08	Invalid bus width for swap.
09	Invalid byte count for swap.
0a	Invalid command issued.
0b	Invalid byte/record count parameter.
0c	Invalid retry count parameter.
0d	Invalid throttle size parameter.
0e	Actual read data record larger than expected.
0f	Linked parameter block error.
10	Online command issued but not supported.
11	Tape interface parity error.
12	Tape read data overrun.
13	Reversed onto BOT.
14	Actual read data record smaller than requested.
15	Correctable tape error occurred.
16	Tape write protected.
17	Tape drive busy.
18	Tape drive not online.
19	Drive not ready.
1a	Filemark not detected.
1b	Tape write data underrun.
1c	Invalid configuration block parameters.
1d	Blank tape encountered while reading.
1e	No "GO" signal. An error was detected during initialization.
1f	Invalid block command length.
20	Invalid block byte count.
21	Data Busy (never asserted) timeout.
22	Data Busy (always asserted)timeout.
23	No write strobes or incorrect number of write strobes.
24	Insufficient number of buffers ready.

CODE	DESCRIPTION (Continued)
80	Local memory error.
81	Parameter latch empty error.
82	Parameter latch full error.
83	PROM checksum error.
84	Parameter FIFO test error.
85	FIFO RAM test error.
86	Tape side test error.
87	Underrun test error A.
88	Underrun test error B.
89	Overrun test error A.
8a	Overrun test error B.
8b	Tape prescale test error A.
8c	Tape prescale test error B.
8d	Tape prescale test error C.
8e	Bus prescale test error A.
8f	Bus prescale test error A.
90	Bus timeout hardware error.
91	Parameter latch hardware error.
92	Parameter block pointer register error.
93	Controller error register failure.
94	Controller status register error.
95	Tape status register failure.

Tape Controllers

Ethernet Controllers

Excelan EXOS 201 Ethernet Controller

The EXOS 201 Ethernet Communications Controller is a high performance communications processor that connects the Multibus-based SGI Workstations to the Ethernet local area network. It is designed to offload much of the communications control burden from the CPU by executing control software locally.

The controller uses an on-board 8088 µprocessor and a PROM-based operating system to manage local operations. The workstation CPU controls the board primarily through a command and reply message scheme. The controller interprets the command messages and generates the replies.

The Ethernet controller performs the following Ethernet functions:

- Serial/parallel and parallel/serial conversion
- Address recognition
- Framing and unframing of messages
- Ethernet error detection and correction
- Frame header generation and removal
- Carrier sense
- Collision detection and enforcement

Controller Interconnect

Exhibit 16 shows the physical signal interconnection between the EXOS Ethernet Controller and the Ethernet cable.

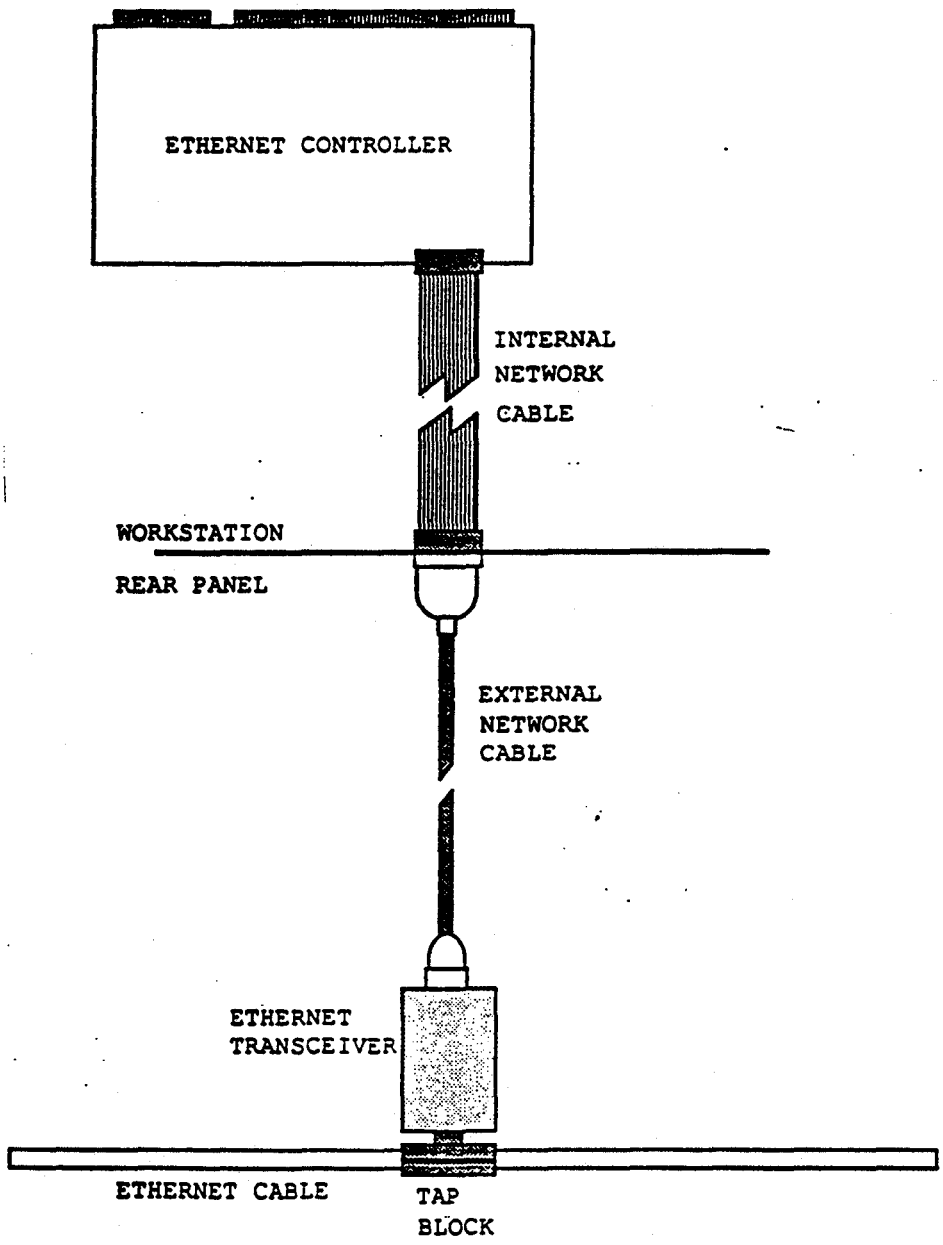


Exhibit 16. Ethernet Controller Signal Interconnection

Configuration

Exhibit 17 shows the Ethernet board layout and configuration:

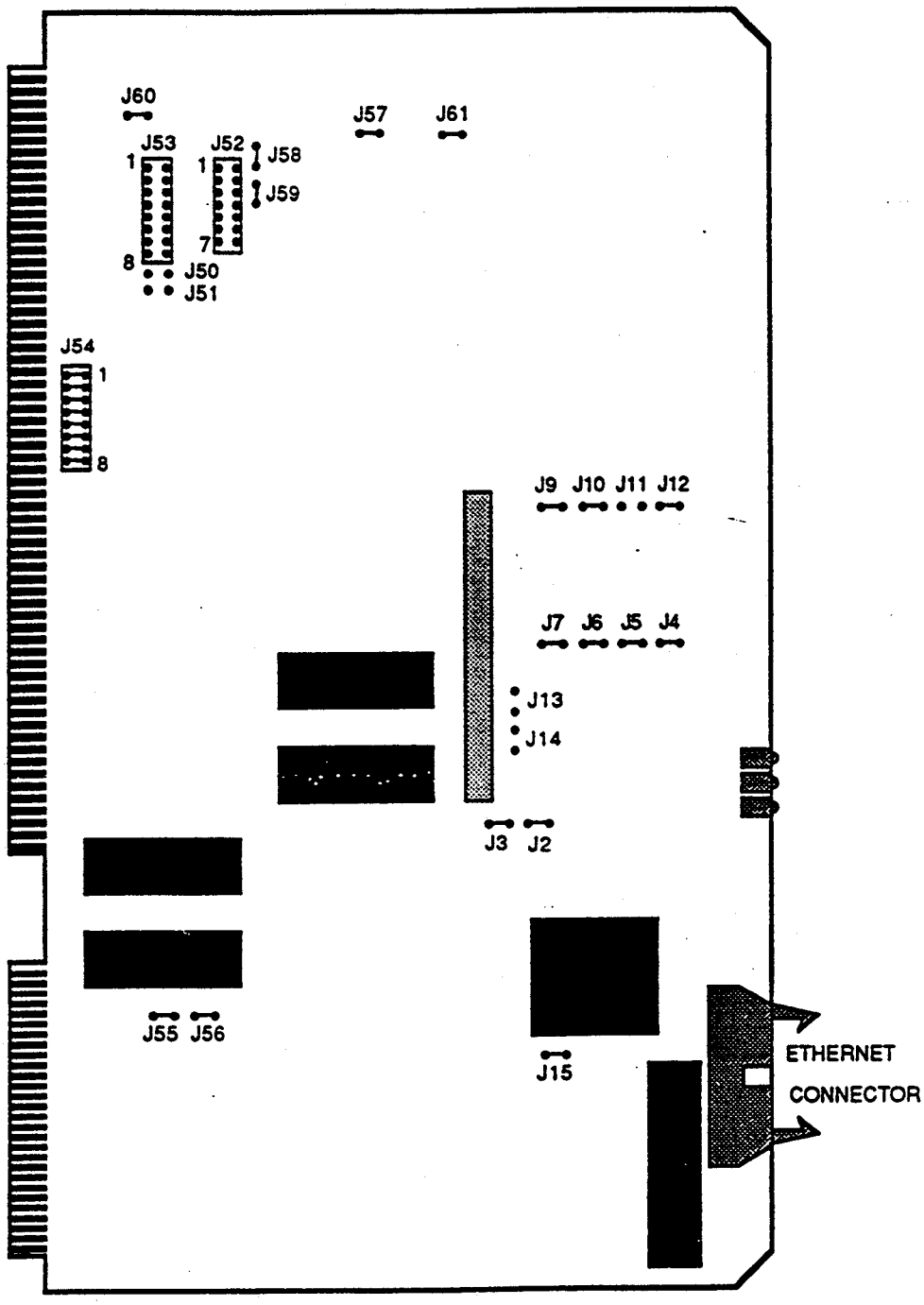


Exhibit 17. EXOS 201 Ethernet Controller Layout

Controller LED Display

EXOS Controller LEDs	
LED	Function
1	Controller Status
2	Ethernet Transmission Status
3	Multibus Status

Exhibit 18. Ethernet Controller LED Display

Controller Status LED

Selftest progress and status are communicated via this LED. It lights at system reset, and remains continuously lit while selftests are in progress. After selftests are completed, this LED will flash on and off (evenly) until the board operating software is downloaded. This usually occurs when the workstation is brought into multiuser state.

Once the software download is complete, the LED is turned off. Any other condition may indicate a board malfunction.

Ethernet Transmit Status LED

This LED is lit only while transmitting data onto the Ethernet.

Multibus Cycle Status LED

This LED, when lit, indicates that a Multibus cycle is in progress. Often when a workstation "hangs", it can be caused by a controller that has failed to release control of the Multibus. If this is the case, and the Ethernet controller is the culprit, then this LED will be continuously lit.

Error Codes

Diagnostic Error Messages

If the on-board diagnostics detect a functional hardware failure, then the controller Status LED will remain lit constantly, or display an error code by flashing a series of long and short pulses. Software errors during the controller download may also result in an error code display. Error codes consist of eight bit hexadecimal numbers and are presented bit-by-bit, starting with the most significant bit. A long pulse may be considered a "1"

and a short pulse a "0". The error code is continuously repeated with a short pause between each one.

Selftest Diagnostic Error Codes		
Error Code	Pulse Code	Description
A0	_·_· ····	Invalid address for configuration message.
A4	_·_· ·_··	Invalid operation mode parameter.
A5	_·_· ·_·_	Invalid host data test pattern.
A7	_·_· ·___	Invalid configuration message format.
A8	_·_· _···	Invalid movable data block parameter.
A9	_·_· _··_	Invalid number of processes parameter.
AA	_·_· _·_·	Invalid number of mailboxes parameter.
AB	_·_· _·__	Invalid number of address slots parameter.
AC	_·_· __··	Invalid number of hosts parameter.
AD	_·_· __·_	Invalid host queue parameter.
AE	_·_· ___·	Improper objects allocation.
AF	_·_· ____	Net boot failed.

Selftest Diagnostic Error Codes (Continued)		
Error Code	Pulse Code	Description
B0	_·__ …·	On-board PROM checksum failed.
B1	_·__ …_	Memory test failed for addresses 0-128K.
B2	_·__ .._·	Memory test failed for addresses 128K-top.
B3	_·__ .._	Counter test failed.
B4	_·__ ._..	Interrupts test failed.
B5	_·__ ._._	Transmission test failed.
B6	_·__ .__·	Receive test failed.
B7	_·__ .___	Local loopback data path test failed.
B8	_·__ _···	CRC test failed.
B9	_·__ _.._	Physical address EPROM checksum failed.
BA	_·__ _._·	System error.
BB	_·__ _·__	Ethernet chip initialization failed.
BC	_·__ __..	Ethernet chip selftest failed.
BD	_·__ __._	Ethernet chip resource counter failed.
BE	_·__ ___·	External loopback test alignment error.
BF	_·__ ____	iSBX board not in place.
C0	__.. …·	Specified time exhausted.
C1	__.. …_	Host memory read/write test failed.
C8	__.. _···	Parity hardware logic failed.
C9	__.. _.._	Multibus timeout interrupt failure.
CA	__.. _._·	Host interrupt test failed.
CB	__.. _·__	Command unit test failed.
CC	__.. __..	Divide error exception.
CD	__.. __._	Undefined interrupt type.
CE	__.. ___·	Command not executed.
CF	__.. ____	Command block sync failed.

SECTION 3: Test Items

1. Which disk controller(s) maybe used with the 72MB disk drives?

   ____ a. DSD 5214

   ____ b. Interphase Storager II

   ____ c. DSD 5217

   ____ d. Interphase 2190

   ____ e. All of the above.

2. The DSD 5217 is strapped at the factory for Multibus interrupt level_____.

3. An error message at the console:

   ```
   md0c: hard error, cmd=14, error=36
   ```

   appears at the console. Assuming the workstation is a 2400T with the standard disk controller, what error condition has occurred.

   _____

   _____

   _____

4. The Read/Write cables from each drive may be interchanged at the controller connectors without altering the drive address jumpers (True/False)._____

5. The IRIS 3000-Series workstations can be configured with up to

_____ megabytes of physical memory.

Contents

DIAG10 1-1
- Introduction 1
- Objectives 1
- Test Item 1
- Resources 1
- SECTION 1: General information and Loading Instructions 3
  - A. **General Information** 3
  - B. **Diagnostics: Component Tested Index** 3
  - C. **Tests: Component Tested Index** 4
  - D. **Loading a Test Program** 4
  - E. **Loading a Diagnostic Program** 5
- SECTION 2: Running Instructions for Ip2diag diagnostics 7
  - A. **IP2diag** 8
- SECTION 3: Test Items 19

DIAG10

Introduction

Programs designed to test the functionality of the workstation hardware are available on a diagnostic tape. This tape is available only to Field Engineers and is not standard with the file systems shipped with the workstation. In this module you will learn how to load and execute them.

Objectives

At the completion of this lesson, you will be able to:

- Match the diagnostic test program with its' functional description.
- Match diagnostic programs with the workstation models with which they are associated.
- Load and execute diagnostic test programs.
- Execute CPU, FPU, and memory diagnostic tests.
- Interpret CPU, FPU, and memory diagnostic test messages.

Test Item

Correctly run through a complete CPU, FPU, and memory diagnostic sequence and verify workstation functionality.

Resources

This guide

Diagnostic Cartridge Tape

The module contains three sections:

- **SECTION 1:**

  This section supplies general information and loading instructions for the Tests and Diagnostics.

- **SECTION 2:**

  This section contains running instructions for diagnostic ip2diag.

- **SECTION 3:**

  This section consists of a number of review questions that you should complete upon finishing the module.

SECTION 1: General information and Loading Instructions

A. General Information

The programs contained on the tape are grouped into *Tests* and *Diagnostics*.

- **Tests:**
  Tests are programs that run under UNIX. They are simple to run and the manual pages in section III supply you with all the information you will require.

- **Diagnostics:**
  Diagnostics are *stand-alone* programs that are booted from the Prom Monitor. They are much more complex to run than the Tests. Section II supplies you with the running instructions for *ip2diag*. In the next module, you'll learn about the graphic subsystem diagnostics.

The tape contains tests and diagnostics for 2400, 2400T, 2500T, 3020, 3030,and 3130 workstations. The *68020* based program names normally end with a **2**.

i.e. **ipfex** for 2500, **ipfex2** for 2500T, 3030 or 3130

B. Diagnostics: Component Tested Index

NAME	TYPE	MODEL	COMPONENT TESTED
*bmtest	diag	2400, 2500	PM2M memory, PM2 processor memory
*bpcd	diag	2400, 2500	Graphic Subsystem
*bpcd2	diag	2400T, 2500T, 3020, 3030	Graphic Subsystem
*dust	diag	2400, 2500	Serial Ports
*gf	diag	2400, 2500	GF2 graphic board
*gf2	diag	2400T, 2500T, 3020, 3030	GF2 graphic board
ipfex	diag	2500	474meg Hard Disk
ipfex2	diag	2500T	474meg Hard Disk
*ip2diag	diag	2400T, 2500T, 3020, 3030	IM1 memory, FP1 fpa board
mdfex	diag	2400	72meg Hard Disk
mdfex2	diag	2400T, 3020	72meg Hard Disk
*memtest	diag	2400, 2500	PM2M memory, PM2 processor memory
sifex2	diag	3030	170meg Hard Disk

*** These programs need an ASCII Terminal connected to port 2.**

C. Tests: Component Tested Index

NAME	TYPE	MODEL	COMPONENT TESTED
bpad	test	2400, 2500	Digitizer Pad
bpad2	test	2400T, 2500T, 3020,3030	Digitizer Pad
cprint	test	ALL	Color Printer
cttest	test	ALL	Quarter Inch Cartridge Tape Drive
dev	test	2400, 2500	Mouse and Keyboard Devices
dev2	test	2400T, 2500T, 3020, 3030	Mouse, Keyboard
enet	test	ALL	Ethernet Communications
fptest	test	2400, 2500	Floating Point Board (Skyboard)
fptest2	test	2400T, 2500T, 3020, 3030	Floating Point Board (FP1)
knobs	test	2400, 2500	Dial & Button Box
knobs2	test	2400T, 2500T, 3020, 3030	Dial & Button Box
lpen	test	2400, 2500	Lightpen
lpen2	test	2400T, 2500T, 3020, 3030	Lightpen
mon	test	2400, 2500	Monitor alignment
mon2	test	2400T, 2500T, 3020, 3030	Monitor alignment
mttest	test	ALL	Half Inch Reel Tape Drive
pattern	test	2400, 2500	Monitor check
pattern2	test	2400T, 2500T, 3020, 3030	Monitor check
timeout	test	2400, 2500	Display Blanking
timeout2	test	2400T, 2500T, 3020, 3030	Display Blanking
zclip	test	2400, 2500	GF2 graphics board with z-clipping
zclip2	test	2400T, 2500T, 3020, 3030	GF2 graphics board with z-clipping

D. Loading a Test Program

1. Boot UNIX from disk while in the Prom Monitor.

```
iris> b
```

2. Enter multi-user mode, if system is in single-user mode.

```
# multi
```

3. Login in as guest.

```
Login: guest
```

4. Change directory to /usr/tmp. If there is no /usr/tmp, then type

```
mkdir /usr/tmp

cd /usr/tmp
```

5. Insert Tests & Diagnostics tape into tape drive.

6. Load test program(s) from tape.

`cpio -ivhmudl <program_name>` to load a specific program or,

`cpio -ivhmudl` to load all programs from tape.

NOTE: To load all the tests and diagnostics, *9 megabytes* of disk space are required.

7. Run selected test program.

Type **<program_name>** of the program you want to run.

NOTE: If you get error: *Segmentation violation -- Core dump*, you are probably attempting to execute a diagnostic program.

If you get error: *Illegal instruction -- Core dump*, you are trying to execute a test for another model.

E. Loading a Diagnostic Program

1. Boot diagnostic program from tape while in the Prom Monitor.

Type `b mt0:<program_name>` at the iris prompt. (if 2400)

Type `b ct0:<program_name>` at the iris prompt. (170 or 380MB disk drive)

2. Boot diagnostic from disk if they are in */usr/tmp*.

   Type **b si0f:/tmp/<program_name>** (if 2400T and 3020)

   NOTE: If you get error: *bad magic number* in iris boot header, you are trying to boot a test program.

   If you get error: *FATAL ERROR: bus error*, you are trying to boot a diagnostic for another model.

SECTION 2: Running Instructions for Ip2diag diagnostics

This section illustrates how to test system boards **IP2, IM1, and FP1** using the *ip2diag* diagnostics.

The tests are **go-nogo** tests. If they fail, you must correct the problem by replacing a board, cable or reseating a board or cable.

The instructions show you the output associated with each command and or test executed. The commands are highlighted in **bold** print.

NOTE:
While running the individual tests you may experience system halts or test loops that do not end. If you encounter one of these conditions, depress reset to terminate the condition and enter PROM monitor. Reboot the test and try again.

A. IP2diag

- The system must have an ASCII terminal installed on serial port #2 in order to use this diagnostic.
- While running this diagnostic you will see three (3) different prompts:
  - **[n]->**
    This is the normal prompt, where **n** equals the command number.
  - **<q>**
    This prompt is used by the diagnostic to inform you that additional information is waiting to be displayed if you depress **return.**
  - **->**
    This prompt is used after you have requested help information concerning a specific command. After the diagnostic displays the help file, it places this prompt in front of the command to be executed.

STEP 1: Booting the diagnostic.

The diagnostic can be booted from the tape or disk if you had created a */usr/tmp* directory and loaded the tests into the directory.

If your customer is pressed for disk space then the latter method should not be used since the tests require approximately *9 meg* of disk space.

Load the diagnostic tape and type:

```
iris> b  ct0:ip2diag
```

The example will show the boot from the disk.

```
iris > b  si0f:/tmp/ip2diag


SGI Extent Filesystem
Loading: si.0f:/tmp/ip2diag
   Text: 00b284  bytes
   Data: 005240  bytes
   Bss : 0074ac  bytes (cleared)
```

```
Jumping to loaded program @ 20000000.
```

The following will appear at the ASCII terminal on port #2.

```
pteinit: j=0, *RTCC = fffffff0
pteinit. IP2 PCB board
  IP2 Diagnostic: Thu Sep 12 15:51:55 PDT 1985
  Type <space> to abort the current command
[1]->
```

The normal prompt is displayed and the diagnostic is telling you that if you want to stop (abort) a running command, use the space bar.

STEP 2: Displaying the command menu.

Use ? to display the available commands.

```
[1]-> ? <return>

badbyte   fpdiag    lowtest   readbyte  toutest   ?
defmacro  fpwalktestmacro     readword  writebyte !
errors    getperror mbmap     readlong  writeword
fptest    help      memtest   revision  writelong
fpreginit hightest  partest   rtc       $
fpoverlap history   quit      seemacro  .
```

The commands that are highlighted in bold print are the ones used by Field Engineers. The other commands are used by test tech's to set up loops and isolating to a failing chip.

The tests that begin with fp are floating point tests (FP1 board).

STEP 3: Displaying help menu.

The following illustrates how to display the help file.

```
                [2]-> help?

help
  BACKSPACE     Delete character
  DELETE           "        "
  CTRL-W        Delete argument
  TAB           Use default argument
  CTRL-L        Redraw command line
  CTRL-U        Erase command line
  ?             Print help test - type at any time
  SPACE         Interrupt command execution

<q> <return>
```

A command-line consists of a command string followed by zero or more arguments. Partially typed command strings are expanded, if possible, when a <space> is typed. Arguments are hex numbers separated by a single <space>. Additional spaces between arguments prompt for additional arguments, until none are left. Typing any character other than a space causes all prompts to disappear. Arguments that are not specified are given default values, which are displayed after the the command is entered.

<q> <return>

Please try again. Thanks..

-> help

When you type a command followed by the question mark (?), the help file is displayed. Notice the prompt (**<q>**) is displayed and the diagnostic waits for you to depress return before it displays the remainder of the help message.

Also, after the help file is displayed, the diagnostic displays the command (in the example, it is HELP) and waits for you to enter arguments, depress return to accept the default arguments, or enter control-U (**^u**) to erase the command.

Execute the command (help) by depressing **return**.

```
                -> help <return>

type <help?> for program information.
[3]->
```

STEP 4: Running Memtest with default arguments (parameters).

This test is used to check memory (the IM1 boards).

Before we run it, lets display the help file for this test.

```
[3]-> memtest?

memtest <tests> <verbose> <count>
  Test memory from LOWTEST to HIGHTEST (see those commands). Tests to run
  are specified as bits in the first argument. Verbose mode reports errors
  as they occur; otherwise, only the error count total is reported.

<q> <return>

Tests are:

     0          All 00000000
     1          All FFFFFFFF
     2          Alternate 00000000 and FFFFFFFF
     3          Alternate 55555555 and AAAAAAAA
     4          Data = Address
     5          Data = -Address
     6          Rotate 00000001
     7          Rotate FFFFFFFE

<q> <return>


-> memtest
```

After displaying the help file the diagnostic is waiting for you to enter arguments, depress return to accept the defaults or ^u to erase the command and enter another command.

We will accept the default arguments by depressing return.

```
                -> memtest <return>

-> memtest ff 1 1
 Testing memory from 00019000 to 007fffff
 pass 1: 01234567,  0 errors,  0 total.
[4]->
```

The test told you where the testing began and ended. Remember, the examples were generated on a workstation with 8 meg of memory, therefore 007fffff is the ending address.

- **Defaults:**
  ff = select all tests.
  1 = verbose error reporting.
  1 = number of times test will run.

- **Output**
  Pass 1: = The pass count of the test.
  01234567 = These numbers are output as the individual tests are run.
  0 errors, 0 total = The error summary.

If errors were detected during the test, then additional error messages would be output indicating what and where.

STEP 5: Running Memtest with supplied arguments.

Let's run only tests 2 & 4, three (3) times using verbose error reporting.

```
[4]-> memtest  14  1  3  <return>

 Testing memory from 00019000 to 007fffff
 pass 1: 24,  0 errors,  0 total
 pass 2: 24,  0 errors,  0 total
 pass 3: 24,  0 errors,  0 total
[5]->
```

NOTE:
Notice that the test begins at location 00019000. The diagnostic program itself resides below this limit. If you needed to test these locations you would need another memory board with its edge address switches set for the low order addresses in the system, and change the switches on the existing low order board to be the high order board.

Also, notice the test argument value of 14 was used to select tests 2 and 4. The following table illustrates the test bit weight within the test argument.

Test	Bit weight
0	01
1	02
2	04
3	08
4	10
5	20
6	40
7	80

STEP 6: Changing the address test limits of Memtest.

The following command shows how to change the address test limits.

Remember, the system used for these examples had 8 meg of memory.

I will now change the lower address test location to 400000. Memtest will begin testing at the second meg (400000). The address is changed using the **lowtest** command.

Lets first request help for the command....

```
[5]-> lowtest?

lowtest  <address>
   Prints the lower memory test bound if no argument is given. A single
   argument replaces the current lower bound. The low bound of memory
   (to avoid the test program) is always printed.
-> lowtest
```

```
-> lowtest  400000

lowtest:     00400000
lowmem : 00019000
```

Again, run memtest, selecting only test 3 with verbose error reporting and two (2) passes.

```
[6]-> memtest  08  1  2

 Testing memory from 00400000 to 007fffff
 pass 1:  3,  0 errors,  0 total
 pass 2:  3,  0 errors,  0 total
[7]->
```

Remember, these tests are *go-nogo*. If they do not find an error that you are experiencing while UNIX is running, margin the power supply voltages of **+5v and -5v** about 5% and run tests again.

STEP 7: Testing the FP1 board.

We will run three floating point tests to test the FP1 board:**fptest, fpwalktest & fpdiag.**

These tests are go-nogo as far as you are concerned.
Use the default parameters and replace boards or cables if failures occur. (Boards that could affect test are: IP2, FP1 and IM1)

As in the previous examples, lets first display the help file for the test about to be run...

```
[7]-> fptest?
fptest  <verbose> <count> <option reg>
  Exercises all numerical floating point operations.
  If any errors occur fptest displays a summary of errors.
  If verbose is 1, errors are reported in detail along with the summary.
  If verbose is 2, a summary of acceptable rounding errors is also displayed.
-> fptest
```

You will accept the default arguments for this test...

```
                -> fptest <return>

fptest  1  1  1
ERROR REPORT - 2c (hex)  FP  OPERATIONS  COMPLETED:

              SGL        SGLUPD     DBL        DBLUPD
opcode        errors     errors     errors     errors
-------------------------------------------------------
ADD             0          0          0          0
SUB             0          0          0          0
REVSUB          0          0          0          0
MUL             0          0          0          0
DIV             0          0          0          0
REVDIV          0          0          0          0
INT-FLOAT       0          0          0          0
FLOT-INT        0          0          0          0
SGL<->DBL       0          0          0          0
NEGATE          0          0          0          0
TST             0                     0
CMP             0                     0
[8]->
```

The above error output is normal, indicating a healthy board.

Lets run the next test.... After first displaying the help file.

```
                [8]-> fpwalktest?

 fpwalktest  <pattern> <targets> <verbose> <count>
   Rotates the 32-bit <pattern> and writes it to one or more of the
   1) result register, 2) register file, 3) ALU, 4) multiplier,
   and 5) division hardware. Any 32 bit pattern can be used,
   but 1 and -2 are suggested. Setting any bits of <verbose> causes error
   messages to be printed: setting <verbose> to two makes the fpwalktest
   stop when an error occurs, allowing you to further probe the FPA registers.

<q> <return>
```

Setting these bits of <target> exercises the following hardware:

0 Result register
1 Register file
2 ALU
3 Multiplier
4 Divide Hardware

```
-> fpwalktest
```

You should accept the defaults by depressing <return>...

```
                -> fpwalktest <return>

-> fpwalktest  1  1f  1  1
Will test X and Y bus for:
  result reg
  reg file
  alu
  mul
  division
0 errors this pass,  0 errors total.
```

The last test is fpdiag. Lets again display the help file, then run it accepting the default parameters...

```
                [9]-> fpdiag?

fpdiag  <verbose> <count>
```

```
->fpdiag
```

```
-> fpdiag <return>
```

```
-> fpdiag  1  1
The following hardware areas are:
                          OK      Worth Checking
Acknowledge               X
Outer Transceivers        X
Special Reg Transceivers  X
Mask Register             X
Error Register            X
Option Register           X
Bus Error Register        X
Condition Register        X
Inner Transceivers        X
X Register File           X
Y Register File           X
X Result Register         X
Y Result Register         X
X ALU input path          X
Y ALU input path          X
X MUL input path          X
Y MUL input path          X
X BUFFER fm ALU/MUL pathX
Y BUFFER fm ALU/MUL pathX
DIV SINGLE LOOKUP path    X
DIV DOUBLE LOOKUP path    X

[a]->
```

And that is all there is to testing the FP1 board.

STEP 8: Exiting IP2diag.

```
[a]-> q <return>
```

```
really... y
```

The system now boots into the PROM monitor.

SECTION 3: Test Items

None

Contents

BUGS10 1-1
- Introduction 1
- Objectives 1
- Sample Test Item 1
- Resources 1
- SECTION 1: The Boot Process 3
  - Boot Process Summary 4
- SECTION 2: Troubleshooting Hints 6
  - TROUBLESHOOTING HINTS: System will not Boot 6
  - TROUBLESHOOTING HINTS: ASCII Terminal Port 2 7
  - TROUBLESHOOTING HINTS: Minimum System to Run 8
  - TROUBLESHOOTING HINTS: Minimum System to Boot 9
  - TROUBLESHOOTING HINTS: Cannot Boot from Disk 10
  - TROUBLESHOOTING HINTS: PROM Monitor - page 1 11
  - TROUBLESHOOTING HINTS: PROM Monitor - page 2 12
  - TROUBLESHOOTING HINTS: Bit Plane Troubleshooting 13
  - TROUBLESHOOTING HINTS: Bit Plane Troubleshooting Map 15
- SECTION 3: Test Items 16

BUGS10

Introduction

The objective of the next three modules is to allow you to apply the knowledge learned and reinforce that learning by diagnosing induced failures on your workstation. This module provides you with some troubleshooting procedures that you will use to isolate failures.

Objectives

At the completion of this lesson, the participant will be able to:

- Recognize PROM monitor program configuration messages.
- Recognize IRIX boot process configuration messages.
- Match IRIX run-levels with their correct function.
- Recognize functions of IRIX configuration shell scripts and files.
- List the minimum board configuration required to:
  - boot IRIX
  - initialize the graphics subsystem

Sample Test Item

None

Resources

This module contains three sections:

- **SECTION 1:**

  This section contains a summary of the boot process for the IRIS-Series 3000 workstation.

- **SECTION 2:**

  This section provides a number of troubleshooting procedures that you will use in the next two modules.

- **SECTION 3:**

  This section contains review questions that you should complete after studying the material in this module.

SECTION 1: The Boot Process

The boot process occurs in a sequential manner which this document follows.

Several notes must be made:

- The demonstration was produced on a 3030 or 3130 workstation using Revision 3.6 software.

- The discussion is generic and meant to help you understand what occurs during the boot process and look at the process as a *first line* diagnostic to be used to help isolate system failures.

  The boot process supplies diagnostic information to the person who understands what occurs during the process.

Boot Process Summary

Sequence of events:

- Reboot or reset: The microcode reads IP2 revision level, sizes memory and reads rear panel configuration switches.

  If the microcode cannot communicate with the primary console, control is switched to the ASCII terminal on port 2.

- If configuration switch 5 is open, the microcode executes the boot command.

  Else, the PROM monitor is entered, from which the boot command must be executed manually.

- The microcode looks for the file **defaultboot** (or another boot file that was specified with the boot command) on the boot device and loads the boot program (Kernel) into main memory.

- The Kernel then performs the **auto-configuration** process.

  If specific types of errors are detected during the configuration process, or communication is lost with the primary console, error information is sent to the ASCII terminal on port 2 and the process is terminated.

- After the auto-configuration process completes and no errors have been detected, the Kernel invokes the **Init** process.

- **Init** scans the file **/etc/inittab** for the statement **initdefault:**

  If **initdefault** is present, **init** places the system into the specified run level. (IRIS default is single user)

  Else, the system prompts the user to enter a run level.

  If the default run level is **2** or **3, init** reads all configuration files in **/etc/inittab** that match the run level and executes all specified processes.

  Else, the boot process halts with the system in **single user** mode. The user must then execute **multi** to continue the bring up process.

Assuming **initdefault** specified run level 2 or 3, or the user executed **multi**, the process continues:

- Upon issuance of a run level change, **init** re-reads the file **/etc/inittab** and reads the files: *rc.s0, brc, bcheckrc, rc* and any other files that match the run level.

  Commands contained in these configuration files are executed in an order specified by the **action** field of each entry in **inittab**. (i.e. sysinit, bootwait, wait)

  - **rc.s0** links */dev/systty* and */dev/syscon*.
  - **brc** removes the old */etc/mtab* file and checks for the system model number.

    If the model number is present in the file **/etc/model**, the process continues.

    Else, the user is asked to enter the model number.
  - **bcheckrc** checks for the date and prompts user to answer if the date is correct and if **FSCK** is to be run.

    If **FSCK** is to be run, the file */etc/fstab* is read to determine what filesystems should be checked.
  - **rc** creates a new mount table file */etc/mtab*, mounts the specified filesystems, removes locks and logs, starts daemons and anything else the user specified in the **rc** file.

- **Init** starts a *getty* process for each ASCII terminal specified in the file **/etc/inittab**.

- The **login** prompt is displayed on all on-line terminals and the system is in **multi-user** mode.

SECTION 2: Troubleshooting Hints

TROUBLESHOOTING HINTS: System will not Boot

If your system does not boot and/or enter multi-user mode in a normal way, or your customer is complaining various terminals were not configured, or filesystems were not mounted, etc., before you start system troubleshooting you should always ask your customer if he/she changed any configuration file just prior to the failures.

i.e.

.login
.cshrc
.profile
bcheckrc
brc
checklist (In Rev. 3.4 and earlier)
crontab
fstab(Replaces rc.fs in Rev. 3.5)
gettydefs
group
inittab
mnttab (In Rev. 3.4 and earlier)
mtab(Replaces mnttab in Rev. 3.5)
model
multi
passwd
rc
rc.fs (In Rev. 3.4 and earlier)
rc.s0
rc.tcp
rc.xns
sys_id
termcap
ttytype

Remember, the key question is: Did you change any configuration files?

TROUBLESHOOTING HINTS: ASCII Terminal Port 2

- Remember, if the microcode or Kernel **cannot communicate with the primary console**, they look for an ASCII terminal on port 2 to output normal and error information.

- Know what the normal boot output should look like for each of the systems you maintain. A good idea is to make a copy of the auto-configuration output and keep it on site. (Be prepared)

- Know what the system acronyms mean.

  i.e.

  GF2 - Geometry Engine Board
  IP2 - Processor Board
  FBC - Frame Buffer Controller

TROUBLESHOOTING HINTS: Minimum System to Run

- If your system is experiencing Bus Errors or the system is dead:
  - The minimum system that will still run Flight is:

    IP2
    4 meg memory
    Disk Controller
    DC4
    UC4
    GF2
    2 bit planes (slots 16 & 17)

  - If Flight or system is still failing after establishing minimum system, try two other BP3's in slots 16 & 17.

- You can move BP3's and memory electronically by changing the board configuration switches. The following is a quick approach to possible bit plane problems if the boot process or Flight are failing:

  - Back out all bit planes except slots 16 and 17. If the system still fails, back out 16 and 17 and install 14 and 15 (with power off). Change the edge switches to reflect the first two boards.

    If the system still fails, the above procedure should eliminate the bit planes as the probable cause, except for possible cable problems.

TROUBLESHOOTING HINTS: Minimum System to Boot

- If you have a system with an FP1 and optional memory boards, the FP1 and all memory boards except the first 4 meg can be backed out and the system will still boot up. (You do not have to recable the boards)

- You can boot the system with GF2, UC4 and DC4 backed out if you have an ASCII terminal on port 2.

- You **cannot** boot if you have a hole on the multibus between IP2 and the Disk Controller.

  i.e. ENET board backed out.

- You could back out the Disk Controller and try booting over the network, but it will fail when the disk drive is accessed, but at least it will boot to a point.

TROUBLESHOOTING HINTS: Cannot Boot from Disk

If the system will not boot from disk, the following may help you isolate the problem.

- Can you list the contents of disk 0?

  ls hd0a:

- If you have a second disk, can you:
  - List the contents of disk 1?

    ls hd1a:

  - Boot from the second disk?

    b hd1a:vmunix1

    The above assumes the second disk has been configured per section 6.2.3 of the 3000 Owner's Guide (3 partitions). Partition a is loaded with the **skeleton** root system per section 4.4.3 of the 3000 Owner's Guide.

- Can you boot from tape?

  b ct0:sifex

- Can you boot over the network from host "doc"?

  n doc:vmunix

  The above command boots the education systems. Your account will have a different host name and may also have a different boot program name.

TROUBLESHOOTING HINTS: PROM Monitor - page 1

If you are experiencing system problems and want to verify if memory can be written into and read from, the PROM Monitor offers several commands that allow basic testing of the memory system.

The commands available are listed below with descriptions as to their functions:

fm{b|w|l} ADDR VALUE [INCR] [CNT]
fill memory as byte, word or long stating at ADDR, with initial VALUE, incrementing VALUE by INCR for CNT times. (INCR defaults to 0; CNT defaults to 1)

dm{b|w|l} ADDR [CNT]
display memory as byte, word or long. (CNT defaults to 1)

em{b|w|l} ADDR
edit memory interactively as byte, word or long.

Memory is addressed with a 24 bit address. The following table shows the address ranges for each meg and the card that contains the locations.

MEG	ADDRESS	CARD
1	000000-0FFFFF	1
2	100000-1FFFFF	1
3	200000-2FFFFF	1
4	300000-3FFFFF	1
5	400000-4FFFFF	2
6	500000-5FFFFF	2
7	600000-6FFFFF	2
8	700000-7FFFFF	2
9	800000-8FFFFF	3
10	900000-9FFFFF	3
11	A00000-AFFFFF	3
12	B00000-BFFFFF	3
13	C00000-CFFFFF	4
14	D00000-DFFFFF	4
15	E00000-EFFFFF	4
16	F00000-FFFFFF	4

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Bit
				512k	256k	128k	64k	32k	16k	8k	4k	2k	1k	512	256	128	64	32	16	8	4	2	1	Address Bit Weight
8m	4m	2m	1m																					

TROUBLESHOOTING HINTS: PROM Monitor - page 2

Two notes about PROM monitor memory commands:

- If you give argument values that the PROM monitor does not like, a bus error may occur or the PROM monitor goes into a halt or loop. If this occurs, just depress system reset.
- When you display memory (dm), if the bytes of data are valid ASCII characters, the characters are displayed next to the memory data.
- Addressing can be performed using three formats:
  - **Byte(8)** - last digit can be 0-F.
  - **Word(16)** - last digit must be 0, 2, 4, 6, 8, a, c, e.
  - **Long(32)** - last digit must be 0, 4, 8, c

The following illustrates several examples of memory reads and writes:

Command	Description
fml 0 0 0 ffffc	Clear first meg.
dml 0 100	Read 100 longwords starting at location 0.
	<Return> to keep reading, q to terminate read.
fml 0 0 4 ffffc	Write address in address for longwords in first meg.
eml 0	Edit (change) first location and continue sequentially until you depress q.
	A bug in the PROM monitor will not let you write an all F's value (FFFFFFFF) into a single location, but the command fml can write an all F's value.

TROUBLESHOOTING HINTS: Bit Plane Troubleshooting

1. Login as **guest**.

2. cd /usr/people/mexdemos/bin.

3. Execute **mex**.

4. Execute **showmap** (draw the largest possible window).

5. **Push** showmap to the background.

6. Execute **cedit** (sweep a small window in upper right corner).

7. cd /usr/people/mexdemos.

8. Execute **interp** (sweep a small window below **cedit** window).

9. Execute **demomakemap**.

10. **Push** the console window to the background.

11. Using **cedit**, make the third square (bottom line left) black.
(Place the cursor over the square, and press the left mouse button.)

12. Using **cedit**, make the last square (top row right) black.
(Place the cursor over the square, and press the left mouse button.)

13. Using **interp**, mark the third and last squares (left mouse button).
(Place the cursor over the square and press the left mouse button.)

14. Using **interp**, interpolate the color range between the third and last square. (Press the middle mouse button).

NOTE: If any bit plane is broken (hot bits) you will see spots.

15. You can repeat steps 11-14 using white to check for cold bits.

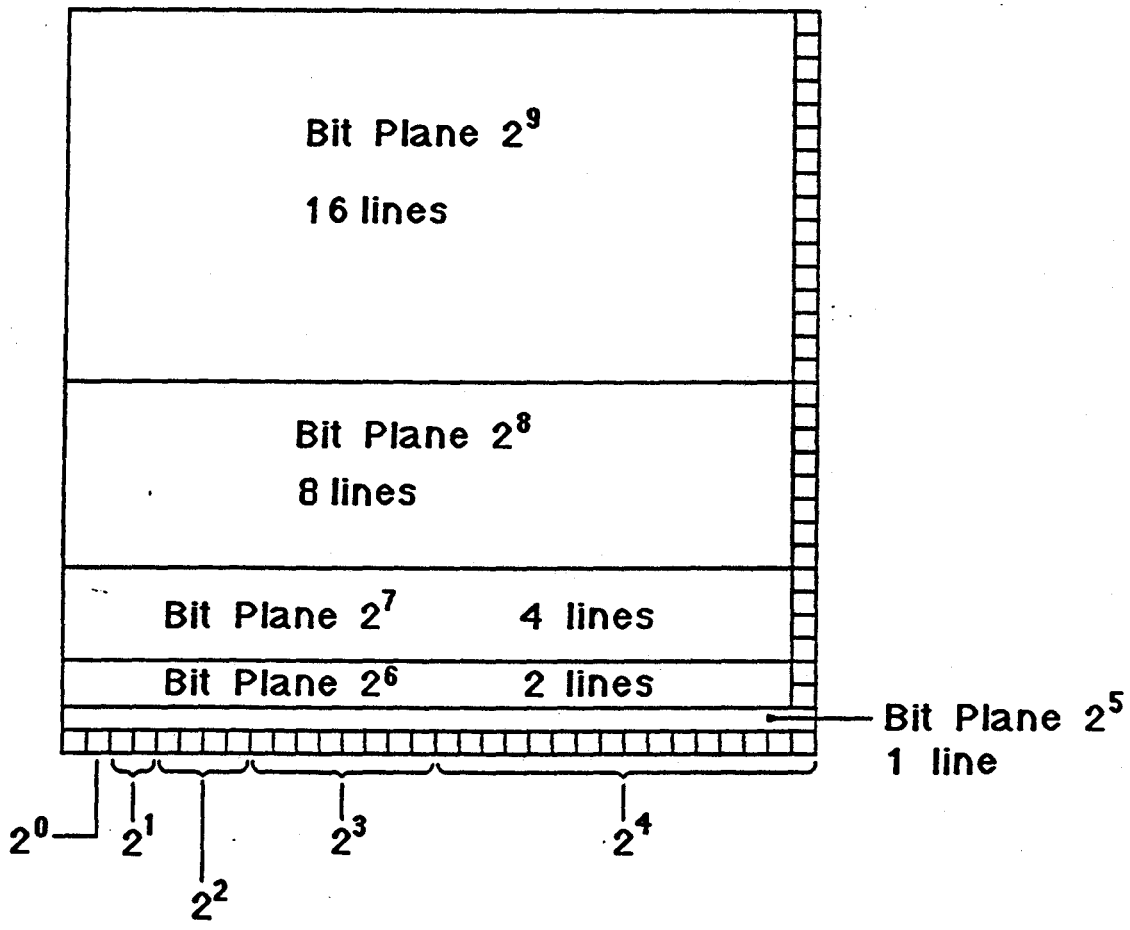
NOTE 1:
Showmap runs in **single buffer mode.** If you open a window that uses **double buffer mode** (i.e. jet), then Showmap will also run in double buffer mode changing the way that the bit planes are addressed.

NOTE 2: Addtional information regarding **cedit, interp, and showmap** may be found in Vol. 1A of the UNIX Programmer's Manual.

NOTE 2: Use the shell command **gclear** to restore the graphics screen display after completing each procedure. **See next page for a map of the bit planes if errors can be seen.**

TROUBLESHOOTING HINTS: Bit Plane Troubleshooting Map

If you can detect areas of the screen that are not correct using the procedure on the previous page, then use this chart to determine what bit plane is defective.



Pop up menus are bit planes 2^{10} and 2^{11}

SECTION 3: Test Items

None

Contents

DIAG1 1-1
- Introduction 1
- Objectives 1
- Test Item 1
- Resources 1
- SECTION 1:General Information 3
- SECTION 2: Running Instructions for Bpcd and Gf diagnostics 4
  - **Bpcd diagnostic** 4
  - **GF2 Board Diagnostic** 18
  - G3 Test Summary 27
- SECTION 3: Test Items 29

DIAG1

Introduction

In this module you will focus on the diagnostic programs that test the functionality of the graphics subsystem. The tests, *bpcd2* and *gf* perform thorough testing of all the graphics-related circuits.

Objectives

At the completion of this lesson, you will be able to:

- Execute graphics subsystem diagnostic tests.
- Interpret graphics subsystem diagnostic test messages.

Test Item

Correctly execute the Graphics Subsystem diagnostic sequence, verifying its' functionality.

Resources

This guide

Diagnostic cartridge tape

The module contains three sections:

- **SECTION 1:**

  This section supplies general information and loading instructions for the graphics subsystem diagnostics.

- **SECTION 2:**

  This section contains execution instructions for diagnostics *bpcd* and *gf*.

- **SECTION 3:**

  This section contains review questions that you should complete after studying this module's material.

SECTION 1:General Information

The diagnostics used to test the functionality of the graphics subsystem are:

- **bpcd**

  This diagnostic is used to test the boards of the graphic system; BP3 bit plane, DC4 display controller, UC4 update controller and GF2 geometry engine/frame buffer controller.

- **gf**

  This diagnostic is used to test the GF2 geometry engine board. The actual test gf is difficult to use, so three tests called g3, g21 and g2 were created from the commands of gf to test the GF2 board. The three tests are comprised of several *macro* commands which are made up of several of the individual commands of the gf test.

NOTE

The sequence illustrated should be the sequence that you follow when attempting to isolate a system failure:

1. Processor boards first. (IP2, IM1 and FP1)

2. Graphics subsystem (BP3, DC4, UC4)

3. Geometry Engine/Frame buffer Controller. (GF2)

SECTION 2: Running Instructions for Bpcd and Gf diagnostics

This section illustrates how to test system boards **BP3, DC4, UC4, and GF2** using the following diagnostics: *bpcd or gf.*

The instructions show you the output associated with each command and or test executed. The commands are highlighted in **bold** print and the resulting output is highlighted in *italics.*

All the test examples illustrated were run on a 3130 with the following configuration: 8 meg of memory, an FP1 board, and 32 bit planes.

Bpcd diagnostic

- The system must have an ASCII terminal installed on port 2 in order to use this diagnostic.
- While running this diagnostic you will see three (3) different prompts:
  - [n]->
    This is the normal prompt, where n equals the command number.
  - <q>
    This prompt is used by the diagnostic to inform you that additional information is waiting to be displayed if you depress **return.**
  - ->
    This prompt is used after you have requested help information concerning a specific command. After the diagnostic displays the help file, it places this prompt in front of the command to be executed.

STEP 1: Booting the diagnostic.

The diagnostic can be booted from the tape or disk if you had created a */usr/tmp* directory and loaded the tests into the directory.

If your customer is pressed for disk space then the latter method should not be used since the tests require *9 meg* of disk space.

If loading from tape use the following boot command:

- **b ct0:bpcd2** If 68020 based workstation.
- **b mt0:bpcd** If 2400. (see note)

The example will show the boot from the disk (it was a 3030).

```
iris > b  si0f:/tmp/bpcd2

SGI Extent Filesystem
Loading: si.0f:/tmp/bpcd2
   Text: 01840c  bytes
   Data: 014ac8  bytes
   Bss : 006a62  bytes (cleared)
Jumping to loaded program @ 20000000.
```

The following will appear at the ASCII terminal on port 2.

```
BPC Diagnostic: Wed Mar 12 14:47:44 PST 1986
  UC4
  DC4
  Interface 3 (UC4 Multibus Interface)
  IP2  (If this had been a 2400, this would be PM2)
Type <space> to abort the current command
[1]->
```

As you can see, the process is very similar to the IP2diag, if fact, both diagnostics were written by the same person.

NOTE:
If you are booting the diagnostic on a 2400 workstation, you MUST disconnect the primary monitor keyboard at the "J"- box and depress reset before you enter the boot command. This will force the output to the ASCII terminal. The following output will appear as the diagnostic is loaded.

```
iris > b  mt0:bpcd

Boot:mt.0:bpcd
OK.
Loading 0x1832C+0x157B0+0x0 at 0x1000
Received 187132/4:
```

```
Stack at 0xF6F61C;  Starting at 0x1000
```

The remainder of the output at the ASCII terminal is the same as it is for the 68020 based workstations.

STEP 2: Displaying the command menu:

Use **?** to display the available commands.

```
[1]-> ? <return>

ball          endmacro  lded          readdda   swap
bpctest       fillrect  ldfmaddr      readfm    time
bplamptest    filltrap  ldmode        readloop  traptest
bptest        fmtest    idrepeat      readmap   uclamptest
chartest      gftest    idxe          readword  ucr
clear         help      ldxs          read      wecode
cmndlist      history   ldye          recttest  we
colorcode     init      idys          request   writefm
colorwetest   joescale  linetest      restore   writemap
color         ldconfig  loopclearword           revisionwriteloop
dclamptest    ldddasaf  loop          rgbcolor  writeword
dcpal         ldddasai  macro         rotateloop write
dcrtest       ldddasdf  mapcolor      save      $
dcr           ldddasdi  maptest       screenmask.
ddatest       ldddaeaf  memtest       scrmsktest?
defmacro      ldddaeai  pixeltest     seemacro  !
depthcue      ldddaedf  quit          sigplanes
display       ldddaedi  ramptest      steptest
drawchar      ldec      randlines     stripetest
drawline      lded      randrect      stripe
[2]->
```

NOTE:
Most of the tests are for use by the floor technicians when trying to isolate a component on a board.

You should use the commands highlighted in **bold** print.

The test **bpctest** will run several of the listed tests in a pre-defined sequence. This test should be the second test run (after IP2diag) and if it detects failures, then the individual failing test could be run.

If an individual test name is followed with the ? character, that tests help file is displayed.

i.e. **bptest?**

STEP 3: Initializing the raster subsystem.

The first command you must execute upon entering *bpcd* is, **init.**

Init initializes the raster subsystem (before you begin testing) or returns the raster system to its initialized state (run init between tests).

Now look at the help file....

```
[2]-> init?
```

init <mode>
Intialize the raster subsystem, or return it to its initialized state.
Must be called before any UC command is executed. Arguments are

```
<q> <return>

    none return all values to defaults.
    1    change the default init values.
    2    size the bitplanes to determine the default values.
    3    interface init only, minimum state change to the raster syste
    4    like 'none', but bitplanes are not altered.
-> init
```

Notice that the test is waiting for you to enter arguments, depress return to accept the defaults, or **cntl-c** to erase the command.

If this is the beginning of testing (which it is), depress return to accept the default for the *mode* argument. The init process will prompt you to enter specific values that will define the configuration of your raster subsystem.

```
-> init <return>

-> init  0
configuration [0,ffff], df:
```

The above parameter can be set to a value from **0-ffff.** It's default is **df.** You will always **accept the default** for this argument.

```
configuration [0,ffff], df: <return>

mode [0,7], 0:
```

This parameter can be set to a value from **0-7**. It's default is **0**. You will **always accept the default** for this argument.

```
mode [0,7], 0: <return>
```

```
dc register [0,ffff], 0:
```

This parameter can be set to a value from **0-ffff**. It's default is **0**. This parameter will define the type of monitor you are using as your primary or secondary monitor. See the table below for the correct code that will define your monitor:

- **If you are using the STANDARD CONFIGURATION:**

  Primary monitor is 60 Hz (NI), enter: **3000**
  Secondary monitor is 30 Hz (I), enter: **0**

- **If you are using an OPTIONAL CONFIGURATION:** (configured in pairs)

  Primary monitor is 60 Hz (NI), enter: **1000**
  Secondary monitor is RS170A, enter: **2800**

  Primary monitor is 30 Hz (I), enter: **1000**
  Secondary monitor is RS170A, enter: **2800**

  Primary monitor is 60 Hz (NI), enter: **1000**
  Secondary monitor is EURO, enter: **2800**

  Primary monitor is 30 Hz (I), enter: **1000**
  Secondary monitor is EURO, enter: **2800**

The system I was on when this document was created was set for the Standard Configuration and the Primary Monitor selected, therefore, I will enter **3000** to the *dc register* prompt.

```
dc register [0,ffff], 0: 3000  <return>
```

```
color index [0,fff], 0:
```

This prompt is asking you what color the test should use as it's index (reference) color. There are 5 colors you can select, see the following table:

Value	Color
0	Red (default color)
1	Green
2	Blue
3	White
4	Black

The color is not important, so you can select any color. We will accept the default by depressing return...

```
color index [0,fff], 0: <return>
```

```
write enable code [0,ffffffff], ffffffff:
```

The value for this parameter is between 0-ffffffff. This is a *mask* for the bit planes to which you are going to allow diagnostic writing. There are 32 bits in the mask, each significant to one of the possible 32 bit planes. If the write enable (WE) bit is set (1), then the corresponding bit plane can be written. To select which planes you want to write into, see the following table:

	BIT PLANE NUMBER															
SLOT NUMBER	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	1 9	1 8	7	6	5	4	3	2	1	0
17																						X	X							X	X	
16																					X	X							X	X	
15																			X	X							X	X				
14																	X	X							X	X						
13							X	X							X	X																
12					X	X							X	X																		
11			X	X							X	X																				
10	X	X							X	X																						

i.e. If you only wanted to write into the bit planes of the board in slot 17, enter **303** to the *write enable* prompt.

Test slots 16 and 17, enter **f0f**.

Test slots 15,16 and 17, enter **3f3f**.

Assume we have all 32 bit planes installed and want to test them all, so accept the default by pressing <return>...

```
write enable code [0,ffffffff], ffffffff: <return>

significant bitplanes [0,ffffffff], ffffffff:
```

This prompt is telling you what bit planes the diagnostic was able to communicate with (what planes are on line). The example implies that all 32 bit planes are present and the diagnostic can see them. This is what I would expect since 32 bit planes are installed.

NOTE:
If you know how many planes are present in your system and this parameter indicates fewer then expected, then the diagnostic may have already found a problem for you.

With this parameter you can select what planes you want tested. The value entered is the same as the write enable code (we). If this is your first pass, it is best to test all the installed planes, so depress return..

```
significant bitplanes [0,ffffffff], ffffffff: <return>

  cfr: display ab, update ab, viewport, backline, ldlinestp
  mdr:
  deflags: pipe4, prom, vfsm-clear
  significant bitplanes:ffffffff
  installed bitplanes:ffffffff
  color index:          0
  write enable code:ffffffff
[3]->
```

Your screen will now be initialized to the color you selected with the *color index* prompt. If you had selected *0* (the default), your screen should be *Red*.

You are now ready to run a test. Remember, you must execute init between tests to

return all the parameters to the selected or default values. If you wish to enter new parameter values, then you must execute **init 2** to force the diagnostic to prompt you again for the individual parameters.

STEP 4: Testing the complete graphics subsystem.

As mentioned earlier, the best approach for isolating errors in the graphics subsystem is running the test *bpctest* after executing the *init* command.

This test runs several of the individual tests against all the boards in the graphics subsystem. If errors are detected, then the individual failing test(s) could be run to verify the failures and narrow the possibilities.

Look at the help file for bpctest...

```
[3]-> bpctest?

bpctest  <iterations> <verbose> <tests>
  Run an assortment of tests on the entire raster system. All tests are
  run non-verbose. Each test reports its error total if verbose is true
  (not 0). Otherwise only the error total of all of the tests is printed.
  Unlike other test commands, bits in the 'tests' argument specify the
  boards to be tested (rather than specific tests to run).

<q> <return>

  bit   meaning
  ---   -----------
  0     Run tests that are applicable to display controller.
  1     Run tests that are applicable to update controller.
  2     Run tests that are applicable to the bitplane boards.
  3     Run bptest (display on and off). This test is very long with UC3.
  4     Run the processor memory test (PM2 and IP2 only)
  5     Run the gf2 test (GL2 systems only - others will crash).
  6     Run the floating point test. IP2 with FP1 only.
-> bpctest
```

NOTES:

- Running **bptest** using only bit 3 (<tests> argument = 08) does not work due to a programming error.

- The default for <tests> is 3f. This runs all the tests except the floating point test because not all systems are shipped with an FP1 board.
- If the test is running on a 2400, only six choices are available; the FP1 is not available on 2400 systems.
- The test *memtest* is the same memtest that is part of the IP2diag.

Now run **bpctest** accepting the defaults...

```
-> bpctest <return>
```

```
-> bpctest  1  1  3f
 Pass    1: 01234567012345670123456701234567   global errors:    0
 read errors:0
 write errors:0
 weird errors:0
(round errors:0)
 memtest complete, 0 errors
 gftest complete,  0 errors
bptest complete, display ab, 0 errors
bptest complete, no display, 0 errors
stripetest complete, display ab, 0 errors
stripetest complete, display ab, sbuf, 0 errors
color/we test complete, 0 errors
fmtest complete, 0 errors
dcrtest complete, 0 errors
maptest complete, 0 errors
linetest complete, 0 errors
viewporttest complete, 0 errors  (scrmsktest)
rectangle test complete, 0 errors(recttest)
character test complete, 0 errors(chartest)
dda test complete, 0 errors
pixel test complete, 0 errors
trapezoid test complete, 0 errors(traptest)
errors this pass = 0, total errors = 0
[4]->
```

NOTE:
Notice that four of the test names in the output are different from those listed in the commands menu. The actual names are highlighted in bold print next to the used name.

If one or more of the tests reports errors, you could run those tests individually.

STEP 5: Running individual Bpcd tests.

The next few examples illustrate running tests against the individual boards in the graphics subsystem. Remember, the individual boards can be tested by changing the default value of the <tests> argument of bpctest.

Because the boards of the graphics subsystem are so tightly coupled, you have to keep in mind, that when testing the individual boards, other boards in the subsystem can affect the board function that you are testing. In each of the following examples, you are shown what board is being tested and what boards you should suspect if the individual test fails. (Test output is not shown for the following examples)

The first example illustrates how to test the DC4 board.

```
[4]-> init <return>

[5]-> bpctest  1  1  1
```

This configuration runs tests *stripetest, color/we test, fmtest, dcrtest, and maptest.*

- If dcrtest and/or maptest detect failures, DC4 is defective.
- If stripetest, color/we test, or fmtest detect failures, then cables, DC4, UC4, or a BP3 could de defective.

The next configuration runs tests *stripetest, color/we test, fmtest, linetest, viewporttest, rectangle test, character test, dda test, pixel test, and trapezoid test.* It checks the Update Controller (UC4).

```
[6]-> init <return>

[7]-> bpctest  1  1  2
```

- If linetest, viewporttest, rectangle test, character test, dda test, pixel test, or trapezoid test detect failures, then UC4 is the most probable cause.
- If stripetest, color/we test, or fmtest detect failures, then cables, DC4, UC4, or a BP3 could de defective.

The next configuration runs tests *stripetest, color/we test, and pixel test.* It is a basic check of the bit plane boards.

```
[8]-> init <return>

[9]-> bpctest  1  1  4
```

- If pixel test detects failures, then a bad bit plane is probably the cause.
- If stripetest and/or color/we test fail, then cables, DC4, UC4, or a BP3 board could be the problem.

The configuration (bpctest 1 1 8) is invalid due to a programming bug. The test *bptest*, which this should run, will run when the default 3f is accepted for bpctest, but will not run when you try to select it with bit three of the tests mask.

The following three examples show you how to exercise the IM1 memory boards, GF2 and FP1 boards using *bpctest.*

1. bpctest 1 1 10	Test memory (IM1)
2. bpctest 1 1 20	Test GF2 board
3. bpctest 1 1 40	Test FP1 board

Two tests that can be run individually and are of interest, are *stripetest and bptest.* Both tests are basic tests of the bit planes, but they test the planes differently.

- Stripetest uses a specific pattern that is displayed on the monitor. The DC4 board and cables are in the loop and should be suspect if failures occur.
- Bptest is really a memory test, remember, the bit planes are nothing more than RAM memory devices. The P2 bus is used to write and read the data into and out of the planes. Video appears at the monitor for this test, but it has no real pattern and it would be hard to spot a display problem in the pattern.

Now run *stripetest*....

```
[a]-> init <return>

[b]-> stripetest <return>
```

```
[b]-> stripetest  1  1  40  0
   pass     1:  cannot test planes f0f00000, C:4567,  D:4567
a0a1a2a3a4a5a6a7b0b1b2b3b4b5b6b7c0c1c2c3d0d1d2d3   badplanes:     0, cumul
ative:    0
[c]->
```

The test is telling you it cannot test the bit planes in slots 10 and 11. This is because there are no cables on these boards, remember, they are used for Z-buffering.

Errors are reported as characters in parentheses after the plane name.

The next example illustrates running *bptest*...

```
[c]-> init <return>

[d]-> bptest <return>
```

```
[d]-> bptest  1  1  3f  0  0
   pass     1: 012345, errors this pass = 0, total errors = 0
[e]->
```

While this test is running, you will notice dots (...) appearing between the test numbers as the individual tests of *bptest* are run.

Now look at the help file for *bptest*...

```
[e]-> bptest?
```

```
[e]-> bptest  <iterations> <verbose> <tests> <chip report> <diag>
   Test the operation of the bitplane memory. Errors are reported as they
   occur if verbose is true (not 0). Otherwise only an error total is
   reported. Individual tests are selected with bits of the test
    argument.

<q> <return>
```

```
        BIT  TEST
         0    All 0 test.
         1    All 1 test.
         2    Word = address.
         3    Word = compliment of address. UC4 only.
         4    Word takes random value.
         5    Exercise saveword/drawword hardware.
-> bptest
```

As you can see, *bptest* is nothing more than a memory test.

This ends the examples of *Bpcd*. The following illustrates how to get back to the PROM monitor.

```
-> bptest  ^u  (This clears the command line)

->  q
```

type 'y' to confirm `y`

NOTE:
Sometimes the system halts when `q` is typed, if this occurs, just depress reset to get back to the PROM monitor.

GF2 Board Diagnostic

- The system must have an ASCII terminal installed on port two to use this diagnostic.
- While running this diagnostic you will see the following prompt, **!**. It is the exclamation point.
- Three diagnostics, *g3, g21, and g2* will be used to test the GF2 board. These tests are comprised of macro commands built using individual commands from the parent diagnostic *gf2*. The macro commands are much easier to use than attempting to perform the same function using *gf2*.
  - g3 is used on 68020 based workstations.
  - g21 and g2 are used on 2400 workstations. These two tests perform the same testing as *g3*.

    NOTE:
    Before you boot either *g21* or g2, you must disconnect the primary monitor's keyboard at the J-box and depress reset. This will force the output to the ASCII terminal on port two.

STEP 1: Booting the diagnostic.

By now you should have a good idea what type of data is displayed when you boot a diagnostic, so only the initial display of the diagnostic will be illustrated. Also illustrated are the different options you have to boot the diagnostics.

System	Media	Command	Comment
2400	Tape	`b mt0:g21`	Loads g21 from tape.
2400	Tape	`b mt0:g2`	Loads g2 from tape.
68020	Tape	`b ct0:g3`	Loads g3 from tape.
3020	Disk	`b md0c:/tmp/g3`	Loads g3 from disk /usr/tmp.
3030	Disk	`b si0f:/tmp/g3`	Loads g3 from disk /usr/tmp

The system that was used to created the output for this document was a 2400 Turbo with the following configuration: 8 meg of memory, 32 bit planes, an FP1 board, and all 14 macro chips on the GF2 board (Z-buffering was installed).

After the boot command was executed, the following appeared at the ASCII terminal when g3 was booted.

```
GF2/IP2/UC4/DC4  CONSOLE V1.9

  changes:
 't' no longer does 'bt'
EPROM version 1.4 kluge: > emb 3500000b   change to 0f
use ^C to escape macro loops!


!
```

If you are on a 2400, g21 must be booted first to allow you access to the same commands that this document illustrates. The following would appear after the boot command is executed:

```
GF2/PM2/UC4/DC4  CONSOLE V1.2

  changes:
use gt0-7 then gt9 Gv


!
```

In both of the initial messages you see the word *changes* followed by some information about command changes; it does not concern the F.E. What is important, is that the prompt (!) is displayed.

Once you have g3 (68020 based) or g21 (2400) loaded and the (!) is displayed, the flow is the same.

The tests are of the *go-nogo* type. If any of the following tests fail, correct the problem by replacing the GF2 board, cables, or reseating the board and/or it's cables.

Remember, other boards in the system could cause these tests to fail. Before replacing

GF2 you should run diagnostics against other boards in the system.

All the output of the examples show normal status (0 errors). If you experience errors, your output will be different.

NOTE:
Understanding how the GF2 board functions is not important, following the recommended order of initializing and testing the board is!

STEP 2: Initializing the GF2 board.

The first two commands (w and SI) initialize the GF2 board and **must** be executed before any testing occurs.

```
! w <return>

OK
all bkpts cleared.
!
```

```
! SI <return>    (upper case S and I)

findge head GA ..... tail GA found: 14  mask: 3fff
scratchsize: 0fff    version: 2.3
!
```

The program (SI) is informing you that it sees all 14 of the GF2 macro chips: the two Geometry Accelerators (head GA and tail GA) and 12 Geometry Engines (.....).

Two of the Geometry Engines are optional and your system may not have them; they are GE5 and GE10 and perform near and far clipping (Z-clipping).

STEP 3: Testing the Geometry Engines.

The next few commands test the Geometry Engines with various data patterns.

```
! cp <return>
```

```
0 errors in 482 words.
!
```

```
                ! cP <return>

0 errors in 196 words.
!
```

```
                ! ct <return>

0 errors in 993  words.
0 errors in 1047 words.
0 errors in 1101 words.
0 errors in 721  words.
0 errors in 728  words.  5 not installed
0 errors in 735  words.
0 errors in 751  words.
0 errors in 767  words.
0 errors in 755  words.
0 errors in 780  words.  10 not installed
0 errors in 700  words.
0 errors in 482  words.
```

The above output is what you would see for a GF2 with all 12 Geometry Engines. If your system only has 10, then the statements that are highlighted in **bold** print would be part of the output.

```
                ! cT <return>

0 errors in 1063 words.
0 errors in 1117 words.
0 errors in 1171 words.
0 errors in 791  words.
0 errors in 798  words.  5 not installed
0 errors in 805  words.
0 errors in 821  words.
0 errors in 837  words.
0 errors in 825  words.
0 errors in 850  words.  10 not installed
0 errors in 770  words.
0 errors in 552  words.
```

STEP 4: Testing the Geometry Accelerators.

The next two commands will test the Geometry Accelerators using various data patterns.

```
                ! cg <return>

0 errors in 588 words.
0 errors in 262 words.
0 errors in 262 words.
0 errors in 262 words.
0 errors in 262 words.
0 errors in 262 words.
0 errors in 262 words.
0 errors in 262 words.
0 errors in 262 words.
0 errors in 456 words.
0 errors in 428 words.
0 errors in 173 words.
0 errors in 278 words.
0 errors in 173 words.
0 errors in 133 words.

                ! cG <return>

0 errors in 658 words.
0 errors in 332 words.
0 errors in 332 words.
0 errors in 332 words.
0 errors in 332 words.
0 errors in 332 words.
0 errors in 332 words.
0 errors in 332 words.
0 errors in 332 words.
0 errors in 526 words.
0 errors in 498 words.
0 errors in 243 words.
0 errors in 348 words.
0 errors in 243 words.
0 errors in 203 words.
```

The previous tests checked the GF2 hardware without any data being output to the monitor. If failures occurred during one of them, chances are good that the problem is in the GF2 board.

The following tests display graphics at the monitor, so failures that occur could be other components in the system.

STEP 5: Booting *g2* if a 2400.

NOTE:
If you were on a 68020 based workstation, you could continue with the testing flow (jump to STEP 6), but if you were on a 2400, then you must boot the graphics test *g2* before continuing.

Execute the *quit (q)* command to return to the PROM monitor so that you can boot *g2*.

```
! q <return>

iris> b mt0:g2 <return>
```

Again the boot message is not shown, but the initial diagnostic output is:

```
GF2/PM2/UC4/DC4  CONSOLE  V1.7

  changes:
use ^c to escape macro loops!
gm tests: gm1 draws 2 red squares


!
```

STEP 6: Graphics Testing.

Depending on the type of monitor you have selected, from this point on, there are minor differences. These differences will be highlighted as the flow continues.

Lets again initialize the GF2 board.

```
! w <return>

OK
all bkpts cleared.
!

! SI <return>

findge head GA ..... tail GA found: 14  mask: 3fff
scratchsize: 0fff    version: 2.3
!
```

The next command is used only if your monitor is a 60 Hz monitor. You only need to execute it once. If your monitor is a 30 Hz, omit this command.

```
! m <return>

1-> 3020
!
```

The next command runs several tests and then draws two red squares connected by a green, red, blue and gray line.

```
! t <return>

Multibus: micro ram: micro test: done
OK
init: scratchsize: 0fff     version: 2.3
< testing 4096 words
<<< scratch test done.
< testing 4096 words
<<< scratch test done.
< testing 4096 words
<<< scratch test done.
< testing 4096 words
<<< scratch test done.
< testing 4096 words
```

```
<<< scratch test done.
draw: scratchsize: 0fff     version: 2.3
findge  head GA ..... tail GA  found: 14    mask: 3fff
byteswap: token test: OK
fifo test:        153 wds
GE test:

Tests done.  (485 vector words)
!
```

The next command draws blinking colored lines (solid and dashed) within 4 white boxes. The two boxes on the left side of the screen are smaller than the two boxes on the right side. When the test is complete, the boxes are completely white.

```
!  cd <return>

findge  head GA ..... tail GA  found: 14    mask:
3fff
scratchsize: 0fff     version: 2.3
```

The next series of commands will initialize the GF2 board, clear the screen and display Red and White squares that spiral in on themselves.

```
!  ii <return>

findge  head GA ..... tail GA  found: 14    mask:
3fff
scratchsize: 0fff     version: 2.3

!  gC <return>

!  gt <return>

!  Gbd <return>
```

The last test displays a box in the lower left corner of the screen that has colored bars moving through it from bottom to top. When the test is complete, the box is left a solid color.

```
                ! ii <return>

findge head GA ..... tail GA found: 14 mask:
3fff
scratchsize: 0fff version: 2.3

                ! gC <return>

                ! gt <return>

                ! Gd <return>
```

This completes the GF2 testing. To return to the PROM monitor, execute the **quit** command.

```
                ! q <return>
```

G3 Test Summary

The following section summarizes the function of each of the g3 diagnostic tests:

i Initializes the GE and FBC hardware.

w Writes the diagnostic's compiled-in microcode to the GF2 board RAM.

S A sequence of diagnostic commands which:

Runs a pass-through test

runs a full test checking the pipeline output

runs a full test checking the output of each GE

runs a full set of GA tests.

reports the number of GEs and GAs present.

cp Sets the GEs to *pass-through* mode then:

sends test vector data to each GE

reads back the result from each GE

notifies the user of the accumulated results.

cP Like *cp* but runs a complete functional test on the GE pipeline.

cT Performs a high-speed test by pre-loading the pipeline so that the GEs do not have to wait for input.

ct Runs a test data file through the pipeline.

cg Tests the GAs.

cG Runs a more stringent test (than cg) of the GA chips.

t Runs a sequence of tests that initialize the GE and FBC hardware, test RAM and draw lines and rectangles, generally performing a good functional test.

cd Sets all GEs to active mode (not passthrough) and runs the same tests as "t".

ii Initializes the FBC and GE pipeline and enables FIFO interrupts. Also initializes the Color Map on the DC4 board. Performed prior to the drawing tests.

gC Configures all GEs active.

gt Runs test files through the GE pipeline.

Gbd Draws a spiral of red and white squares.

Gd Draws a colored square in the lower-left corner of the display and a pattern of dots approximating an "L" shape near the center of the display.

SECTION 3: Test Items

None

Contents

BUGS10 1-1
- Introduction 1
- Objectives 1
- Sample Test Item 1
- Resources 1
- SECTION 1: The Boot Process 3
  - Boot Process Summary 4
- SECTION 2: Troubleshooting Hints 6
  - TROUBLESHOOTING HINTS: System will not Boot 6
  - TROUBLESHOOTING HINTS: ASCII Terminal Port 2 7
  - TROUBLESHOOTING HINTS: Minimum System to Run 8
  - TROUBLESHOOTING HINTS: Minimum System to Boot 9
  - TROUBLESHOOTING HINTS: Cannot Boot from Disk 10
  - TROUBLESHOOTING HINTS: PROM Monitor - page 1 11
  - TROUBLESHOOTING HINTS: PROM Monitor - page 2 12
  - TROUBLESHOOTING HINTS: Bit Plane Troubleshooting 13
  - TROUBLESHOOTING HINTS: Bit Plane Troubleshooting Map 15
- SECTION 3: Test Items 16

BUGS10

Introduction

The objective of the next three modules is to allow you to apply the knowledge learned and reinforce that learning by diagnosing induced failures on your workstation. This module provides you with some troubleshooting procedures that you will use to isolate failures.

Objectives

At the completion of this lesson, the participant will be able to:

- Recognize PROM monitor program configuration messages.
- Recognize IRIX boot process configuration messages.
- Match IRIX run-levels with their correct function.
- Recognize functions of IRIX configuration shell scripts and files.
- List the minimum board configuration required to:
  - boot IRIX
  - initialize the graphics subsystem

Sample Test Item

None

Resources

SECTION 1: The Boot Process

The boot process occurs in a sequential manner which this document follows.

Several notes must be made:

- The demonstration was produced on a 3030 or 3130 workstation using Revision 3.6 software.

- The discussion is generic and meant to help you understand what occurs during the boot process and look at the process as a *first line* diagnostic to be used to help isolate system failures.

  The boot process supplies diagnostic information to the person who understands what occurs during the process.

Assuming **initdefault** specified run level 2 or 3, or the user executed **multi**, the process continues:

- Upon issuance of a run level change, **init** re-reads the file **/etc/inittab** and reads the files: *rc.s0, brc, bcheckrc, rc* and any other files that match the run level.

  Commands contained in these configuration files are executed in an order specified by the **action** field of each entry in **inittab.** (i.e. sysinit, bootwait, wait)

  - **rc.s0** links */dev/systty* and */dev/syscon.*
  - **brc** removes the old */etc/mtab* file and checks for the system model number.

    If the model number is present in the file /etc/**model**, the process continues.

    Else, the user is asked to enter the model number.
  - **bcheckrc** checks for the date and prompts user to answer if the date is correct and if **FSCK** is to be run.

    If **FSCK** is to be run, the file */etc/fstab* is read to determine what filesystems should be checked.
  - **rc** creates a new mount table file */etc/mtab*, mounts the specified filesystems, removes locks and logs, starts daemons and anything else the user specified in the **rc** file.

- **Init** starts a *getty* process for each ASCII terminal specified in the file /etc/inittab.

- The **login** prompt is displayed on all on-line terminals and the system is in **multi-user** mode.

TROUBLESHOOTING HINTS: ASCII Terminal Port 2

- Remember, if the microcode or Kernel **cannot communicate with the primary console**, they look for an ASCII terminal on port 2 to output normal and error information.

- Know what the normal boot output should look like for each of the systems you maintain. A good idea is to make a copy of the auto-configuration output and keep it on site. (Be prepared)

- Know what the system acronyms mean.

  i.e.

  GF2 - Geometry Engine Board
  IP2 - Processor Board
  FBC - Frame Buffer Controller

TROUBLESHOOTING HINTS: Minimum System to Boot

- If you have a system with an FP1 and optional memory boards, the FP1 and all memory boards except the first 4 meg can be backed out and the system will still boot up. (You do not have to recable the boards)

- You can boot the system with GF2, UC4 and DC4 backed out if you have an ASCII terminal on port 2.

- You **cannot** boot if you have a hole on the multibus between IP2 and the Disk Controller.

  i.e. ENET board backed out.

- You could back out the Disk Controller and try booting over the network, but it will fail when the disk drive is accessed, but at least it will boot to a point.

TROUBLESHOOTING HINTS: PROM Monitor - page 1

If you are experiencing system problems and want to verify if memory can be written into and read from, the PROM Monitor offers several commands that allow basic testing of the memory system.

The commands available are listed below with descriptions as to their functions:

fm{b|w|l} ADDR VALUE [INCR] [CNT]
fill memory as byte, word or long stating at ADDR, with initial VALUE, incrementing VALUE by INCR for CNT times. (INCR defaults to 0; CNT defaults to 1)

dm{b|w|l} ADDR [CNT]
display memory as byte, word or long. (CNT defaults to 1)

em{b|w|l} ADDR
edit memory interactively as byte, word or long.

Memory is addressed with a 24 bit address. The following table shows the address ranges for each meg and the card that contains the locations.

MEG	ADDRESS	CARD
1	000000-0FFFFF	1
2	100000-1FFFFF	1
3	200000-2FFFFF	1
4	300000-3FFFFF	1
5	400000-4FFFFF	2
6	500000-5FFFFF	2
7	600000-6FFFFF	2
8	700000-7FFFFF	2
9	800000-8FFFFF	3
10	900000-9FFFFF	3
11	A00000-AFFFFF	3
12	B00000-BFFFFF	3
13	C00000-CFFFFF	4
14	D00000-DFFFFF	4
15	E00000-EFFFFF	4
16	F00000-FFFFFF	4

2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	7	6	5	4	3	2	1	0	Address Bit
8 m	4 m	2 m	1 m	5 1 2 k	2 1 5 k	1 2 6 k	6 8 4 k	3 2 k	1 6 k	8 k	4 k	2 k	1 k	1 k	5 2 6	1 2 8	6 4	3 2	1 6	8	4	2	1	Address Bit Weight

TROUBLESHOOTING HINTS: Bit Plane Troubleshooting

1. Login as **guest.**

2. cd /usr/people/mexdemos/bin.

3. Execute **mex.**

4. Execute **showmap** (draw the largest possible window).

5. **Push** showmap to the background.

6. Execute **cedit** (sweep a small window in upper right corner).

7. cd /usr/people/mexdemos.

8. Execute **interp** (sweep a small window below **cedit** window).

9. Execute **demomakemap.**

10. **Push** the console window to the background.

11. Using **cedit**, make the third square (bottom line left) black.
(Place the cursor over the square, and press the left mouse button.)

12. Using **cedit**, make the last square (top row right) black.
(Place the cursor over the square, and press the left mouse button.)

13. Using **interp**, mark the third and last squares (left mouse button).
(Place the cursor over the square and press the left mouse button.)

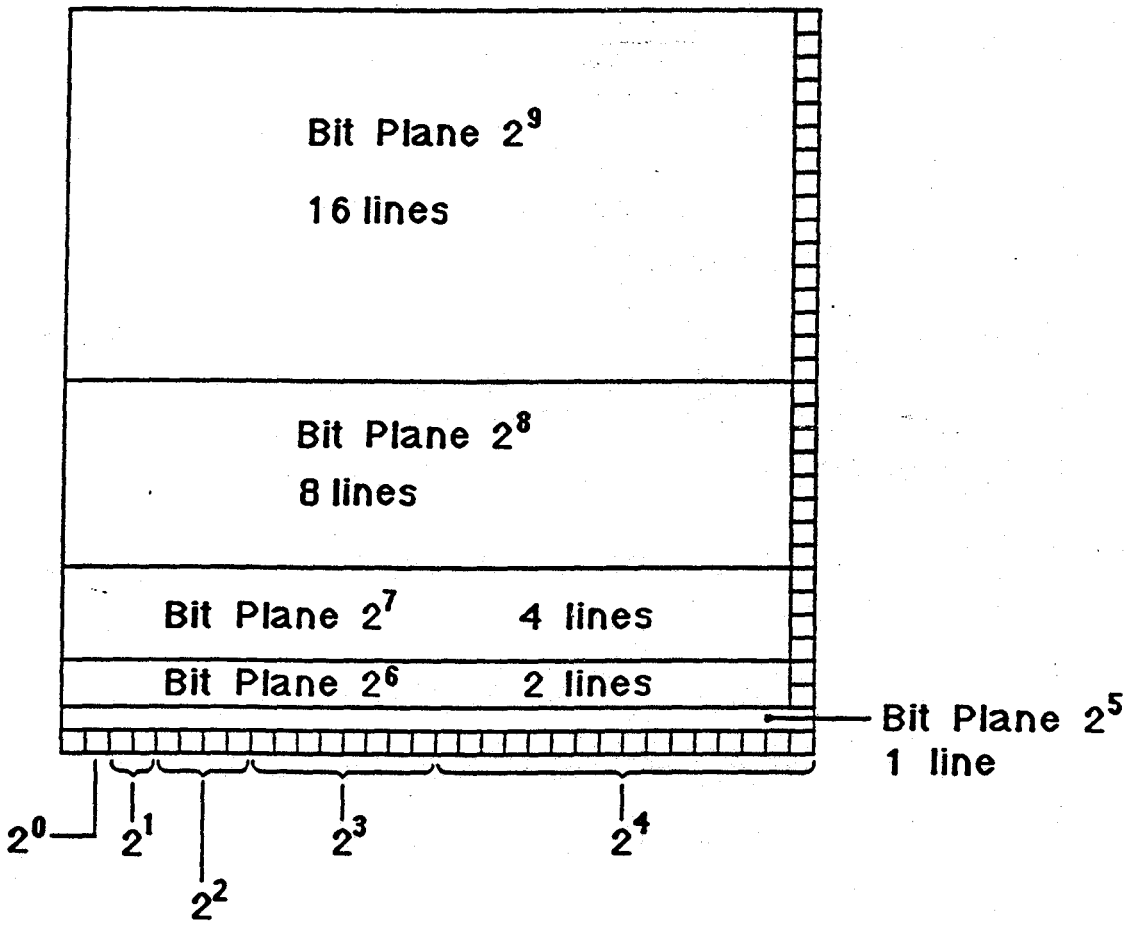
14. Using **interp**, interpolate the color range between the third and last square. (Press the middle mouse button).

NOTE: If any bit plane is broken (hot bits) you will see spots.

15. You can repeat steps 11-14 using white to check for cold bits.

TROUBLESHOOTING HINTS: Bit Plane Troubleshooting Map

If you can detect areas of the screen that are not correct using the procedure on the previous page, then use this chart to determine what bit plane is defective.



Pop up menus are bit planes 2^{10} and 2^{11}