# IRIS Terminal Guide

*Version 1.3*

Silicon Graphics, Inc.
Mountain View
California 94043

**Documentation:**
Daniel Sears
Marcia Allen
Robin Bristow
Diane Wilford

**Drawings:**
Annette Whelan

5001-020-001-3

# 1. Introduction

This document explains how to install and operate an IRIS Terminal. It contains step-by-step procedures for installing the components that make up an IRIS Terminal system. Read this document carefully before installing an IRIS Terminal.

The IRIS Terminal can be installed in different host operating system environments. The supported environments include the UNIX[1] and the VAX/VMS[2] operating systems. Instructions for installing the host-resident software in a UNIX environment are in Section 5. Instructions for installing this software in a VAX/VMS environment are in Appendix D. Section 6 contains operation procedures (e.g. boot, Monitor adjustment, shutdown and diagnostics) that are host-independent. UNIX-specific instructions for compiling and running demonstration programs are in Section 6.7. VAX/VMS-specific instructions are in Appendix D. The *IRIS User's Guide* documents the host-resident software that allows a programmer to write application programs with the IRIS Terminal.

The IRIS Terminal components are delivered assembled and ready for connection with cables provided in the delivery cartons. In addition, a magnetic tape is included with host software that resides on the host computer and terminal software that is downloaded into the IRIS Terminal each time it is booted.

Silicon Graphics provides a comprehensive product support and maintenance program for the IRIS Terminal. For further information, the toll-free numbers for Silicon Graphics Customer Service are:

| **Silicon Graphics Customer Service** | |
|---|---|
| (800) 252-0222 | North America (except California) |
| (800) 345-0222 | California |

1. UNIX is a trademark of Bell Telephone Laboratories.
2. VAX/VMS is a trademark of Digital Equipment Corporation.

# 2. Unpacking the IRIS Terminal Components

The IRIS Terminal system is shipped in two reinforced cardboard cartons. One contains the Electronics Cabinet and the other contains the Monitor and other components. Each component is delivered assembled and ready for connection with the cables provided in the IRIS Terminal delivery cartons. If additional equipment or spare parts are ordered, they will be shipped in additional cartons.

Before installation, the delivery cartons should be inspected for damage. If any of the cartons or their contents appear damaged, contact the carrier and Silicon Graphics Customer Service (see Section 1). After inspection, move the cartons to the installation site. See Table 2-1 for a list of guidelines for site selection. Although site selection is the customer's responsibility, Silicon Graphics representatives will provide consulting services upon request.

1. Inspect the delivery cartons for damage.

   WARNING: The delivery cartons should be moved on a pallet jack or cart capable of supporting 200 lbs. If they must be lifted, two strong people are needed.

   WARNING: Do not turn the delivery cartons on edge.

2. Move the cartons to the installation site.
3. Cut the plastic straps on the brown carton.
4. Cut the tape that seals the top of the carton, open the carton and remove the tray containing the Keyboard, Mouse, cables and other equipment.
5. Remove the foam spacers covering the Monitor.
6. Remove the carton by lifting it up and off the base pallet.
7. The Monitor is shipped inside a large plastic bag. Remove the Monitor from the bag and place it on the surface where it will be used.

   WARNING: Do not attempt to pick up the Monitor by the white plastic inserts on the sides.

8. Check for damage to the Monitor.

| Category | IRIS 1000 | IRIS 1200 |
|---|---|---|
| Temperature | 50 - 86F° (operating) | " |
| | 40 - 176F° (non-operating) | " |
| Relative humidity | 40 - 80% | " |
| Minimum clearance | 3" all sides | " |
| Monitor desk space | 36" width | " |
| | 30" length | " |
| Cabinet floor space | 16" width | 24" width |
| | 33" length | 33" length |
| | 21" height | 29" height |
| Power | 115 volts | " |
| | 60 hertz | " |
| | single phase | " |
| | 10 amps | 15 amps |
| | two wires + ground | " |
| Power consumption | 470 watts maximum | 750 watts |
| | 640 KVA | 1040 KVA |
| Heat dissipation | 1600 BTU/hour | 2560 BTU/hour |
| Card slots | 10 | 20 |

Table 2-1: IRIS Terminal Environmental Specifications

9. Compare the equipment included in the tray with the list in Section 3. If any parts appear to be missing, contact Silicon Graphics Customer Service (see Section 1).
10. Cut the plastic straps on the white carton.
11. Open the top of the carton and remove the foam cap.
12. Lift the carton off of the Cabinet.
13. Lift the Cabinet off the base pallet and set it on the floor.
14. Remove the foam spacers.
15. The Cabinet is shipped inside a large plastic bag. Remove the Cabinet from the bag.
16. Check for damage to the Cabinet.
17. Remove the front panel from the Cabinet and check that the boards inside are firmly attached.
18. The Cabinet has four castors on its base that allow it to be rolled across a surface. Roll it to the location where it will be used.

# 3. IRIS Terminal Specifications

| Component | Height | Width | Length | Weight |
|---|---|---|---|---|
| IRIS 1000 Cabinet | 21.0" | 10.0" | 27.0" | 100.0 lbs |
| IRIS 1200 Cabinet | 29.0" | 18.0" | 27.0" | 200.0 lbs |
| Monitor | 18.0" | 20.0" | 21.0" | 97.0 lbs |
| Keyboard | 1.5" | 19.0" | 8.5" | 3.0 lbs |
| Mouse | 1.0" | 2.0" | 3.0" | 0.5 lbs |
| Transceiver | 2.0" | 7.0" | 4.0" | 0.8 lbs |

Table 3-1: IRIS Terminal Component Specifications

## 3.1 Hardware Components

Each IRIS Terminal system has four hardware components (see Figure 3-1).

- The *Electronics Cabinet* is a floor-standing unit with a 10-slot (IRIS 1000) or 20-slot (IRIS 1200) backplane and a power supply. The Cabinet uses forced air cooling and is mounted on casters.
- The *Monitor* is a high-resolution 19-inch color monitor.
- The *Keyboard* is an 83-key up-down encoded keyboard.
- The *Mouse* is a 3-button mouse.

Two hardware component options are the Dial Box and Switch Box.

- The *Dial Box* (optional) has 8 independently programmable valuators for sending analog information to an application program for the IRIS Terminal.
- The *Switch Box* (optional) has 32 independently programmable switches and 32 independently programmable LED indicator lights. An 8 character LED display gives status information for the Dial Box and the Switch Box.

## 3.2 Ethernet Equipment

The IRIS Terminal can be connected to an Ethernet local area network with an Ethernet transceiver and drop cable.

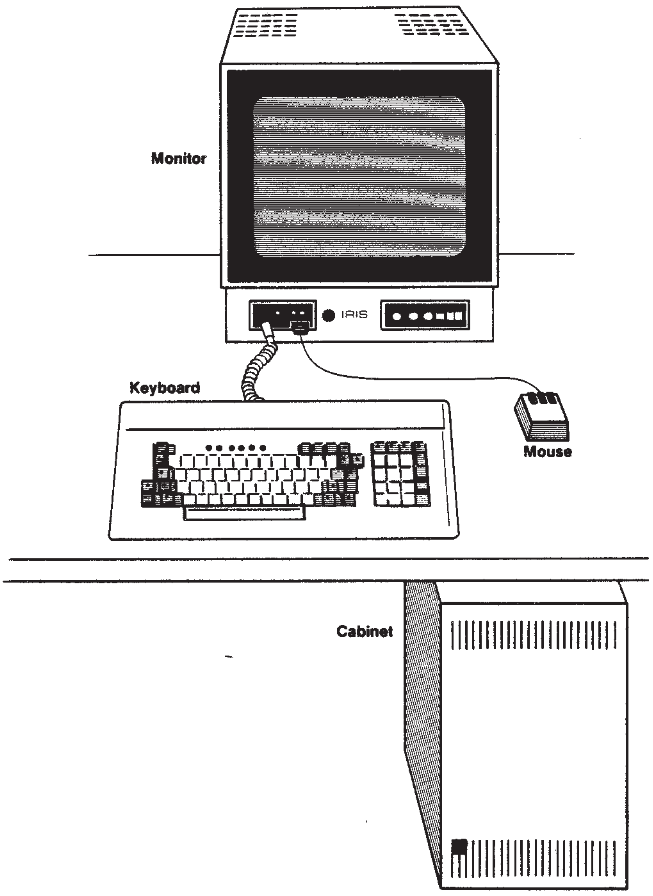

Figure 3-1: IRIS Terminal System

- The *Ethernet Transceiver* routes messages between the IRIS Terminal and the Ethernet.
- 1 15-conductor drop cable connects the Cabinet to an Ethernet transceiver.

## 3.3 Cables

Each IRIS Terminal is supplied with a cable set tor connecting the IRIS Terminal components.

- 4 bundled, color-coded, coaxial video cables connect the video output of the Cabinet to the Monitor.
- 1 25-conductor control cable connects the Cabinet to the Monitor. This cable sends and receives signals between the Cabinet and the Mouse. Keyboard and `Reset` button on the IRIS Control Panel.
- 2 10-foot, 3-wire, grounded AC power cables provide power for the Monitor and Cabinet.
- 1 37-pin flat cable (optional) connects the Dial Box to the Switch Box.
- 1 9-pin cable (optional) connects the Switch Box to `Port 2` on the Cabinet I/O Panel.
- 1 3-wire AC power cable (optional) provides power for the Dial Box and the Switch Box.

## 3.4 Monitor

The Monitor has two control panels, the IRIS Control Panel on the front left and the Monitor Control Panel on the front right. On the back of the Monitor are several ports for receiving video signals, a power socket and a control cable port.

### IRIS Control Panel

The IRIS Control Panel has two ports for connecting the Keyboard and Mouse to the Monitor, a `Reset` button and two indicator lights (see Figure 3-2).

- 1 DIN socket labeled `Keyboard` is a port for connecting the Keyboard to the Monitor.
- 1 slide-locking D socket labeled `Mouse` is a port for connecting the Mouse to the Monitor.
- 1 LED labeled `Power` indicates that power for the Cabinet is switched on.
- 1 LED labeled `Halt` indicates that the processor is stopped.

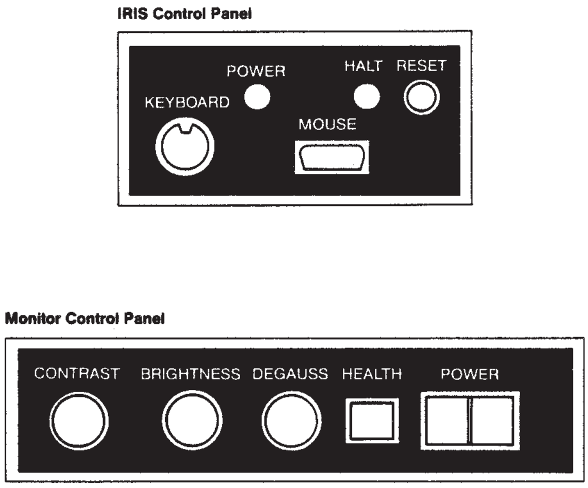

Figure 3-2: IRIS Control Panel and Monitor Control Panel

- 1 button labeled `Reset` resets the IRIS Terminal. The IRIS Terminal must be rebooted each time the `Reset` button is pushed.

**Monitor Control Panel**

The Monitor Control Panel has several features for adjusting the Monitor (see Section 6.6) and a `Power` switch for controlling power for the Monitor (see Figure 3-2).

- 1 knob labeled `Brightness` adjusts the white and black levels equally. Turning this knob clockwise increases the Monitor's brightness.
- 1 knob labeled `Contrast` adjusts the white levels. Turning this knob clockwise increases the Monitor's contrast.
- 1 button labeled `Degauss` demagnetizes the Monitor screen.
- 1 light labeled `Health` indicates that power for the Monitor is switched on and most of the Monitor is operating properly.
- 1 switch labeled `Power` controls power for the Monitor.

**Monitor Back Panel**

The Monitor Back Panel has several ports that connect the Monitor to the Cabinet (see Figure 3-3).

- 2 BNC sockets labeled `Ext Sync` are used for the video sync connection from the Cabinet.
- 2 BNC sockets labeled `V D` are not used.
- 2 BNC sockets labeled `R` receive the red video signal from the Cabinet.
- 2 BNC sockets labeled `G` receive the green video signal from the Cabinet.
- 2 BNC sockets labeled `B` receive the blue video signal from the Cabinet.
- 1 5-amp fuse.
- 1 10-amp 100/120 volt power plug provides power for the Monitor from the Cabinet.
- 1 25-pin plug connects a control cable from the Cabinet to the Monitor.

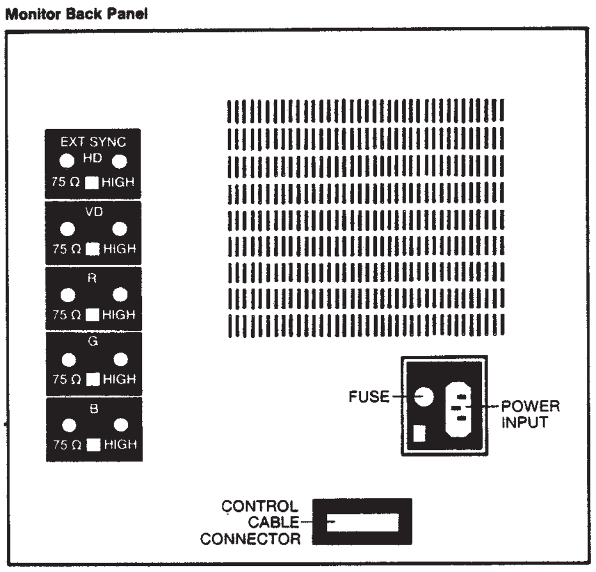

Figure 3-3: Monitor Back Panel

## 3.5 Cabinet

There are two control panels on the back of the IRIS 1000 and IRIS 1200: an I/O Panel and a Power Panel. A `Power` switch controls power for the IRIS Terminal system.

### Power Switch

The `Power` switch is located differently on the IRIS 1000 Terminal and the IRIS 1200 Terminal.

- The `Power` switch for the IRIS 1000 Terminal is located on the front lower-left corner of the Cabinet.
- The `Power` switch for the IRIS 1200 Terminal is located inside the floppy disk drive slot on the front upper-left corner of the Cabinet.

### Cabinet I/O Panel

The Cabinet I/O Panel has ports for connecting the Cabinet to the Monitor and a host computer (see Figures 3-4a and 3-4b) and several control and indicator features.

- `Port 1` is the receptacle for the control cable that is connected between the Monitor Back Panel and the Cabinet I/O Panel.
- `Port 2` is the receptacle for the RS-232 or RS-423 serial line to a host computer.
- `Port 3` and `Port 4` are reserved.
- 1 15-pin D socket labeled `Ethernet` connects the IRIS Terminal to an Ethernet drop cable.
- 1 flat cable connector labeled `Floppy Drive` connects a floppy disk to the IRIS 1000.

  NOTE: the IRIS 1200 Terminal may have a boot floppy drive installed inside the Cabinet near the power switch.

- 1 BNC socket labeled `Sync` is a port for the video sync connection from the Cabinet to the Monitor through a coaxial cable.
- 1 BNC socket labeled `Red` is a port for transmitting the red video signal from the Cabinet to the Monitor through a coaxial cable.
- 1 BNC socket labeled `Green` is a port for transmitting the green video signal from the Cabinet to the Monitor through a coaxial cable.
- 1 BNC socket labeled `Blue` is a port for transmitting the blue video signal from the Cabinet to the Monitor through a coaxial cable.

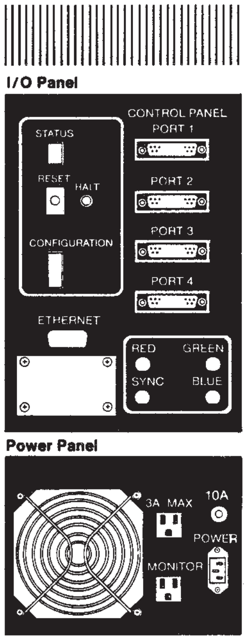

Figure 3-4a: IRIS 1000 Cabinet Back Panel

- 1 `Reset` button is located on the Cabinet I/O Panel. Pressing this button resets the processor and this in turn resets the rest of the system. After the `Reset` button has been pressed, the IRIS Terminal must be rebooted. This button and the `Halt` light correspond to the `Reset` button and `Halt` light on the IRIS Control Panel on the Monitor. Either button may be used to halt the system.
- 1 `Halt` LED indicates that the processor is stopped.
- 1 alphanumeric diagnostic LED labeled `Status` on the Cabinet I/O Panel indicates system status and displays the results of startup diagnostics (see Section 6.9 and Appendix A).
- A series of nine DIP switches labeled `Configuration` is located on the Cabinet I/O Panel. These switches control the IRIS Terminal's serial line baud rate, startup diagnostics and boot environment (see Appendix A).

**Cabinet Power Panel**

The Cabinet Power Panel has two power outlets and a power plug (see Figure 3-4a and 3-4b).

- 1 male 3-pin input power plug labeled `Power` provides power for the IRIS Terminal system.
- 1 switched 2-amp power outlet labeled `Monitor` provides power for the Monitor.
- 1 unswitched 3-amp convenience outlet labeled `3A Max` provides power for peripheral equipment.
- 1 10-amp circuit protector (IRIS 1000).
- 1 15-amp circuit protector (IRIS 1200).

**Power Switch**

The `Power` switch located on the front of the Cabinet controls power for the Cabinet and the Monitor. It does not control the power for any auxiliary equipment connected to the Cabinet through the convenience outlet.

## 3.6 Software Specification

The UNIX distribution tape contains host software that resides on the host computer and terminal software that is downloaded into the IRIS Terminal each time it is booted. See Appendix D for a description of the VAX/VMS software distribution tape.

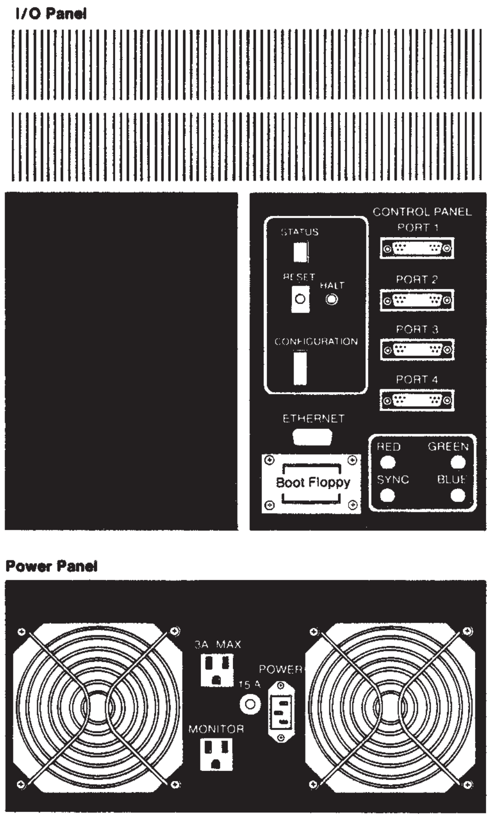

Figure 3-4b: IRIS 1200 Cabinet Back Panel

The software on the distribution tape is divided into four directories: *boot*, *c*, *f77* and *man*.

*boot*
- *iris* is a binary file containing software that is downloaded into the IRIS Terminal during the boot procedure.
- Several files with *.dsk* suffixes are binary files containing various demonstration programs. These files can be downloaded into the IRIS Terminal (see Section 6.7).

*c*
- *Makefile* is a make description file for compiling the C version of the Remote Graphics Library (*libgl.a*), the serial download program (*dliris*) and some demonstration programs.
- *termcap* is a file containing a TERMCAP description tor the IRIS Terminal.
- *device.h* is a C include file with symbolic name definitions for devices, e.g. mouse and keyboard buttons. Inclusion of this file in programs intended for the IRIS Terminal is optional.
- *get.h* is a C include file with definitions for values returned by the Graphics Library get commands. Inclusion of this file in programs intended for the IRIS Terminal is optional.
- *gl.h* is a C include file with default values for colors, screen boundaries, etc. for the Graphics Library. This file should be included with every program intended for the IRIS Terminal.
- *boot* is a Bourne Shell script that is used by the *Makefile* to identify the default directory for the IRIS Terminal boot programs.
- *dliris.c*, *irisboot.h* and *remprom.h* are source and include files for *dliris*, the program used to serially download software into the IRIS Terminal.
- *io.c*, *lib.c*, *rpc.h* and *Xnsioctl.h* are source and include files for generating the C version of *libgl.a*, the Remote Graphics Library.
- *sqiral.c* is a C program that draws a square spiral (see Section 6.7).
- *track.c* is a C program that displays a box which can be moved by the mouse (see Section 6.7).

*f77*
- *Makefile* is a make description file for compiling the FORTRAN version of the Remote Graphics Library (*libgl.a*), the serial download program (*dliris*) and some demonstration programs.
- *device.h* is a FORTRAN include file with symbolic name definitions for devices, e.g. mouse and keyboard buttons. Inclusion of this file in programs intended for the IRIS Terminal is optional.

- *get.h* is a FORTRAN include tile with definitions for values returned by the Graphics Library get commands. Inclusion of this file in programs intended for the IRIS Terminal is optional.
- *gl.h* is a FORTRAN include tile with default values for colors, screen boundaries, etc. for the Graphics Library. It should be included with every program intended for the IRIS Terminal
- *io.c*, *lib.f*, *rpc.h* and *Xnsioctl.h* are source and include files for generating the FORTRAN version of *libgl.a*, the Remote Graphics Library.
- *curve.f* is a FORTRAN program that draws curves.
- *float.f* is a FORTRAN program that fills the IRIS Terminal screen with randomly colored pixels.
- *planets.f* is a FORTRAN program that models a solar system.
- *planets.doc* is a document describing the *planets* program.
- *sqiral.f* is a FORTRAN rogram that draws a square spiral (see Section 6.7).
- *star.f* is a FORTRAN program that displays a star.
- *track.f* is a FORTRAN program that displays a box that can be moved around the screen with the mouse (see Section 6.7).

*man*

- *man1* is a directory that contains a manual entry for *dliris*.
- *man3* is a directory that contains manual entries for the commands in the Graphics Library.

## 3.7 Documentation

The IRIS Terminal is delivered with a complete set of documentation.

- The *IRIS Terminal Guide* (this booklet) explains how to install and test an IRIS Terminal
- The *IRIS User's Guide* describes the *IRIS Graphics Library* and how to write application programs for the IRIS Terminal. It is divided into three parts. The *System Overview* is a general introduction to the IRIS Terminal's architecture and terminal software. The *IRIS Graphics Library* describes how to write graphics applications programs for the IRIS Terminal. The *Reference Manual* describes each command in the *IRIS Graphics Library*.
- The C and FORTRAN editions of the *IRIS Graphics Library* are quick reference cards with overviews of each command in the Graphics Library.

# 4. Hardware Installation

This section describes how to install and connect the components that make up an IRIS Terminal system (see Figures 4-1a and 4-1b). Prior to installation, each component should be unpacked and placed near its final location. Since the IRIS Terminal components are delivered assembled, they only need to be connected with the cables provided in the delivery cartons.

> WARNING: *Do not* connect the IRIS Terminal to an external power source until each cable has been connected and checked.

## 4.1 Keyboard to Monitor Connection

The Keyboard cable is connected to the IRIS Control Panel located on the lower left front of the Monitor (see Figure 3-2).

1. Connect the DIN plug on the Keyboard cable to the DIN socket labeled `Keyboard` on the IRIS Control Panel.

## 4.2 Mouse to Monitor Connection

The Mouse cable is connected to the IRIS Control Panel located on the lower left front of the Monitor (see Figure 3-2).

1. Connect the slide-locking D socket on the Mouse cable to the D plug labeled `Monitor` on the IRIS Control Panel.

## 4.3 Monitor to Cabinet Video Connections

The color-coded bundle of coaxial video cables is connected between the Cabinet I/O Panel and the Monitor Back Panel (see Figures 3-3, 3-4a, 3-4b, 4-1a and 4-1b).

1. For single Monitor operation, set *all* of the input impedance switches to the `75 Ω` position.

   If several Monitors are connected in a series (daisy chain), set the input impedance switches to the `High` position for all but the last Monitor, which should be set to the `75 Ω` position.

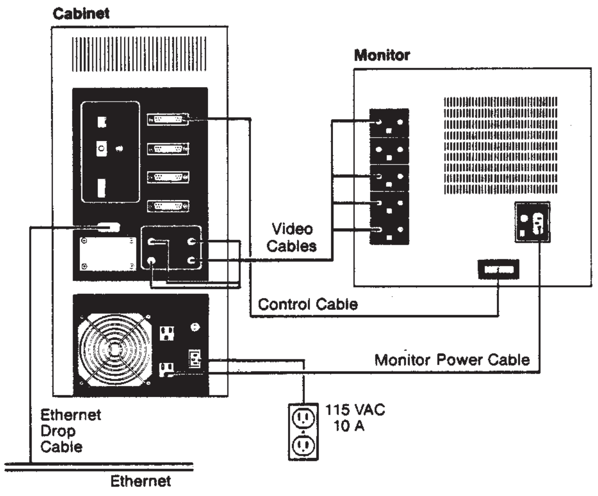

Figure 4-1a: IRIS 1000 Monitor to Cabinet Connections

2. Connect each cable end to an input socket on the Monitor Back Panel. Since they are identical, either socket can be used.
3. Push each cable into its connector and rotate its lock into place.
4. Connect the other end of each color-coded cable to the corresponding color output socket on the Cabinet I/O Panel.
5. Push the cable into the connector and rotate the lock into place.

## 4.4 Monitor to Cabinet Control Cable Connection

The Control Cable connects the Cabinet and the Monitor.

1. Connect the female end of the Control Cable to the 25-pin socket on the Monitor Back Panel.
2. Connect the male end of the Control Cable to `Port 1` on the Cabinet I/O Panel.
3. Fix the Control Cable into place by tightening the captive screws on each side of both plugs.

## 4.5 Monitor to Cabinet AC Power Cable Connection

The Monitor power outlet is located on the Cabinet Power Panel (see Figures 3-4a and 3-4b). This switched AC outlet is controlled by the Cabinet `Power` switch.

1. Connect the female end of the AC power cable to the `Input` power socket on the Monitor Back Panel.
2. Connect the male end of the Monitor power cable to the AC outlet labeled `Monitor` on the Cabinet Power Panel.

## 4.6 IRIS Terminal to Host Serial Line Connection

The IRIS Terminal can be connected to a host computer through a serial line (see Appendix E for a specification of the IRIS Terminal RS-232 interface).

1. Connect an RS-232 or RS-423 cable from the host computer to `Port 2` on the Cabinet I/O Panel.
2. Fix the cable into place with the captive screws at each end.
3. Set the `Serial Line Baud Rate` configuration switches (switches 2 and 3) on the Cabinet I/O Panel to an appropriate baud rate (see Section 6.2 and Appendix A). The serial line speed affects the time required to download the terminal software. At least 9600 baud is recommended.

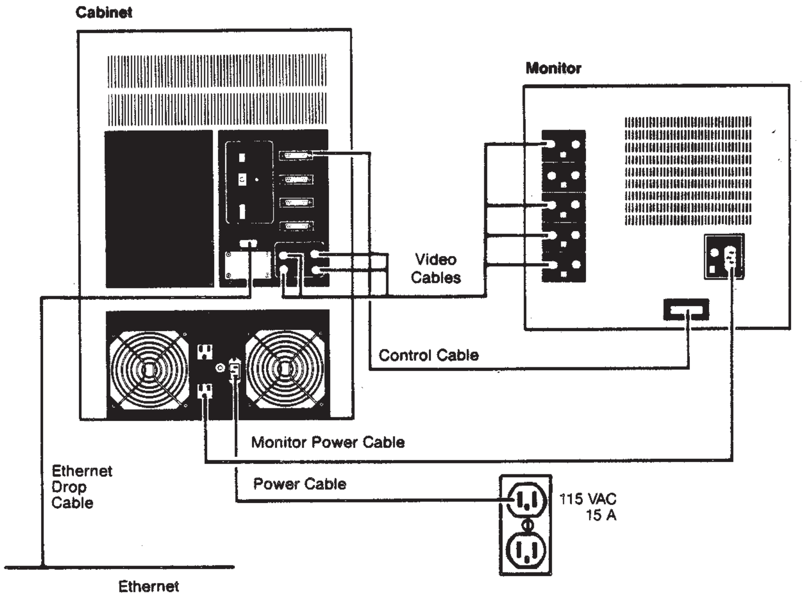

Figure 4-1b: IRIS 1200 Monitor to Cabinet Connections

## 4.7 IRIS Terminal to Ethernet Connection

The IRIS Terminal can communicate through an Ethernet local area network. The IRIS Terminal can be connected to an Ethernet local area network while the network is operating.

1. Select an appropriate tap point on the Ethernet coaxial cable.

   NOTE: Approved Ethernet coaxial cable is marked with rings at 8.2 foot intervals. Transceivers should be placed at these rings to minimize the chance of transceiver reflections with phase angles that add and cause transmission errors.

2. Tap into the Ethernet cable (instructions are included with each transceiver).
3. Connect the transceiver to the Ethernet cable.
4. Connect the male end of the drop cable to the Ethernet port on the Cabinet I/O Panel.
5. Connect the female end of the drop cable to the transceiver.

## 4.8 Cabinet to Floppy Disk Connection (IRIS 1000)

On the IRIS 1000 Terminal, a floppy disk drive can be connected to the Cabinet with a 34-pin communications cable and a power cable.

1. Unpack the floppy disk unit.
2. Plug the floppy disk power cable into the `3A Max` convenience outlet on the Cabinet Power Panel. (See Figure 3-4b.)
3. Plug the 34-pin blue ribbon communications cable from the floppy disk drive into the 34-pin socket on the Cabinet I/O Panel.

## 4.9 Cabinet AC Power Connection

The Cabinet power socket is located on the Cabinet Power Panel.

CAUTION: Do not connect the IRIS Terminal to a switched power outlet.

1. Connect the female end of the AC power cable to the power socket on the Cabinet Power Panel (see Figure 3-4a and 3-4b).
2. Connect the male end of the Cabinet power cable to an appropriate outlet. See Table 2-1 for a specification of the power requirements of the IRIS Terminal.

## 4.10 Cabinet to Dial and Switch Box Connection (optional)

The IRIS Terminal can be connected to an optional Dial and Switch Box for sending information to an application program for the IRIS Terminal.

1. Connect the 37-pin flat cable from the port on the Dial Box to the bottom left port on the Switch Box.
2. Connect the 9-pin cable from the top left port on the Switch Box to `Port 2` on the Cabinet I/O Panel.
3. Plug the 3-wire AC power cable on the Switch Box into the `3A Max` convenience outlet on the Cabinet Power Panel (See Figure 3-4b).

# 5. Software Installation

This section contains instructions for installing host-resident software in a UNIX environment. Instructions for installing host software in a VAX/VMS environment are in Appendix D.

The distribution software for the IRIS Terminal is delivered on a 1600 bpi magnetic tape in either *tar* or *cpio* format.[1] This tape includes host software that resides on the host computer and terminal software that must be downloaded from the host computer into the IRIS Terminal each time the IRIS Terminal is booted (see Section 3.6 for a description of the software on the distribution tape).

In preparation for the IRIS Terminal installation, the following manuals are useful.

[1] *UNIX Programmer's Manual (4.2BSD)*
University of California, Berkeley;
August 1983.

[2] *Make - A Program for Maintaining Computer Programs*
S. I. Feldman,
Bell Laboratories;
August 1978.

The distribution software can be stored anywhere on the host file system. For purposes of explanation, the examples below assume that the distribution software will be installed in a directory called */usr/iris*.

1. Login to UNIX.
2. Create a directory to hold the distribution software.

   ```
   $ mkdir /usr/iris
   ```
3. Change directories to the new directory */usr/iris*.

   ```
   $ cd /usr/iris
   ```

1. See Volume 1 of the *UNIX Programmer's Manual*.

4. Load the distribution tape onto a tape drive and read the distribution software into the new directory.

   ```
   $ tar x      or      $ cpio -i < /dev/rmt1
   ```

   A label indicates the format of the software on the distribution tape.

5. Change directories to */usr/iris/c*.

   ```
   $ cd /usr/iris/c
   ```

6. Add the TERMCAP entry in *c/termcap* for the IRIS Terminal to the terminal description data base in */etc/termcap*.

7. Define the Shell variable *IRISBOOT* to be the directory that contains the IRIS terminal program *iris*.

   ```
   $ IRISBOOT=/usr/iris/boot
   $ export IRISBOOT
   ```

8. Compile the Remote Graphics Library (*libgl.a*), the serial download program (*dliris*) and some demonstration programs.

   ```
   $ make all
   ```

The procedure for compiling the FORTRAN versions of these files is similar to the procedure above. The relevant files are in */usr/iris/f77*.

# 6. Operation

This section contains instructions for operating the IRIS Terminal. Most of these procedures are host-independent. Section 6.2 describes how to boot the IRIS Terminal over a serial line in a UNIX environment. Appendix D has a similar procedure for a VAX/VMS environment. Section 6.8 has information on compiling and running demonstration programs in a UNIX environment. For a VAX/VMS environment, see Appendix D.

After connecting the IRIS Terminal components and installing the distribution software on the host system, the IRIS Terminal can be booted. This involves setting some configuration switches, turning on the power and (in some cases) explicitly downloading the terminal software. The following four sections are mutually exclusive.

## 6.1 Ethernet

If the `Boot Environment` configuration switches are set to "00100", then the IRIS Terminal will boot automatically over the Ethernet.

1. Connect the Ethernet drop cable to the Ethernet port on the Cabinet I/O Panel (see Section 4.7).
2. Set the `Boot Environment` configuration switches (switches 5 through 9) to "00100" where "0" means `Closed` and "1" means `Open` (see Appendix A).
3. Set the Monitor `Power` switch located on the Monitor Control Panel to the `On` position.
4. Set the Cabinet `Power` switch located on the front of the Cabinet to the `On` position.
5. The IRIS Terminal initially displays a series of dots to indicate that downloading is underway. After about fifteen seconds, the IRIS Terminal screen clears and a rectangle in the bottom of the screen changes colors. This is the window for the VT-52 terminal emulator. The IRIS Terminal will prompt for the name of the host computer.

   ```
   Connect to what host?
   ```

   Reply with the name of the host computer (i.e. */bin/hostname*).

6. A login prompt will appear in the upper left-hand corner of this window.

   ```
   login:
   ```

   To use the IRIS Terminal, login to an account.

To reboot the IRIS Terminal, press the `Reset` button on the IRIS Control Panel and follow the boot procedures outlined above.

## 6.2 Serial Line

If the `Boot Environment` configuration switches are set to '01100', then the IRIS Terminal will emulate a simple ASCII terminal interfaced to the host computer. The user must then login and download the terminal software.

1. Connect a serial line from the host computer to `Port 2` on the Cabinet I/O Panel (see Section 4.6).
2. Set the `Boot Environment` configuration switches (switches 5 through 9) to "01100" where "0" means `Closed` and "1" means `Open` (see Appendix A).
3. Set the `Serial Line Baud Rate` configuration switches (switches 2 and 3) to a suitable baud rate (see Section 4.6 and Appendix A). For example, to set the baud rate to 9600 baud, set these switches to "11" where "0" means `Closed` and "1" means `Open`.
4. Set the Monitor `Power` switch located on the Monitor Control Panel to the `On` position.
5. Set the Cabinet `Power` switch located on the front of the Cabinet to the `On` position.
6. The IRIS Terminal will provide a simple ASCII terminal interface to the host computer and display a login prompt. Login to an account with permission to use the IRIS Terminal.

   ```
   login:
   ```

7. The command *dliris* downloads files into the IRIS Terminal. By default it downloads the terminal software (*iris*). Section 6.7 has other examples of downloading files with *dliris*.

   ```
   $ /usr/iris/c/dliris iris
   ```

   The IRIS Terminal initially displays a series of dots to show that downloading is underway. The time necessary for downloading the terminal software is dependent on the baud rate of the serial line to the host computer and the load on the host computer. For a 9600 baud connection (recommended), this time is about 5 minutes.

8. After downloading is complete, the IRIS Terminal screen will clear and a rectangle in the bottom of the screen will change colors. This

is the window for the VT-52 terminal emulator. fhe IRIS Terminal will prompt for the name of the host computer.

```
Connect to what host?
```

Reply with *serial*. The window will be redrawn and a prompt will appear in the upper left-hand corner.

To reboot the IRIS Terminal, press the `Reset` button on the IRIS Control Panel and repeat the boot procedures outlined above.

## 6.3 PROM Monitor

If the `Boot Environment` configuration switch is set to "01000", the IRIS Terminal will be under the control of the PROM Monitor when the power is turned on. The PROM Monitor is a simple command interpreter through which all startup options can be invoked.

1. Connect a serial line or Ethernet drop cable to the appropriate port on the Cabinet I/O Panel (see Figure 3-4a and 3-4b).
2. Set the `Boot Environment` configuration switches (switches 5 through 9) to "01000" where "0" means `Closed` and "1" means `Open` (see Appendix A).
3. Set the Monitor `Power` switch located on the Monitor Control Panel to the `On` position.
4. Set the Cabinet `Power` switch located on the front of the Cabinet to the `On` position.

The IRIS Terminal will initially display a PROM Monitor prompt.

```
iris>
```

See Table 6-1 for a list of the commands available through the PROM Monitor. If the PROM Monitor gets stuck, press the `Reset` button.

To reboot the IRIS Terminal, press the `Reset` button on the IRIS Control Panel and repeat the boot procedures outlined above.

## 6.4 Floppy Disk

If the `Boot Environment` configuration switches are set to "00000", then the IRIS Terminal will attempt to read a boot file called *defaultboot* from a floppy disk drive. On the IRIS 1000 Terminal, the floppy disk drive must be connected externally on top of the Cabinet (see Section 4.8). On the IRIS 1200 Terminal, the floppy disk drive is an option that sits inside the Cabinet.

The IRIS Terminal is booted automatically from the floppy disk drive after the IRIS Terminal is turned on. A variation of this boot procedure can be used explicitly through the PROM Monitor (see Section 6.3).

| Command | Description |
|---|---|
| `dboot` | Download a file called *defaultboot* from a floppy disk. Identical to the "Floppy Disk" boot procedure (see Section 6.4). |
| `dboot` *[file]* | Download a file called *file* from a floppy disk. |
| `dboot *` | List files on a floppy disk. Press the `Reset` button to return to the PROM Monitor. |
| `help` | Display a list of PROM Monitor commands. |
| `nboot` | Download a file called *defaultboot* over the Ethernet. Identical to the "Ethernet" boot procedure (see Section 6.1 and 6.7). |
| `nboot` *[host]:[file]* | Download a file called *file* from *host* over the Ethernet. |
| `restart` | Restart the PROM Monitor. |
| `termulate` | Start a simple ASCII interface to the host computer for serial line. Identical to the "Serial Line" boot procedure (see Section 6.2 and 6.7). |

Table 6-1: PROM Monitor Commands

1. Set the `Boot Environment` configuration switches (switches 5 through 9) to "00000" where "0" means `Closed` and "1" means `Open` (see Appendix A).
2. Remove the paper insert from the floppy disk drive before inserting a diskette. Be sure to replace it when the disk drive is not in use.
3. Insert a diskette into the floppy disk drive.
4. (IRIS 1000) Toggle the `Power` switch on the back of the floppy disk drive to the `On` position.
5. Set the `Power` switch on the Monitor Control Panel to the `On` position.
6. Set the `Power` switch on the front of the Cabinet to the `On` position. After the IRIS Terminal is turned on, the floppy disk drive will read a file named *defaultboot* from the floppy diskette and the IRIS Terminal will be booted automatically.
7. After downloading is complete, the IRIS Terminal screen will clear and a rectangle in the bottom of the screen will change colors. This is the window for the VT-52 terminal emulator. The IRIS Terminal will prompt for the name of the host computer.

```
Connect to what host?
```

Reply with *serial*. The window will be redrawn and a prompt will appear in the upper left-hand corner.

To reboot the IRIS Terminal, press the `Reset` button on the IRIS Control Panel and repeat the boot procedures outlined above.

## 6.5 IP/TCP

The IRIS Terminal can be connected to an Ethernet local area network with the IP/TCP protocols.

1. Connect the Ethernet drop cable to the Ethernet port on the Cabinet I/O Panel (see Section 4.7).
2. Set the Monitor `Power` switch located on the Monitor Control Panel to the `On` position.
3. Set the Cabinet `Power` switch located on the front of the Cabinet to the `On` position.
4. Boot the IRIS terminal program (*iris*) with either the serial download utility (*dliris*; see Section 6.2) or a floppy disk (see Section 6.4).
5. The IRIS terminal program will prompt to find the IRIS Terminal's internet address: The syntax is *x.x.x.x*, i.e. four numbers separated by periods. If the representation of a host name in the 4.2BSD */etc/hosts* table is 42.3, then the address given is "42.0.0.3".

```
Enter local host's IP address: 42.0.0.3
```

6. If the network has a gateway address, it should be entered to inform the IRIS terminal program.

```
Enter the local gateway's network hardware address: 0.0.0.0
```

7. Enter the network address of the host that the IRIS Terminal will be connected to.

```
Enter host name: 42.0.0.2
```

8. A login prompt will appear in the upper left-hand corner of this window.

```
login:
```

   To use the IRIS Terminal, login to an account.

To reboot the IRIS Terminal, press the `Reset` button on the IRIS Control Panel and follow the boot procedures outlined above.

## 6.6 Monitor Adjustment

The Monitor Control Panel on the right side of the Monitor has several knobs for adjusting the brightness and contrast of the Monitor.

NOTE: Color rendering and stability may drift for the first 45 minutes after startup.

1. After the Monitor has warmed up, adjust the `Brightness` control knob until the gray raster is barely brighter than the black areas on the edge of the screen. Lighter brightness settings will impair image sharpness and color fidelity.
2. Turn the `Contrast` control knob to the maximum (clockwise) setting for optimum clarity.
3. If the color purity or convergence appear to be out of adjustment, hold down the `Degauss` button on the Monitor Control Panel for about 10 seconds and then release it. The image should improve noticeably.

## 6.7 UNIX Demonstration Programs

After the IRIS Terminal has been booted, it can be tested by running the demonstration programs included on the distribution tape. For C and FORTRAN there are source files for *sqiral* and track in */usr/iris/c* and */usr/iris/f77*.

- *sqiral* draws a square spiral. The spiral is constructed with line drawing commands from the IRIS Graphics Library. After the spiral is finished, the terminal emulator window is redrawn.
- *track* displays a box that can be moved around the screen with the mouse. It is a simple application of the IRIS Graphics Library object creation and editing commands with double buffer mode. To quit *track*, press all three mouse buttons simultaneously.

For FORTRAN there are also source files for *curve*, *floats*, *planets* and *star* in */usr/iris/f77*.

- *curve* initially displays a red arrow on the screen. The mouse controls the arrow's movement. Pushing the left mouse button creates a small square on the screen. Lines are drawn between squares with each additional key click.
- *floats* randomly fills the screen with different colored pixels. *floats* also tests floating point numbers.
- *star* draws a series of stars. First, *star* initializes the color map with 247 shades of red. *star* then draws a star in the upper right corner of the screen with the lightest shade. The star is redrawn with a slight offset with each brighter shade of red. After all the stars are drawn, star exits and the terminal emulator window is redrawn.
- *planets* models a solar system. See the documentation in *planets.doc* in */usr/iris/f77*.

For a more complete explanation of the IRIS Graphics Library commands used in these programs, see the *IRIS User's Guide*. These programs are compiled with the terminal software (see Section 5). To explicitly compile a C program,

```
$ cd /usr/iris/c
$ cc track.c libgl.a -o track
```

To run it,

```
$ track
```

To compile a FORTRAN program,

```
$ cd /usr/iris/f77
$ f77 -w -O -o planets planets.f libgl.a
```

To run it,

```
$ planets
```

Any of these demonstration programs can be terminated by pressing the `ESCAPE` key and then a `BREAK` character (usually `CONTROL`-c).

In addition to these simple demonstration programs, there are a few programs that can be downloaded into the IRIS Terminal. These demonstration programs are located in the directory */usr/iris/boot*. Each program is named with a *.dsk* suffix. The procedures for downloading these programs are outlined below. They differ depending on whether the IRIS Terminal is connected to the host computer with a serial line or an Ethernet connection.

**Serial Line**

If the IRIS Terminal is connected to the host computer via a serial line, then a demonstration program can be downloaded in the same way as the terminal software (see Section 6.2).

1. Set the `Boot Environment` configuration switches (switches 5 through 9) for a normal serial line boot. They should be set to "01100" where "0" means `Closed` and "1" means `Open` (see Appendix A).
2. Login to the host computer.

   ```
   login:
   ```

3. Download the demonstration programs.

   ```
   $ /usr/iris/c/dliris thedemo1.dsk
   ```

The IRIS Terminal initially displays a series of dots to indicate that downloading is underway. The introduction to the demonstration program will appear after a few minutes. To choose a demonstration program, hold down the right mouse button, move the cursor over a menu entry and release the button.

**Ethernet**

If the IRIS Terminal is connected to the host computer via an Ethernet local area network, then the procedure for downloading the demonstration programs in *boot* requires using the PROM Monitor (see Section 6.3).

1. Set the `Boot Environment` configuration switches (switches:; through 9) for the PROM Monitor. Thev should be set to "01000" where "0" means `Closed` and "1" means `Open` (see Appendix A).
2. Download the demonstration program.

```
iris> nboot thedemo1.dsk
```

The IRIS Terminal initially displays a series of dots to indicate that downloading is underway. The introduction to the demonstration program then appears after a few seconds. To choose a demonstration program, hold down the right mouse button, move the cursor over a menu entry and release the button.

**Floppy Disk (optional)**

Some demonstration programs can be downloaded into the IRIS Terminal. These are stored on floppy diskettes.

1. Set the `Boot Environment` configuration switches (switches 5 through 9) to "01000" where "0" means `Closed` and "1" means `Open` (see Appendix A).
2. Remove the paper insert from the floppy disk drive before inserting a diskette. Be sure to replace it when the disk drive is not in use.
3. Insert a diskette into the floppy disk drive.
4. (IRIS 1000) toggle the `Power` switch on the back of the floppy disk drive to the `On` position.
5. Set the `Power` switch on the Monitor Control Panel to the `On` position.
6. Set the `Power` switch on the front of the Cabinet to the `On` position.
7. Determine what files are on the diskette.

```
iris> d *
flight.dsk
Dboot:
```

   To escape this "disk boot mode" press the `Reset` button.

8. Enter a file name and press `RETURN`.

```
iris> d flight.dsk
```

The IRIS Terminal initially displays a series of dots to indicate that downloading is underway. The introduction to the demonstration programs will appear after a few seconds. To choose a demonstration program, hold down the right mouse button, move the cursor over a menu entry and release the button.

## 6.8 IRIS Terminal Shutdown

The IRIS Terminal should not be left on indefinitely. However, since the Monitor has a long warmup period, the IRIS Terminal should be left on continuously during work hours. The IRIS Terminal can be shut down by turning off the `Power` switch on the Cabinet. The Monitor `Power` switch may be left on since power tor the Monitor is controlled by the Cabinet `Power` switch.

1. Set the `Power` switch on the front of the Cabinet to the `Off` position.

## 6.9 Diagnostics

There are two kinds of diagnostics for the IRIS Terminal.

- Automatic diagnostics are run at startup.
- Optional startup diagnostics can be selected with the `Checkout` configuration switch (switch 4; see Appendix A).

### Startup Diagnostics

The IRIS Terminal executes a series of processor-specific diagnostics during the normal startup sequence. If an error is detected, an alternating status code is displayed on the `Status` diagnostic display on the Cabinet I/O Panel. See Table 6-2 for a list of startup diagnostics. If any of these error codes are displayed, contact Silicon Graphics Customer Service (see Section 1).

The hex code shown on the `Status` diagnostic display will change as the startup sequence progresses. This is helpful in netermining the progress of the startup sequence.

### Optional Startup Diagnostic

If the `Checkout` configuration switch (switch 4; see Appendix A) is set to the `Open` position, the IRIS Terminal will write and read each processor memory location with two different values. This diagnostic should be run when the IRIS Terminal is booted for the first time or when a problem is suspected.

| Test | Error Code | Description |
|---|---|---|
| Unexpected exception | 0,B | Be prepared to catch certain exceptions on exit of boot state. |
| Local memory | 0,C | Test the minimum processor memory size needed to operate (first 128K). Random value write/verify test. |
| Refresh timer too short | 0,D | Test timing and exception handling for the memory refresh timer (arm timer, buzz loop and check count). |
| Refresh timer too long | 0,E | This is the other extreme of the refresh timer test. An exception doesn't happen or it doesn't happen soon enough. |
| Context register | 0,6 | Write and read all possible values (16). |
| Segment map | 0,7 | Write and read random values to all segment map registers. |
| Page map | 0,8 | Write and read random values to all page map registers. |

Table 6-2: Startup Diagnostics

# Appendix A: Configuration Switches

| Switch | Name | Position | Meaning |
|---|---|---|---|
| 1 | Reserved | 1[1] | Leave in the $\boxed{\texttt{Open}}$ position. |
| 2, 3 | Serial line | 00 | 300 baud |
| | | 01 | 19,200 baud |
| | | 10 | 1200 baud |
| | | 11 | 9600 baud |
| 4 | Checkout | 0 | No additional testing. |
| | | 1 | Additional testing (time-consuming). |
| 5 - 9 | Boot Environment | 00000 | Floppy disk boot (see Section 6.4). |
| | | 00100 | Network boot (see Section 6.1). |
| | | 01000 | PROM Monitor (see Section 6.3). |
| | | 01100 | Serial line boot (see Section 6.2). |
| | | 00110 | IEEE-488 boot. |
| | | all others | Undefined. |

Table A-1: IRIS Terminal Configuration Switches

1. 1 means $\boxed{\texttt{Open}}$ and 0 means $\boxed{\texttt{Closed}}$.

# Appendix B: VAX-11 Ethernet Software Installation

IRIS Terminals and Workstations can communicate through an Ethernet local area network with the Xerox Network System[1] (XNS) protocols. IRIS systems are preconfigured to support Ethernet communication before they are shipped. IRIS Terminals can be booted from IRIS Workstations, IRIS Workstations can transfer files across the network, .md IRIS Terminals and Workstations can remotely login into IRIS Workstations.

In addition, Digital Equipment Corporation VAX-11[2] minicomputers running 4.2BSD UNIX can communicate on the same network with IRIS Terminals and Workstations. There are two procedures for installing network support on VAX-11 minicomputers: installing the Ethernet software and installing an Ethernet Controller. NOTE: If possible, an IRIS Terminal should be tested over a serial line before connecting it to the Ethernet.

There are five steps to installing the Ethernet software.

- Install network utilities.
- Boot generic kernel.
- Test network utilities.
- Install driver and protocol code.
- Test network utilities.

The Ethernet software must be installed by someone with systems programming experience. This person will install a device driver for the Ethernet Controller. These instructions assume the reader is familiar with the following documents:

[1] *Installing and Operating 4.2BSD on the VAX*
S. J. Leffler and W. N. Joy,
University of California, Berkeley;
July 1983.

1. Xerox Network System is a trademark ot Xerox Corporation.
2. VAX-11 is a trademark of Digital Equipment Corporation.

[2] *Building 4.2BSD UNIX Systems with Config*
S. J. Leffler,
University of California, Berkeley;
July 1983.

[3] *4.2BSD System Manual*
W. N. Joy, et al.,
University of California, Berkeley;
July 1983.

## B.1 Ethernet Software

The distribution tape contains all the software necessary for a VAX-11 minicomputer to communicate through an Ethernet local area network. This software includes a device driver for the Ethernet Controller, the XNS protocols, network server daemons and application programs. In addition, a version of the 4.2BSD UNIX kernel is supplied with the Ethernet driver already installed. This kernel can be booted and run on any system where a generic 4.2BSD UNIX kernel will run. It is then possible to test host Ethernet capability without first going through the kernel installation procedure.

The files and directories mentioned below are included on a seperate distribution tape. They can be copied anywhere on the host file system. For purposes of explanation, they are assumed to be installed in */usr/iris/net*.

There are three distribution levels: binary, driver source and protocol source. In the software specification that follows, some files are followed with a *.[co]* suffix to indicate that they are dependent on the distribution level.

*boot*

- *MAKENETDEV* is a Shell script that creates the network special files and adds appropriate lines to */etc/ttys* and */etc/ttytype*. *MAKENETDEV* takes two arguments: the major device number of the network driver, and the number of minor devices.
- *Makefile* is a make description file for compiling *sgbounce* and *sgboot*.
- */etc/rc.local* contains commands for starting the network daemons.
- */etc/ttys* is an example of the */etc/ttys* file that *MAKENETDEV* generates.
- *sgboot.c* contains source code for *sgboot*, the network bootserver.
- *sgbounce.c* contains source code for *sgbounce*, the network nameserver.

*cmd*

- *Makefile* is a make description file for compiling the XNS network utilities.
- *xcp.c* contains source code for *xcp*, the remote file copy utility.

- *xlogin.c* contains source code for *xlogin*, the remote login utility.
- *xnsd.c* contains source code for *xnsd*, the network utility daemon.
- *xx.c* contains source code for *xx*, the remote shell utility.

*kernel*

- *SGENERIC* is a bootable 4.2BSD UNIX kernel. It differs trom the standard distribution kernel in that an Ethernet driver and protocol code have alreadv been installed.
- *SYSCONF* is an example of how the Ethernet device and XNS protocol code is added to the system configuration file in */sys/conf*.
- *Xns.h* contains network structure definitions tor XNS protocols.
- *Xnsconn.h* contains the system-specific representation for a network connection.
- *Xnsioctl.h* defines the *ioctl* codes used by the Ethernet Controller device driver.
- *Xnstrace.h* is an include file for the protocol debug / trace module.
- *conf.c* contains device definitions. It is a copy of the 4.2BSD file */sys/vax/conf.c* with an Ethernet character device added as major device 40.
- *files* is a list of files read by */etc/config* when a new kernel *Makefile* is constructed. It is a copy of the 4.2BSD file */sys/conf/files* with three type codes added.
- *if_il.[co]* is the device driver for the Ethernet Controller. It contains the 4.2BSD UNIX driver with a character device interface added and some changes to call the XNS protocol routines.
- *if_xns.h* is an include file shared by the device driver and protocol modules.
- *mbuf.h* is a copy of the 4.2BSD file */sys/h/mbuf.h* with three type codes added.
- *xns.[co]* contains the XNS protocols.
- *xns_subs.[co]* contains butter management routines.
- *xns_tty.[co]* provides the interface between XNS and the operating system.
- *xns_uba.[co]* provides the interface between the VAX I/O hardware and the operating system.

*lib*

- *Makefile* is a *make* description file for compiling the XNS network library (*libxns.a*).

- *conserr.c* contains routines used by *xnsd* to print error routines on the system console.
- *mpr.c* contains debug print routines.
- *trdwri.c* contains routines that provide timed system calls.
- *ttymodes.c* contains functions for setting terminal modes.
- *utmp.c* contains utility routines for manipulating the */etc/utmp* file.
- *xcmd.c* contains functions for running a command on another host.
- *xconnect.c* contains functions for making connections with another host.
- *xnsfile.c* contains routines that find an available network special file.
- *xnswrite.c* contains functions for sending typed packets.

*man*

- *net.doc* contains *troff* source for Appendix B (this document) of the *IRIS Terminal Guide*.
- *sgboot.1m* is a manual entry for *sgboot*, the network bootserver.
- *sgbounce.1m* is a manual entry for *sgbounce*, the network nameserver.
- *xcp.1* is a manual entry for *xcp*, the remote file copy utility.
- *xlogin.1* is a manual entry for *xlogin*, the remote login utility.
- *xx.1* is a manual entry for *xx*, the remote shell utility.

## B.2 Application Software Installation

The distribution software for the VAX-11 Ethernet Software is delivered on a 1600 bpi magnetic tape in either *tar* or *cpio* format.[3] The distribution software can be stored anywhere on the host file system. For purposes of explanation, the examples below assume that the distribution software will be installed in a directory called */usr/iris/net*.

1. Login to UNIX.
2. Create a directory to hold the distribution software.

```
$ mkdir /usr/iris/net
```

3. Change the current directory to the new directory */usr/iris/net*.

```
$ cd /usr/iris/net
```

3. See Volume 1 of the *UNIX Programmer's Manual*.

4. Load the distribution tape onto a tape drive and read the distribution software into the new directory.

   ```
   $ tar x        or      $ cpio -i < /dev/rmt1
   ```

   A label indicates the format of the software on the distribution tape.

The sections that follow have instructions for installing the network include files and installing the network library, daemons and utilities.

## Network Include Files

The network include files are in *net/kernel*. These files must be installed before the network library or any of the network utilities or daemons may be compiled.

1. Make a directory for the network include files.

   ```
   $ mkdir /usr/include/xns
   ```

2. Copy the network include files into the new directory.

   ```
   $ cd /usr/iris/net/kernel
   $ cp Xns.h /usr/include/xns
   $ cp Xnsconn.h /usr/include/xns
   $ cp Xnsioctl.h /usr/include/xns
   $ cp Xnstrace.h /usr/include/xns
   $ cp if_xns.h /usr/include/xns
   ```

## Network Library

The source files for the network library (*libxns.a*) are in *net/lib*. This library must be compiled before any of the other application software can be compiled.

1. Compile the network library.

   ```
   $ cd /usr/iris/net/lib
   $ make all
   ```

## Network Daemons

The source files for the network daemons (*sgboot* and *sgbounce*) are in *net/boot*.

1. Compile the network daemons.

   ```
   $ cd /usr/iris/net/boot
   $ make all
   ```

2. Install the network daemons.

   ```
   $ cp sgboot /etc
   $ cp sgbounce /etc
   ```

**Network Utilities**

The source files for the network utilities (*xcp*, *xlogin*, *xx* and *xnsd*) are in *net/cmd*.

1. Compile the network utilities.

```
$ cd /usr/iris/net/cmd
$ make all
```

2. Install the network utilities.

```
$ cp xnsd /etc
$ cp xcp /usr/local/bin
$ cp xlogin /usr/local/bin
$ cp xx /usr/local/bin
```

3. *xnsd* should be owned by "root".

```
$ chown root /etc/xnsd
```

## B.3 *SGENERIC* Kernel Test

*SGENERIC* is a bootable 4.2BSD UNIX kernel. The procedures below require the host operating system to be in single-user mode. They will verify that the hardware on the host and the IRIS Terminal are working correctly.

1. Backup the "root" file system onto tape.
2. Install the file *SGENERIC* in the root directory of the host computer's file system (or the directory where *vmunix* is loaded).
3. Boot *SGENERIC* before altering the VAX hardware.
4. After *SGENERIC* has been booted and UNIX is in single-user mode, bring the system back down.
5. Install the Ethernet Controller Board (see Appendix C).
6. Boot *SGENERIC* again. The kernel should find the Ethernet Controller and print its Ethernet address, UNIBUS address, and interrupt vector during auto configuration.

## B.4 Configuration and Special Files

There are several system configuration files that must be edited to allow IRIS Terminals and Workstations to login over the network.

1. Run the *MAKENETDEV* Shell script. This will create network special files and add appropriate lines to */etc/ttys*. *etc.ttys* is an example of the */etc/ttys* file that *MAKENETDEV* generates. *MAKENETDEV* takes two arguments: the major device number of the network driver, and the number of minor devices. The correct use of *MAKENETDEV* with *SGENERIC* is:

```
# MAKENETDEV 40 32
```

This specifies that there are 32 special files for major device 40. The command will create minor devices 0 through 31 and enable logins on minor devices 1 through 15. Note that device 0 must not be used for logins or other I/O.

2. Edit */etc/ttytype* to include special device types for the IRIS Terminal.
3. Make */etc/init* read */etc/ttys* and start *getty*'s for the new devices.

```
# kill -1 1
```

4. Begin multi-user mode.

```
# CONTROL-d
```

5. Start the daemons for the IRIS Terminal.

```
# /etc/sgbounce `/bin/hostname` /usr/iris/boot &
# /etc/sgboot `/bin/hostname` /usr/iris/boot &
```

These daemons will allow an IRIS Terminal to download a boot file from a VAX host and then login to that host.

6. Start the daemon for the network utilities.

```
# /etc/xnsd &
```

This daemon starts remote processes for the network utilities.

7. Test the network by copying files across the network.

```
# xcp foo `/bin/hostname`:/tmp/foo
# xx `/bin/hostname`:date
```

8. Test the IRIS Terminal by following the boot procedures in Section 6.1 and running some of the demonstration programs in Section 6.7. This will verify that the Ethernet hardware is working correctly. If *SGENERIC* works correctly, then the rest of the Ethernet software can be installed as outlined below.
9. If the network server daemons work correctly, then append the file *net/boot/etc.rc.local* to */etc/rc.local*.

```
# cat etc.rc.local >> /etc/rc.local
```

The commands in this file will start the network daemons when the system begins multi-user mode.

## B.5 Kernel Installation

The distribution files in *net/kernel* are 4.2BSD UNIX files with only Ethernet-specific code added. Before installing these files, check for differences (i.e. use */bin/diff* between the file on the host system and the distribution file.

1. The Ethernet device and XNS protocol code must be added to the system configuration file in */sys/conf. net/kernel/SYSCONF* is an example of how this is done. The device entry for *il0* is for the hardware interface. The XNS pseudo-device selects the protocol code.
2. Check for differences between */sys/conf/files* and */net/kernel/files*.
3. Add the XNS-specific lines at the end of */net/kernel/files* to */sys/conf/files*.
4. Run */etc/config* to generate a system configuration.

   ```
   # /etc/config /sys/conf [HOST NAME]
   ```

5. Check for differences between */net/kernel/mbuf.h* and */sys/h/mbuf.h*.
6. Add the XNS-specific lines in */net/kernel/mbuf.h* to */sys/h/mbuf.h*.
7. A source file for the Ethernet device driver must be put in *sys/vaxif*. For a binary distribution tape, an empty source file can be created.

   ```
   # touch /sys/vaxif/if_il.c
   ```

   For a source distribution tape, the source file should be copied into the appropriate directory.

   ```
   # cp if_il.c /sys/vaxif
   ```

8. Install the object code for the Ethernet device driver in the kernel configuration directory in */sys*. For a source distribution tape, this step can be skipped since the driver will be installed by the make procedure. For a binary distribution tape, it is critical that the date last edited for this file be more recent than the corresponding empty source file.

   ```
   # cd /usr/iris/net/kernel
   # cp if_il.o /sys/[HOST NAME]
   ```

9. Create a symbolic link between the directories */usr/include/xns* and */sys/xns*.

   ```
   # ln -s /usr/include/xns /sys
   ```

10. Source files for the XNS protocol code *xns.c*, *xns_subs.c*, *xns_tty.c* and *xns_uba.c* must be put in */sys/xns*. For a binary distribution tape, an empty source file can be created.

    ```
    # touch /sys/xns/xns.c
    # touch /sys/xns/xns_subs.c
    # touch /sys/xns/xns_tty.c
    # touch /sys/xns/xns_uba.c
    ```

For a source distribution tape, the source file should be copied into the appropriate directory.

```
# cp xns.c /sys/xns
# cp xns_subs.c /sys/xns
# cp xns_tty.c /sys/xns
# cp xns_uba.c /sys/xns
```

11. Install the object code for the XNS protocols in the kernel configuration directory in */sys*. For a source distribution tape, this step can be skipped since the driver will be installed by the *make* procedure. For a binary distribution tape, it is critical that the date last edited for these files be more recent than the corresponding empty source file.

```
# cp xns.o /sys/[HOST NAME]
# cp xns_subs.o /sys/[HOST NAME]
# cp xns_tty.o /sys/[HOST NAME]
# cp xns_uba.o /sys/[HOST NAME]
```

12. Check for differences between *net/kernel/conf.c* and */ sys/vax/conf.c*. The prototype device switch is in */net/kernel/conf.c*. The new character device must appear in the data structure *cdevsw* in the file *conf.c*. Major devices are defined in the 4.2BSD UNIX distribution as items 0 through 32. The device number for the network is 40 so that *SGENERIC* can work on a system with local device drivers.
13. Add the Ethernet driver-specific lines in */net/kernel/conf.c* to */sys/vax/conf.c*.
14. Change the current directory to the kernel configuration directory in */sys*.

```
# cd /sys/[HOST NAME]
```

15. Check the dependencies for the *Makefile* generated by */etc/config*.

```
# make depend
```

16. Make a new kernel.

```
# make
```

17. Boot the newly created kernel and start multi-user mode.
18. Test the network by copying files across the network.

```
$ xcp foo `/bin/hostname`:/tmp/foo
$ xx `/bin/hostname`:date
```

19. Test the IRIS Terminal by following the procedures in Section 6.1.

## NAME

sgboot – provide network boot service

## SYNOPSIS

**sgboot** [*system name*] [*boot directory*]

## DESCRIPTION

*sgboot* is a daemon that, provide XNS boot service to IRIS Terminals on an Ethernet local area network. This service is provided on the socket *BOOT-SOCKET* defined in */usr/include/xns/Xns.h*. After the client obtains a connection, *sgboot* expects the client to supply the name of the boot file to send. *sgboot* then transmits the file and closes the connection. The pathname of the boot file may be absolute or relative to the boot directory specified on the command line.

The system name on the command line should be identical to the kernel's idea of the host name (i.e. */bin/hostname*). This daemon is normally run in the background. For example,

```
# sgboot `/bin/hostname` /usr/iris/boot &
```

*sgboot* can be started by hand, as shown above, but should normally be started in the file */etc/rc.local*.

## NOTE

Multiple copies of *sgboot* may be running at a given time. The total number of instances of *sgboot* equals the number of IRIS Terminals that may be booted in parallel.

## SEE ALSO

Silicon Graphics, Inc., *IRIS Terminal Guide*, Appendix B

## NAME

sgbounce – provide network name service

## SYNOPSIS

**sgbounce** [*system name*] [*boot directory*]

## DESCRIPTION

*sgbounce* is a daemon that provides name service to IRIS Terminals on an Ethernet local area network. The system name on the command line should be identical to the kernel's idea of the host name (i.e. */bin/hostname*). *sgbounce* is normally run in the background. For example,

```
# sgbounce `/bin/hostname` /usr/iris/boot &
```

*sgbounce* can be started by hand, as shown above, but should normally be started in the file */etc/rc.local*.

## NOTE

Only one copy of *sgbounce* may be running at one time.

## SEE ALSO

Silicon Graphics, Inc., *IRIS Terminal Guide*, Appendix B
sgboot(1M)

## NAME

xcp – remote file copy

## SYNOPSIS

**xcp** file1 file2
**xcp** [ **–r** ] file... directory

## DESCRIPTION

*xcp* copies files between machines. *file1* is copied to *file2* or *file* is copied to *directory/file*.

Each *file* or *directory* argument is either a remote file name of the form *rhost:path*, or a local file name (with a '/' inserted before any ':'s). The login name of the user sending a file over the network must be recognized by the remote machine or *rhost* may take the form *rhost.rname* to use *rname* rather than the current login name on the remote host. *xcp* does not prompt for passwords; your current local login name must exist on *rhost* and allow remote command execution via *xlogin*(l). If path is not a full path name, it is interpreted relative to your login directory on rhost. A path on a remote host may be quoted (using \, ", or ') so that the metacharacters are interpreted remotely. *xcp* handles third party copies, where neither source nor target files are on the current machine.

If the argument **–r** is specified and any of the source files are directories, *xcp* copies each subtree rooted at that name; in this case the destination must be a directory.

## SEE ALSO

*xlogin(1)*, *xx(1)*

## BUGS

Doesn't detect all cases where the target of a copy might be a file in cases where only a directory should be legal.

Is confused by any output generated by commands in a *.login*, *.profile*, or *.cshrc* file on the remote host.

## NAME

xlogin – remote login

## SYNOPSIS

**xlogin** rhost

## DESCRIPTION

*xlogin* connects your terminal on the current local host system to the remote host system *rhost*.

All echoing takes place at the remote site, so that (except for delays) *xlogin* is transparent. Flow control via ^*S* and ^*Q* and flushing of input and output on interrupts are handled properly. A line of the form "~." disconnects from the remote host.

*xlogin* times out after 60 seconds if no login is attempted.

## SEE ALSO

*xcp(1)*, *xx(1)*

## BUGS

More terminal characteristics should be propagated.

**NAME**

xx - remote shell

**SYNOPSIS**

**xx** host command

**DESCRIPTION**

*xx* connects to the specified *host* and executes the specified *command*. *xx* copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are propagated to the remote command; *xx* normally terminates when the remote command does.

The remote login name must be equivalent (in the sense of *sh(1)*) to the originating account; no provision is made for specifying a password with a command.

If you omit *comand*, then instead of executing a single command, you will be logged in on the remote host using *xx(1)*.

Shell metacharacters which are not quoted are interpreted on the local machine, while quoted metacharacters are interpreted on the remote machine. Thus the command

```
$ xx otherhost cat remotefile > > localfile
```

appends the remote file *remotefile* to the local file *localfile*, while

```
$ xx otherhost cat remotefile "> >" otherremotefile
```

appends *remotefile* to *otherremotefile*.

**SEE ALSO**

*xlogin(1)*, *xcp(1)*

**BUGS**

If you are using *csh(1)* and put an *xx(1)* in the background without redirecting its input away from the terminal, it will block even if no reads are posted by the remote command.

You cannot run an interactive command (like *vi(1)*); use *xlogin(1)*.

Stop signals stop the local *xx* process only; this is arguably wrong, but currently hard to fix tor reasons too complicated to explain here.

# Appendix C: Ethernet Hardware Installation

## C.1 Ethernet Hardware

There are six components necessary to connect a VAX 11/780 or 11/750 to an Ethernet local area network.

- The *Ethernet Controller* is a UNIBUS[1]-hex board for interfacing a VAX-11/780 or VAX-l1/750 to an Ethernet local area network.
- 1 *Ethernet Transciever* taps the Ethernet coaxial cable for a connection to a network node.
- 1 75-foot, 15-conductor drop cable connects the Cabinet to an Ethernet transceiver.
- 1 Flat cable connects the Ethernet Controller to the drop cable.
- 1 slide lock connects the flat cable to the drop cable.
- 1 panel connector (VAX-11/750 only) holds the Ethernet Controller in place.

## C.2 Hardware Installation

The Ethernet Controller is a UNIBUS-hex board that contains all of the data communications controller logic necessary to interface a VAX-11/780 or VAX-11/750 minicomputer to an Ethernet local area network. It is installed in the UNIBUS subsystem on the host computer. The UNIBUS subsystem connects peripheral devices to a VAX-11 family of minicomputers. The procedures for installing the Ethernet Controller in the UNIBUS subsystem differ slightly between the VAX-11/780 and the VAX-11/750.

1. Inspect the Ethernet Controller for damage.
2. Check that the switches on the Ethernet Controller are set to:

1. UNIBUS is a trademark of Digital Equipment Corporation.

| Switch | Setting |
|---|---|
| v23 | 01011110 |
| v22 | 01011111 |

Table C-1: Ethernet Controller Switch Settings

3. Shut down UNIX (i.e. run */etc/shutdown*).
4. Prepare the UNIBUS subsystem.

   *VAX-11/780*
   a. Turn off the power on the UNIBUS subsystem using the switch on the rear of the box.
   b. Carefully slide the UNIBUS subsystem out of the cabinet.
   c. Remove the top and bottom covers of the UNIBUS subsystem.

   *VAX-11/750*
   a. Turn off the power for the CPU.
   b. Set the front panel switch to `Halt`.
   c. Open the front -and back ends of the processor cabinet, exposing the UNIBUS subsystem.
5. Determine an appropriate slot in the UNIBUS for the Ethernet Controller. The Ethernet Controller will fit into one SPC slot.
6. Remove the DMA jumper (*ca1* to *cb1*) from the selected slot.
7. Install the Ethernet Controller in the selected slot.
8. Lock down the Ethernet Controller.
9. Connect the 16-pin Ethernet cable to the Ethernet Controller.

   *VAX-11/780*
   a. Route the flat cable from the Ethernet Controller to the back of the UNIBUS subsystem.
   b. Install the flat cable from the Ethernet Controller to the back of the UNIBUS subsystem.
   c. Connect the 15-pin connector on the flat cable to the 15-pin connector on the transceiver.

   *VAX-11/750*
   a. Install the mounting plate with the connector to an available port slot on the rear of the cabinet.
   b. Route the flat cable from the back of the UNIBUS subsystem to the Ethernet Controller.

c. Connect the 15-pin connector on the transceiver to the new port.

10. Attach the transceiver to the Ethernet coaxial cable.
11. Connect the drop cable to the transceiver.
12. Restore the UNIBUS subsystem.

*VAX-11/780*

a. Replace the top and bottom covcrs on the UNIBUS subsystem.

b. Slide the UNIBUS subsystem back in place.

c. Turn on the power on the UNIBUS subsystem using the switch on the rear of the box.

*VAX-11/750*

a. Close the front and back ends of the processor cabinet.

b. Set the front panel switch to `Boot`.

c. Restart the host computer.

13. Boot UNIX.
14. If UNIX operates correctly with the new interface installed, shut down the system again and boot *SGENERIC* (see Section B.2).
15. If there are no problems after restarting the host computer, the Ethernet software can be installed (see Appendix B).

# Appendix D: VAX/VMS Software Installation

## D.1 VMS Software Specification

The VAX/VMS distribution tape contains software that resides on the host computer. The procedures that follow assume that the IRIS Terminal has already been unpacked and attached to a host computer with a serial line. See Section 4 for instructions on how to install the IRIS Terminal hardware.

- *CURVE.FOR* is a demonstration program that initially displays a red arrow on the screen. The mouse controls the arrow's movement. Pushing the left mouse button creates a small square on the screen. Lines are drawn between squares with each additional key click.
- *DEVICE.H* is an include file with device definitions, e.g. mouse buttons. IRIS Terminal users will probably want to include this file in an application program.
- *DLIRIS.MAR* serially downloads the IRIS terminal program (*IRIS.*) into the IRIS Terminal.
- *FLOATS.FOR* is a demonstration program that randomly fills the screen with different colored pixels. *FLOATS* also tests floating point numbers.
- *FRTINSTAL.COM* is a command file that assembles and links the serial download program (*DLIRIS*), creates the IRIS Graphics Library (*SGILIB.OLB*) and compiles and links the demonstration programs with *SGILIB.OLB*.
- *FRTINSTAL.TXT* is a text file with update notes to this appendix.
- *GET.H* is an include file with definitions for values returned by the IRIS Graphics Library *get* commands. IRIS Terminal users will probably want to include this file with their application programs.
- *GL.H* is an include file with definitions for default color map values, screen boundaries, and function declarations. IRIS Terminal users will probably want to include this file with their application program.

- *IO.MAR* contains a set of routines that send and receive information to and from the IRIS Terminal.
- *IRIS.* is the binary terminal program used with *DLIRIS*.
- *PLANETS.DOC* describes the *PLANETS* demonstration program.
- *PLANETS.FOR* models a solar system. See the file *PLANETS.DOC*.
- *SQIRAL.FOR* is a demonstration program that draws a square spiral. The spiral is constructed with line drawing commands from the IRIS Graphics Library. After the spiral is finished, the terminal emulator window is redrawn.
- *STAR.FOR* is a demonstration program that draws a series of stars. *STAR* initializes the color map with 247 shades of red. It then draws a star in the upper right corner of the screen with the lightest shade. The star is redrawn with a slight offset with each brighter shade of red. After all the stars are drawn, *STAR* exits and the terminal emulator window is redrawn.
- *TRACK.FOR* is a demonstration program that displays a blue box that can be moved around the screen with the mouse. It is a simple application of the IRIS Graphics Library object creation and editing commands with double buffer mode. To quit *TRACK* press all three mouse buttons simultaneously.

## D.2 VMS Software Installation

The VMS distribution tape is a standard 1600 bpi BACKUP tape. The tape may contain two BACKUP files. Only the file *FORTR1C.BCK* is of interest for this distribution. This tape includes host software that resides on the host computer and terminal software that must be downloaded from the host computer into the IRIS Terminal each time it is booted.

The VMS distribution software should be installed on a device that is accessible to the people who will be using the IRIS Terminal.

1. Login as the system manager.
2. Create a directory on the user disk.

```
$ ASSIGN (user disk name): IRIS$DISK
$ CREATE/DIRECTORY IRIS$DISK:[IRIS]
$ CREATE/DIRECTORY IRIS$DISK:[IRIS.FORTR1C]
$ SET DEFAULT IRIS$DISK:[IRIS.FORTR1C]
$ SET PROTECTION=(G:R,W:R)/DEFAULT
```

3. Mount the installation tape on the tape drive, with the write ring removed and the density set at 1600 BPI.
4. Read the tape:

```
$ ASSIGN (tape drive name): TAPE
$ MOUNT/FOREIGN/NOWRITE/DENSITY=1600 TAPE
$ BACKUP/VERIFY/LOG TAPE:FORTR1C.BCK []*.*
```

5. Install the IRIS Terminal host software.

   ```
   $ @FRTINSTAL.COM
   ```

   This script does the following:

   - Creates the FORTRAN version of the IRIS Graphics Library (*SGILIB.OLB*).
   - Compiles and links the demonstration programs with *SGILIB.OLB*.
   - Assembles and links the serial download program (*DLIRIS*).

See Section D.4 for instructions on running these demonstration programs.

## D.3 VMS Boot Procedures

The procedures for booting an IRIS Terminal are host-independent for both the Ethernet boot (see Section 6.1) and the floppy disk boot (see Section 6.4). The serial line boot procedure is different since the download command must be explicitly run on the VMS host. The steps that follow describe the serial line boot procedure for VMS.

If the `Boot Environment` configuration switches are set to "01100", then the IRIS Terminal will emulate a simple ASCII terminal interfaced to the host computer. The user must then login and download the terminal software.

1. Connect a serial line from the host computer to `Port 2` on the Cabinet I/O Panel (see Section 4.6).
2. Set the `Boot Environment` configuration switches (switches 5 through 9) to "01100" where "0" means `Closed` and "1" means `Open` (see Appendix A).
3. Set the `Serial Line Baud Rate` configuration switches (switches 2 and 3) to a suitable baud rate (see Section 4.6 and Appendix A). For example, to set the baud rate to 9600 baud, set these switches to "11" ("0" means `Closed` and "1" means `Open`.
4. Set the Monitor `Power` switch located on the Monitor Control Panel to the `On` position.
5. Set the Cabinet `Power` switch located on the front of the Cabinet to the `On` position.
6. The IRIS Terminal will provide a simple ASCII terminal interface to the host computer and display a login prompt. Login to an account with permission to use the IRIS Terminal.

```
USERNAME: IRIS
PASSWORD:
```

7. Set the default directory to the directory containing the IRIS Terminal software.

   ```
   $ SET DEFAULT IRIS$DISK:[IRIS.FORTR1C]
   ```

8. Define the symbol *IRIS$INPUT* to be *IRIS*, the binary terminal program.

   ```
   $ DEFINE IRIS$INPUT IRIS.
   ```

9. Set the terminal characteristics to emulate a VT-52 terminal.

   ```
   $ SET TERM/VT52
   ```

10. Run the *DLIRIS* program to serially download the binary terminal program into the IRIS Terminal.

    ```
    $ RUN DLIRIS
    ```

    The IRIS Terminal initially displays a series of dots to show that downloading is underway. The time necessary for downloading the terminal software is dependent on the baud rate of the serial line to the host computer and the load on the host computer. For a 9600 baud connection (recommended), this time is about 5 minutes.

11. After downloading is complete, the IRIS Terminal screen will clear and a rectangle in the bottom of the screen will change colors. This is the window for the VT-52 terminal emulator. The IRIS Terminal will prompt for the name of the host computer.

    ```
    Connect to what host?
    ```

    Reply with *serial*. The window will be redrawn and a prompt will appear in the upper left-hand corner.

To reboot the IRIS Terminal, press the $\boxed{\texttt{Reset}}$ button on the IRIS Control Panel and repeat the boot procedures outlined above.

## D.4 VMS Demonstration Programs

After the IRIS Terminal has been booted, it can be tested by compiling and running the demonstration programs included on the distribution tape. For a more complete explanation of the IRIS Graphics Library commands used in these programs, see the *IRIS User's Guide*.

- *CURVE* initially displays a red arrow on the screen. The mouse controls the arrow's movement. Pushing the left mouse button creates a small square on the screen. Lines are drawn between squares with each additional key click. To exit *CURVE*, press all three mouse buttons.

- *FLOATS* randomly fills the screen with different colored pixels. *FLOATS* also tests floating point numbers.
- *PLANETS* models a solar svstem.
- *SQIRAL* draws a square spiral. The spiral is constructed with line drawing commands from the IRIS Graphics Library. After the spiral is finished, the terminal emulator window is redrawn.
- *STAR* draws a series of stars. First, *STAR* initializes the color map with 247 shades of red. *STAR* then draws a star in the upper right comer of the screen with the lightest shade. The star is redrawn with a slight offset with each brighter shade of red. After all the stars are drawn, *STAR* exits and the terminal emulator window is redrawn.
- *TRACK* displays a box that can be moved around the screen with the mouse. It is a simple application of the IRIS Graphics Library object creation and editing commands with double buffer mode. To quit *TRACK*, press all three mouse buttons simultaneously.

These demonstration programs serve as tests of the IRIS Terminal and the IRIS Graphics Library. They are compiled and linked with the IRIS Graphics Library (*SGILIB.OLB*) by the command script *FRTINSTAL.COM*. The demonstration programs can be run with a single command. For example,

```
$ RUN SQIRAL
```

The source files for each demonstration program are included on the distribution tape. Here is an example of how to compile and link a demonstration program with the IRIS Graphics Library.

1. Compile a demonstration program. For example,

   ```
   $ FORTRAN/F77/NOSTANDARD/I4/OPTIMIZE SQIRAL
   ```

2. Link the object module with the IRIS Graphics Library. For example,

   ```
   $ LINK SQIRAL,SGILIB/LIB
   ```

3. Run the executable module. For example,

   ```
   $ RUN SQIRAL
   ```

   Any of these demonstration program can be terminated by pressing the $\boxed{\texttt{ESCAPE}}$ key and then a $\boxed{\texttt{BREAK}}$ character (usually $\boxed{\texttt{CONTROL}}$-c).

In addition to these simple demonstration programs, there are demonstration programs that can be downloaded into the IRIS Terminal over a serial line with *DLIRIS* or from a floppy disk.

**Serial Line**

If the IRIS Terminal is connected to the host computer via a serial line, then a demonstration program can, be downloaded in the same way as the terminal, software (see Section D.3).

1. Set the $\boxed{\texttt{Boot Environment}}$ configuration switches (switches 5 through 9) for a normal serial line boot. They should be set to "01100" where "0" means $\boxed{\texttt{Closed}}$ and "1" means $\boxed{\texttt{Open}}$ (see Appendix A).
2. Login to the host computer.

```
USERNAME: IRIS
PASSWORD:
```

3. Set the default directory to the directory containing the IRIS Terminal software.

```
$ SET DEFAULT IRIS$DISK:[IRIS.FORTR1C]
```

4. Define the symbol *IRIS$INPUT* to be the name of the demonstration program.

```
$ DEFINE IRIS$INPUT FLIGHT.
```

5. Download the demonstration programs.

```
$ RUN DLIRIS
```

The IRIS Terminal initially displays a series of dots to indicate that downloading is underway. The introduction to the demonstration program will appear after a few minutes. To choose a demonstration program, hold down the right mouse button, move the cursor over a menu entry and release the button.

**Floppy Disk**

Some demonstration programs can be downloaded into the IRIS Terminal. These are stored on floppy diskettes.

1. Set the $\boxed{\texttt{Boot Environment}}$ configuration switches (switches 5 through 9) to "01000" where "0" means $\boxed{\texttt{Closed}}$ and "1" means $\boxed{\texttt{Open}}$ (see Appendix A).
2. Remove the paper insert from the floppy disk drive before inserting a diskette. Be sure to replace it when the disk drive is not in use.
3. Insert a diskette into the floppy disk drive.
4. (IRIS 1000) toggle the $\boxed{\texttt{Power}}$ switch on the back of the floppy disk drive to the $\boxed{\texttt{On}}$ position.
5. Set the $\boxed{\texttt{Power}}$ switch on the Monitor Control Panel to the $\boxed{\texttt{On}}$ position.

6. Set the $\boxed{\texttt{Power}}$ switch on the front of the Cabinet to the $\boxed{\texttt{On}}$ position.
7. Determine what files are on the diskette.

```
iris> d *
flight.dsk
Dboot:
```

   To escape this "disk boot mode" press the $\boxed{\texttt{Reset}}$ button.
8. Enter a file name and press $\boxed{\texttt{RETURN}}$.

```
iris> d flight.dsk
```

The IRIS Terminal initially displays a series of dots to indicate that downloading is underway. The introduction to the demonstration progams will appear after a few seconds. To choose a demonstration program, hold down the right mouse button, move the cursor over a menu entry and release the button.

# Appendix E: IRIS Terminal-Host RS-232 Interface

## E.1 Hardware Interface Assumptions

An IRIS Terminal can be serially connected to a host computer through `Port 2` on the Cabinet I/O Panel. This port is a *Data Terminal Equipment* (DTE) type RS-232 port. A "null modem" cable is required to connect an IRIS Terminal to most hosts. A "straight over" cable may be used to connect the IRIS Terminal to a modem, with the exception that the IRIS Terminal does not provide *Request to Send* (RTS) or *Clear to Send* (CTS) support.

The IRIS Terminal asserts the *Data Terminal Ready* (DTR) signal on pin 20. This is provided for hosts that expect to see *Data Carrier Detect* (DCD) on pin 8 and/or *Data Set Ready* (DSR) on pin 6. The IRIS Terminal does not require the DCD or DSR signals for a serial line connection. It will operate with a three-wire connection (*Transmit Data* on pin 2, *Receive Data* on pin 3 and *Signal Ground* on pin 7). Typically, to connect a modem to the IRIS Terminal, a cable is required with pins 4 and 5 connected on the modem end.

## E.2 Software Interface Assumptions

If the host has sustained output, the IRIS Terminal will attempt to perform how control on data sent to it from the host. The IRIS Terminal sends the ASCII characters *XOFF* (Control-S) and *XON* (Control-Q) to the host to signal *Stop Data Transmission* and *Resume Data Transmission* respectively.

Data sent from the IRIS Terminal to the host is regulated in a software handshake between the Host Graphics Library routine and the IRIS Terminal. This handshake is based on a series of block transfers of data, each preceded by an acknowledge from the Host Graphics Library routine receiving the data. That is, if a Graphics Library routine causes the IRIS Terminal to return a large block of data, the Host Graphics Library routine allows a small block of data at a time to be sent to it from the IRIS Terminal, and allows the next block to be sent by sending an ACK character to the IRIS Terminal. Graphics Library routines like *readpixels* perform this kind of flow control. The IRIS Terminal assumes that 50 characters may be sent to the host at a time. This limitation is transparent to the user in an application program, since the Graphics Library routine does the work.

Given the above system level assumptions, the host must have an ASCII terminal driver which suspends and resumes data transmission when XON/XOFF are sent, and which is capable of buffering at least 50 characters from the terminal when the user's application program is swapped out.

# Appendix F: IRIS Serial Protocol Description

## F.1 Overview

This document describes the set of routines that must be implemented on each host/operating system combination to communicate with the IRIS. The IRIS distribution tape contains a file (*io.c*) with the source code for such an implementation of these communication routines in C for UNIX. The protocol documented here is subject to change without notice.

The IRIS serial protocol is basically the same whether it is run on a serial line or over an Ethernet. We assume that the communications medium is error-free. This is achieved on the Ethernet using the XNS or IP/TCP protocols to support an error-free byte stream, and is usually true of RS-232 direct connections to a host.

The protocol is designed so that if no special escape characters are sent, the IRIS will behave like a standard terminal connected to the host running full-duplex to a serial or network port. Characters typed on the IRIS keyboard are transmitted directly to the host, and characters received from the host are interpreted in the same way as they would by a DEC VT52 terminal. Thus, when the IRIS terminal program is loaded, the IRIS can be used as a standard terminal.

All graphical communication is initiated by the host. In the vast majority of cases, the host will send a graphics escape character (Control-P = DLE = 0x10) to the IRIS, followed by a graphical command, and after processing the command, the IRIS reverts immediately to its state as a standard terminal. In the discussion that follows, we call these two modes graphics mode and terminal mode. The IRIS is in graphics mode only between the time it receives the graphics escape character and the time that the graphical interaction is complete.

More complicated graphical transactions are possible — the graphics command may require the IRIS to return some information to the host while in graphics mode (the viewing matrix, some feedback information, etc.), and in this case, the IRIS remains in graphics mode until the entire transaction is complete. The IRIS never initiates a graphical transaction.

With one exception, while the IRIS is in graphics mode, no characters typed by the user are sent to the host. (The characters may be queued up by the Remote Graphics Library and sent to the host when a request arrives for them, but they are not sent as soon as they are typed.) If the user wishes to send a character to the host (for example to interrupt some run-away process), an escape character is typed (currently ESCAPE = ASCII 0x1B), and the next character typed is transmitted.

## F.2 The Remote Graphics Library

The Host Graphics Library has three parts—the graphics stubs, the communication routines, and the system interface routines. The graphics stubs are quite simple (in fact, they are generated automatically in different languages from a description of their parameters). For example, the `move(x, y, z)` command simply makes calls on the communication routines to send the graphics escape character, the move token, and then three floating-point numbers. The `getmatrix(&matrix)` command calls communication routines to send the escape character, the `getmatrix` token, and then to receive an array of floating point numbers. Other graphics commands may have slightly more complicated interactions, but all of them do their real work by calling on the communications primitives.

The communications library includes routines to send and receive bytes and arrays of bytes, *short*'s and arrays of *short*'s, *float*'s and arrays of *float*'s, and so on. Another routine sends the graphics escape character and a graphics command code. Finally, five routines in the IRIS Graphics Library must be handled in a special way (see Section F.4).

The system interface routines include a pair of functions to convert from the host's noating point format to IEEE and the reverse, one to nush the graphics stream (assuming it is buffered), routines to initialize the network, and others to change the communication mode (from 6 bit to 8 bit—described later).

Most of the rest of the document is a detailed description of the actions of the communication routines.

## F.3 Slow and Fast Communication

In order to be compatible with as many host/operating system combinations as possible, the IRIS Graphics Library makes the assumption that it may be possible to send only 6 bits of information per character. The host may require parity, and it may not be possible to send certain of the other 7 bit combinations. When communications start up, they do so in this mode.

If the host can be configured to send binary data, the command `setfastcom()` should be written to put the communication line in binary mode. In the example at the end of this document, the routine is quite simple, since UNIX is automatically in this mode. On other operating systems, this might involve a system call. `setslowcom()` has the opposite effect.

Both of these routines (and others) call the routine `netinit()` that sends 50 nulls to the terminal. 50 nulls is longer than any possible graphical transaction, so no matter what state the terminal was in, this will make sure that it is put in terminal mode. (If it was in graphics mode, half-finished with an array transmission, this will cause an error and force the IRIS into terminal mode.)

`putgchar()` is the routine that is called to send a character to the IRIS. In the attached example, it implements a simple buffer that guarantees that any graphics command will not be broken across a buffer boundary. Thus, at the end of each buffer transmission, the IRIS will be in terminal mode. `flushg()` is an internal routine to flush the host buffers out to the IRIS.

## F.4 Special IRIS Graphics Library Commands

The five routines `ginit()`, `gexit()`, `greset()`, `gfinish()`, and `gflush()` should be implemented as shown in the included example. The automatically generated graphics stubs include calls to `xginit()` and `xgreset()`.

## F.5 Echo Control

The routines `echoff()` and `echon()` should make the operating system dependent calls to turn off or on local echoing so that when the IRIS is sending information back to the host, the host will not try to echo it. This may not be necessary for operating systems that provide a non-echoing read.

## F.6 Floating Point Number Conversion

The routines `IEEE2F()` and `F2IEEE()` convert from the IEEE floating point format to the host's internal floating point format, and vice-versa.

## F.7 Data Transmission Routines

The remainder of the communications routines simply transmit and receive data. In order to accommodate different hosts, all data is assumed to be sent in the IRIS format. The IRIS has a 68000 or 68010 with its own peculiar byte orderings.

## F.8 IRIS Memory Organization

Suppose that the four hexadecimal values 0x00, 0x01, 0x02, and 0x03 are stored in the IRIS in such a way that 0x00 is stored in byte 0, 0x01 in byte 1, and so on. If we read address 0 as a word (16 bits), the IRIS will see the value 0x0001. The word at address 2 is 0x0203. The long word (32 bits) at address 0 is 0x00010203. (On a DEC-11 machine, for example, the words at 0 and 2 would be 0x0100 and 0x0302, and the long at address 0 would be 0x03020100.)

## F.9 Six-bit and Eight-bit Transmission

The six-bit values are guaranteed to be standard ASCII printing values between space (0x20) and '_' (0x60). To send a six-bit value, add 0x20 to it. To interpret a received character as a six-bit value, subtract 0x20, and mask off all but the 6 low-order bits. Eight-bit values are simply transmitted as is. In six-bit mode, all data items are transmitted 6 bits at a time, beginning with the 6 low-order bits. For example, the word 0x1234 is sent as the following sequence of bytes: 0x54 = 'T', 0x28 = '(', and finally 0x21 = '!'. In binary, 0x1234 = 0001001000110100. The 6 bit chunks (from right to left) are: 110100, 001000 and 000001 (the last is padded with 2 zeroes). We then add 0x20 to each, giving: 0x54, 0x28 and 0x21.

Eight-bit values are sent beginning with the high-order bytes so that the IRIS can simply slap them into memory in the order that they arrive.

## F.10 Protocols

`gcmd()` sends an eight-bit graphics code, preceded by the graphics escape character (0x10).

`sendo()` and `sendb()` send a boolean and a character (byte), respectively. The IRIS considers a boolean to be exactly the same as a character. where 0x00 is false, and 0x01 is true. Thus the two routines in the example are identical.

`sends()`, `sendl`, and `sendf` send a short (16 bits), a long (32 bits), and a float (32 bits, IEEE format), respectively.

The corresponding routines to receive a boolean, a byte, a short, a long, and a float are `recO`, `recB`, `recS`, `recL`, and `recF`, respectively. All are implemented in a straightforward manner.

The most complicated protocols are those associated with the sending and receiving arrays. Consider first the routines that send arrays to the IRIS. The routines are `sendos()`, `sendbs()`, `sendqs()`, `sends()`, `sendls()`, and `sendfs()`. These routines send, respectively, an array of boolean's, byte's, Fontchar's, and arrays of short's, long's and float's.

All of the commands take the array as the first argument and the number of items of the appropriate type to send as the second argument. The IRIS has a single receive array routine, and it expects an array of longs. Thus, the host routine has the responsibility for arranging the information into longs so that when placed in the IRIS, it will be interpreted correctly. The only tricky one is `sendqs()`—the routine that sends an array of Fontchars. Each of the routines that sends arrays figures out the minimal number of longs that must be sent to transmit all of the information, and sends them using the `sendl()` primitive described above. Every 8 longs, a '.' is sent for synchronization. Thus a long array transmission consists of groups of 48 bytes of information (in the six-bit case), followed by a synchronization character. (This is why 50 nulls are sent during initialization.)

`sendc()` sends a character string. On the IRIS, character strings are just null-terminated arrays of characters, so the example code simply sends it exactly as if it were an array of bytes. Other languages and operating systems may have to massage strings a bit more before sending them.

The routines to receive arrays include `recOs()`, `recBs()`, `resSs()`, `recLs()`, and `recFs()` to receive arrays of boolean's, byte's, short's, long's, and float's, respectively. Each takes as a parameter an array for the results. The first thing each routine does is to receive a long telling the number of items of that type will be sent. Next comes the graphics escape character, followed by enough longs (all sent 6 bits at a time) to get the entire array. Every eighth item (beginning with the first) is followed by a carriage-return (0x0d). After receiving the carriage-return, the host replies with a '.' for synchronization. This continues to the end of the array, and the host must be responsible for writing the exact number of items into memory (for example, if only one byte is sent, the IRIS will send it as the first of a long, and the host must write exactly one byte into the user's array).

## F.11 Examples

This section includes a number of examples against which the routines above can be checked.

### IEEE Format

Following are a number of floating point numbers expressed in IEEE floating point format.[1]

1. "A Proposed Standard for Binary Floating Point Arithmetic", Computer, March, 1981.

| | |
|---|---|
| 0.0 | 0x00000000 |
| 1.0 | 0x3f800000 |
| 2.0 | 0x40000000 |
| -1.0 | 0xbf800000 |
| -2.0 | 0xc0000000 |
| 63.0 | 0x427c0000 |
| -63.0 | 0xc27c0000 |
| 0.5 | 0x3f000000 |
| -0.5 | 0xbf000000 |
| 0.125 | 0x3e000000 |
| -0.125 | 0xbe000000 |
| 0.0078125 | 0x3c000000 |
| -0.0078125 | 0xbc000000 |
| 0.000000001 | 0x3089705f |
| -0.000000001 | 0xb089705f |
| 1024.0 | 0x44800000 |
| -1024.0 | 0xc4800000 |
| 1000000.0 | 0x49742400 |
| -1000000.0 | 0xc9742400 |
| 0.333333333333 | 0x3eaaaaaa |
| -0.333333333333 | 0xbeaaaaaa |

## Transmission Protocols

What follows are a number of small programs that call a variety of routines in the Remote Graphics Library. As a result of these calls, a certain sequence of bytes should be put out on the serial line or network.

The programs below are written in C, but it should be simple to transliterate them to another language. Similarly, the files *io.c* and *lib.c* will be *io.f* and *lib.f* in FORTRAN, etc. In most cases, the programs do not do anything sensible graphically; they merely illustrate the graphics protocol in as many situations as possible.

### Example 1

```
main()
{
      ginit();
      color(RED);
      clear();
      gexit();
}
```

The program above causes the following bytes to be sent out:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 |
| \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 |
| \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 | \0 |
| \0 | \0 | 020 |  |  | 020 | 3 | ! | 020 | B |  | ! |  |  | 020 | 8 |
|  | end |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

`ginit()` (in the file *io.c*) calls `netinit()` which first sends out 50 nulls, and then makes a call to `setslowcom()`. `setslowcom()` calls `gcmd(0)`. Then `ginit()` calls `xginit()` which calls `gcmd(83)`. `gcmd()` puts a 12 bit number out on the serial line 6 bits at a time, preceded by the graphics escape character which is ASCII ^P = DLE = 16 (decimal) = 020 (octal). Each six-bit pattern has 040 (octal) added to it so that it is in the range of printable characters.

In the example above, the nulls are represented by \0, the DLEs by 020, and every other character by itself. The blanks in the table are ASCII spaces (040 octal). The string "end" marks the end of the table. In the case above it is obviously necessary, since the final character sent is a space. Future examples will omit the first three lines of nulls, since `ginit()` will always be called first, so every example will begin with 2 nulls. Finally, note that there are exactly 16 entries in each line - this is important if the lines happen to end with spaces.

In this example only we shall explain one command in some detail; almost all of the others can be worked out in the same way from the files *lib.c* and *io.c*.

Consider the command `color(RED)`. *RED* is the constant 1, so this is equivalent to the call `color(1)`. In `lib.c`, the color routine looks something like this:

```
color(arg1)
short arg1;
{
        gcmd(34);
        sends(arg1);
}
```

`gcmd()` sends a DLE (= 020), followed by 34 (= 100010 binary) sent in two six-bit chunks. First the low-order 6 bits are sent after adding on 100000 (binary) giving 1000010 (binary) = 0102 (octal) = ASCII 'B'. The six high-order bits are all zero, so adding 100000 (binary) to it gives an ASCII' '(space). `gcmd()` thus sends DLE, then 'B', then a space.

`sends()` sends a 1 as 2 six-bit chunks followed bv a two-bit chunk (low-order bits first). The chunks are treated exactly the same way as the `gcmd()` does, and the net result is a '!' followed by two spaces.

`gexit()` sends nothing—it just causes any host-buffered characters to be flushed.

**Example 2**

```
main()
{
      ginit();
      pushmatrix();
      viewport(0, 1023, 0, 767);
      rotate(900, 'x');
      pnt(0.0, 1.0. -10.0);
      gexit();
}
```

yields:

```
     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16
------------------------------------------------------------------------------------------------
    \0    \0   020               020     3     !   020     &     "   020     S     "
           _     /                             _     +          020     ;     "     $     .
     X     !   020     U     !           #                                                @     _
                             (     !         end
```

**Example 3**

```
float matrix[4] [4] = {1.0, 0.0, 0.0, 0.0,
                       0.0, 2.0, 0.0, 0.0,
                       0.0, 0.0, 3.0, 0.0,
                       0.0, 0.0, 0.0, 4.0};

main()
{
      ginit();
      loadmatrix(matrix);
      gexit();
}
```

yields:

```
     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16
------------------------------------------------------------------------------------------------
    \0    \0   020               020     3     !   020     -     !           !
           .                       @     _

                                   !
                 .
                 0           !

     @           !  end
```

Note the periods sent by `dosync()` every 8 longs in an array transfer.

**Example 4**

```
main()
{
        ginit();
        charstr("abcd");
        charstr("abcde");
        charstr("abcdef");
        charstr("abcdefg");
        charstr("abcdefgh");
        gexit();
}
```

yields:

```
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16

   \0   \0  020            020    3    !  020    3         %
         .    D    -    F    8    A    !    C    )    6    8            020    3
         &                             .    D    -    F    8    A    !    B    %
    &         E    !  020    3         '                             .    D    -
    F    8    A    !    A    !    @    9    E    !  020    3         (
                   .    D    -    F    8    A    !         <    F    9    E    !
  020    3         )                             .    D    -    F    8    A    !
    H    =    F    9    E    !    Z    I    G    >            end
```

**Example 5**

In this example for the first time, the IRIS returns some values.

```
main()
{
        ginit();
        getmap();
        gexit();
}
```

yields:

```
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16

   \0   \0  020            020    3    !  020    )    !  end
```

Then the IRIS responds with:

```
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16

                      012  end
```

The 012 (octal) above is an ASCII line-feed.

**Example 6**

```
main()
{
      Matrix mm;
      ginit();
      getmatrix(mm);
      gexit();
}
```

yields:

```
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
---------------------------------------------------------------------------------
   \0   \0  020            020    3    !  020    *    !  end
```

Got from IRIS:

```
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
---------------------------------------------------------------------------------
         0                                                     [       012  end
```

Sent to IRIS:

```
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
---------------------------------------------------------------------------------
    .  end
```

Got from IRIS:

```
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
---------------------------------------------------------------------------------
                                             J    J    J    *    [

  012  end
```

Sent to IRIS:

```
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
---------------------------------------------------------------------------------
    .  end
```

Got from IRIS:

```
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
---------------------------------------------------------------------------------
                                   ^    _    _    ?    _    "
              ^    _    [    ?    _    "    J    J    Z    ?    _    "
                                        @    _         012  012  end
```

**Example 7**

```
Colorindex colors[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                       13, 14, 15, 16, 17, 18, 19, 20};

main()
{
      ginit();
      cmov2i(100, 100);
      writepixels(20, colors);
      cmov2i(100, 100);
      readpixels(20, colors);
      gexit();
}
```

yields:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
----------------------------------------------------------------
```

Sent to IRIS:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
----------------------------------------------------------------
  \0  \0 020         020   3   ! 020   @       D   !
       D   !                 020   V   "   4           H
               .   "       0               $       P
   &       0   !           (       P   !           *       0   "
           '       P   "           .       0   #           0
   P   #           .   2       0   $           4       P   $
     020   @       D   !                   D   !
 020   /   "   4         end
```

Got from IRIS:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
----------------------------------------------------------------
       4               4
         012 end
```

Sent to IRIS:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
----------------------------------------------------------------
   . end
```

Got from IRIS:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
----------------------------------------------------------------
   $       P               &       0   !           (       P   !
           *       0   "           ,       P   "           .
   0   #           0       P   #           2       0   $
 012 end
```

Sent to IRIS:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16

   . end
```

Got from IRIS:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16

   4       P   $         012 012 end
```

**Example 8**

```
Colorindex red[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
Colorindex green[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
Colorindex blue[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

main()
{
        ginit();
        RGBmode();
        gconfig();
        cmov2i(100, 100);
        writeRGB(10, red, green, blue);
        cmov2i(100, 100);
        readRGB(10, red, green, blue);
        gexit();
}
```

yields:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
```

Sent to IRIS:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16

  \0  \0 020         020   3   ! 020   9   " 020   Y     020   @
       D   !                   D   !                   020   W   "
   *           4                       .           0
   $       P               &       0   !   "       (       P   !
           *       0   "           4                       .   "
       0               $       P               &       0   !
       (       P   !           *       0   "       4
               .   "       0               $       P
   &       0   !           (       P   !           *       0   "
         020   @       D   !                   D   !
     020   0   "   *         end
```

Got from IRIS:

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16

       *               *                           "       0
         012 end
```

Sent to IRIS:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| . | end | | | | | | | | | | | | | | |

Got from IRIS:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $ | | P | | | | & | | 0 | ! | | | ( | | P | ! |
| | | * | | 0 | " | | | 012 | | * | | | | | |
| | " | | 0 | | | | 012 | end | | | | | | | |

Sent to IRIS:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| . | end | | | | | | | | | | | | | | |

Got from IRIS:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $ | | P | | | | & | | 0 | ! | | | ( | | P | ! |
| | | * | | 0 | " | | | 012 | | * | | | | | |
| | | | | | | | 012 | end | | | | | | | |

Sent to IRIS:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| . | end | | | | | | | | | | | | | | |

Got from IRIS:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 012 | 012 | end | | | | | |

| Number | 1st Char | 2nd Char | Name | Number | 1st Char | 2nd Char | Name |
|---|---|---|---|---|---|---|---|
| 0 | `' '` | `' '` | `xsetslowcom` | 43 | `'K'` | `' '` | `deftexture` |
| 1 | `'!'` | `' '` | `xsetfastcom` | 44 | `'L'` | `' '` | `delete` |
| 2 | `'"'` | `' '` | `gversion` | 45 | `'M'` | `' '` | `delobj` |
| 3 | `'#'` | `' '` | `gdownload` | 46 | `'N'` | `' '` | `doublebuffer` |
| 4 | `'$'` | `' '` | `pagecolor` | 47 | `'O'` | `' '` | `draw` |
| 5 | `'%'` | `' '` | `textcolor` | 48 | `'P'` | `' '` | `draw2` |
| 6 | `'&'` | `' '` | `textport` | 49 | `'Q'` | `' '` | `draw2i` |
| 7 | `'''` | `' '` | `arc` | 50 | `'R'` | `' '` | `drawi` |
| 8 | `'('` | `' '` | `arcf` | 51 | `'S'` | `' '` | `editobj` |
| 9 | `')'` | `' '` | `arcfi` | 52 | `'T'` | `' '` | `endpick` |
| 10 | `'*'` | `' '` | `arci` | 53 | `'U'` | `' '` | `endselect` |
| 11 | `'+'` | `' '` | `attachcursor` | 54 | `'V'` | `' '` | `finish` |
| 12 | `','` | `' '` | `backbuffer` | 55 | `'W'` | `' '` | `font` |
| 13 | `'-'` | `' '` | `bbox` | 56 | `'X'` | `' '` | `frontbuffer` |
| 14 | `'.'` | `' '` | `bbox2` | 57 | `'Y'` | `' '` | `gconfig` |
| 15 | `'/'` | `' '` | `bbox2i` | 58 | `'Z'` | `' '` | `genobj` |
| 16 | `'0'` | `' '` | `bboxi` | 59 | `'['` | `' '` | `gentag` |
| 17 | `'1'` | `' '` | `blink` | 60 | `'\'` | `' '` | `getbuffer` |
| 18 | `'2'` | `' '` | `callobj` | 61 | `']'` | `' '` | `getbutton` |
| 19 | `'3'` | `' '` | `charstr` | 62 | `'^'` | `' '` | `getcmmode` |
| 20 | `'4'` | `' '` | `circ` | 63 | `'_'` | `' '` | `getcolor` |
| 21 | `'5'` | `' '` | `circf` | 64 | `' '` | `'!'` | `getcursor` |
| 22 | `'6'` | `' '` | `circfi` | 65 | `'!'` | `'!'` | `getdepth` |
| 23 | `'7'` | `' '` | `circi` | 66 | `'"'` | `'!'` | `getdisplaymode` |
| 24 | `'8'` | `' '` | `clear` | 67 | `'#'` | `'!'` | `getfont` |
| 25 | `'9'` | `' '` | `clearhitcode` | 68 | `'$'` | `'!'` | `getheight` |
| 26 | `':'` | `' '` | `clipline` | 69 | `'%'` | `'!'` | `gethitcode` |
| 27 | `';'` | `' '` | `clippnt` | 70 | `'&'` | `'!'` | `getlsbackup` |
| 28 | `'<'` | `' '` | `clippoly` | 71 | `'''` | `'!'` | `getlstyle` |
| 29 | `'='` | `' '` | `closeobj` | 72 | `'('` | `'!'` | `getlwidth` |
| 30 | `'>'` | `' '` | `cmov` | 73 | `')'` | `'!'` | `getmap` |
| 31 | `'?'` | `' '` | `cmov2` | 74 | `'*'` | `'!'` | `getmatrix` |
| 32 | `'@'` | `' '` | `cmov2i` | 75 | `'+'` | `'!'` | `getobjfont` |
| 33 | `'A'` | `' '` | `cmovi` | 76 | `','` | `'!'` | `getplanes` |
| 34 | `'B'` | `' '` | `color` | 77 | `'-'` | `'!'` | `getresetls` |
| 35 | `'C'` | `' '` | `cursof!` | 78 | `'.'` | `'!'` | `getscrmask` |
| 36 | `'D'` | `' '` | `curson` | 79 | `'/'` | `'!'` | `gettexture` |
| 37 | `'E'` | `' '` | `curve` | 80 | `'a'` | `'!'` | `getvaluator` |
| 38 | `'F'` | `' '` | `curveit` | 81 | `'1'` | `'!'` | `getviewport` |
| 39 | `'G'` | `' '` | `defcursor` | 82 | `'2'` | `'!'` | `getwritemask` |
| 40 | `'H'` | `' '` | `deflinestyle` | 83 | `'3'` | `'!'` | `xginit` |
| 41 | `'I'` | `' '` | `defobjfont` | 84 | `'4'` | `'!'` | `xgreset` |
| 42 | `'J'` | `' '` | `defrasterfont` | 85 | `'5'` | `'!'` | `gRGBcolor` |

| Number | 1st Char | 2nd Char | Name | Number | 1st Char | 2nd Char | Name |
|---|---|---|---|---|---|---|---|
| 86 | '6' | '!' | gRGBcursor | 129 | '!' | '"' | polyi |
| 87 | '7' | '!' | gRGBmask | 130 | '"' | '"' | popattributes |
| 88 | '8' | '!' | insert | 131 | '#' | '"' | popmatrix |
| 89 | '9' | '!' | isobj | 132 | '$' | '"' | popviewport |
| 90 | ':' | '!' | istag | 133 | '%' | '"' | pushattributes |
| 91 | ';' | '!' | keyboard | 134 | '&' | '"' | pushmatrix |
| 92 | '<' | '!' | linewidth | 135 | ''' | '"' | pushviewport |
| 93 | '=' | '!' | loadmatrix | 136 | '(' | '"' | qbutton |
| 94 | '>' | '!' | lookat | 137 | ')' | '"' | qenter |
| 95 | '?' | '!' | lsbackup | 138 | '*' | '"' | qkeyboard |
| 96 | '@' | '!' | makeobj | 139 | '+' | '"' | qread |
| 97 | 'A' | '!' | maketag | 140 | ',' | '"' | qreset |
| 98 | 'B' | '!' | mapcolor | 141 | '-' | '"' | qtest |
| 99 | 'C' | '!' | mapw | 142 | '.' | '"' | qvaluator |
| 100 | 'D' | '!' | mapw2 | 143 | '/' | '"' | readpixels |
| 101 | 'E' | '!' | modify | 144 | '0' | '"' | readRGB |
| 102 | 'F' | '!' | move | 145 | '1' | '"' | rect |
| 103 | 'G' | '!' | move2 | 146 | '2' | '"' | recti |
| 104 | 'H' | '!' | move2i | 147 | '3' | '"' | rectfi |
| 105 | 'I' | '!' | movei | 148 | '4' | '"' | recti |
| 106 | 'J' | '!' | multimap | 149 | '5' | '"' | replace |
| 107 | 'K' | '!' | multmatrix | 150 | '6' | '"' | resetls |
| 108 | 'L' | '!' | noise | 151 | '7' | '"' | RGBcolor |
| 109 | 'M' | '!' | objfont | 152 | '8' | '"' | RGBcursor |
| 110 | 'N' | '!' | objstr | 153 | '9' | '"' | RGBmode |
| 111 | 'O' | '!' | onemap | 154 | ':' | '"' | RGBwritemask |
| 112 | 'P' | '!' | ortho | 155 | ';' | '"' | rotate |
| 113 | 'Q' | '!' | ortho2 | 156 | '<' | '"' | scale |
| 114 | 'R' | '!' | perspective | 157 | '=' | '"' | screenpnt |
| 115 | 'S' | '!' | pick | 158 | '>' | '"' | scrmask |
| 116 | 'T' | '!' | picksize | 159 | '?' | '"' | select |
| 117 | 'U' | '!' | pnt | 160 | '@' | '"' | setbutton |
| 118 | 'V' | '!' | pnt2 | 161 | 'A' | '"' | setcursor |
| 119 | 'W' | '!' | pnt2i | 162 | 'B' | '"' | setdepth |
| 120 | 'X' | '!' | pnti | 163 | 'C' | '"' | setlinestyle |
| 121 | 'Y' | '!' | polarview | 164 | 'D' | '"' | setmap |
| 122 | 'Z' | '!' | polf | 165 | 'E' | '"' | setplanes |
| 123 | '[' | '!' | polf2 | 166 | 'F' | '"' | settexture |
| 124 | '\' | '!' | polf2i | 167 | 'G' | '"' | setvaluator |
| 125 | ']' | '!' | polfi | 168 | 'H' | '"' | singlebuffer |
| 126 | '^' | '!' | poly | 169 | 'I' | '"' | strwidth |
| 127 | '_' | '!' | poly2 | 170 | 'J' | '"' | swapbuffers |
| 128 | ' ' | '"' | poly2i | 171 | 'K' | '"' | swapinterval |

| Number | 1st Char | 2nd Char | Name |
|---|---|---|---|
| 172 | 'L' | '"' | sync |
| 173 | 'M' | '"' | tie |
| 174 | 'N' | '"' | transform |
| 175 | 'O' | '"' | translate |
| 176 | 'P' | '"' | unqbutton |
| 177 | 'Q' | '"' | unqkeyboard |
| 178 | 'R' | '"' | unqvaluator |
| 179 | 'S' | '"' | viewport |
| 180 | 'T' | '"' | window |
| 181 | 'U' | '"' | writemask |
| 182 | 'V' | '"' | writepixels |
| 183 | 'W' | '"' | writeRGB |
| 184 | 'X' | '"' | tpon |
| 185 | 'Y' | '"' | tpoff |
| 186 | 'Z' | '"' | pagewritemask |
| 187 | '[' | '"' | textwritemask |
| 188 | '\' | '"' | xgexit |
| 189 | ']' | '"' | clkon |
| 190 | '^' | '"' | clkoff |
| 191 | '_' | '"' | lampon |
| 192 | ' ' | '#' | lampoff |
| 193 | '!' | '#' | setbell |
| 194 | '"' | '#' | ringbell |

# Appendix G: Generic Distribution Tape

The generic IRIS Terminal distribution tape contains example files for implementing the IRIS Graphics Library on unsupported operating systems. In addition, a serial download program like *dliris* may also be needed. See the source file for *dliris.c* on the standard distribution tape or *dliris.f* on the generic distribution tape.

| **Tape Format** | |
|---|---|
| 1600 bpi | 33 files |
| 9 track | No tabs |
| ASCII format | Unlabeled |
| 1 tape mark between files | BLKSIZE = LRECL = 80 |

Table G-1: Generic Distribution Tape Format

*C Include Files*

- *device.h* contains symbolic name definitions for devices, e.g. mouse and keyboard buttons.
- *get.h* contains definitions for values returned by the Graphics Library get commands.
- *gl.h* contains default values for colors, screen boundaries, etc. for the Graphics Library.

*C Demonstration Programs*

- *sqiral.c* draws a square spiral.
- *track.c* displays a box which can be moved by the mouse.

*UNIX/C Graphics Library Source Files*

- *io.c* is a UNIX/C implementation of the routines that communicate between the IRIS Terminal and the host. See Appendix F.
- *lib.c* contains function definitions for the Graphics Library.
- *rpc.h* contains remote procedure call command definitions for *io.c*. See Appendix F.

*FORTRAN Include Files*

- *device.h* contains symbolic name definitions for devices, e.g. mouse and keyboard buttons.
- *get.f* contains definitions for values returned by the Graphics Library *get* commands.
- *gl.f* contains default values for colors, screen boundaries, etc. for the Graphics Library.

*FORTRAN Demonstration Programs*

- *planets.f* models a solar system. See the file *planets.doc*.
- *sqiral.f* draws a square spiral.
- *track.f* displays a box which can be moved by the mouse.
- *star.f* displays a series of stars.
- *curve.f* draws curves.
- *floats.f* fills the IRIS Terminal screen with randomly colored pixels.

*UNIX/FORTRAN Graphics Library source Files*

- *fsys.c* contains system-dependent I/O routines.
- *io.h* contains system-dependent I/O definitions.
- *io.f* is a UNIX/FORTRAN implementation of the routines that communicate between the IRIS Terminal and the host. See Appendix F.
- *lib.f* contains function definitions for the Graphics Library.
- *rpc.h* contains remote procedure call command definitions for *fsys.c*. See Appendix F.

*Generic Include Files*

- *device.prim* contains device definitions to generate an include file like *device.h* for different languages.
- *get.prim* contains get command definitions to generate an include file like *get.h* for different languages.
- *gl.prim* contains Graphics Library definitions to generate an include file like *gl.h* for different languages.

*Generic Graphics Library Source Files*

- *lib.prim* contains language-independent descriptions of what data is transferred to and from the IRIS Terminal when each graphics primitive is executed by a program running on the host. It is used to generate *lib.c* and *lib.f*.

- *remprom.prim* contains PROM communication detinitions to generate an include file like *remprom.h* for different languages. These definitions are used to create a serial download program like *dliris*.
- *rpc.prim* contains remote procequre call command definitions to generate an include file like *rpc.h* for different languages. Thhese definitions are used in the I/O routines that must be implemented for each host operating system. See Appendix F.

*Miscellaneous*

- *escape.doc* contains a list of the Graphics Library functions and their associated escape sequences. See Appendix F.
- *planets.doc* describes the *planets* demonstration program.
- *protocol* is a document that describes the implementation of the routines that communicate between the IRIS Terminal and the host. See Appendix F.
- *Generic.source* is for internal identification purposes.
- *Generic.tbl* contains a description of the files on the distribution tape.
- *Tape.list* is for internal identification purposes.
- *Tape.source* is for internal identification purposes.