Installing and Maintaining HP BASIC/UX 6.2

HEWLETT PACKARD

HP Part No. E2040-90002
Printed in USA

Printing History

First Edition—August 1991

Contents

1

Before You Install

This chapter provides the following information:

- How HP BASIC/UX and HP-UX are related.
- What is included with BASIC/UX.
- An overview of the installation process.

How HP BASIC/UX and HP-UX are Related

As the name implies, HP BASIC/UX runs in the HP-UX environment. HP BASIC/UX 6.2 provides the functionality of version 6.2 of HP Series 200/300 workstation BASIC (called "HP BASIC/WS 6.2") running as a process "on top of" the HP-UX operating system.

HP-UX is the "Core" System

HP-UX is Hewlett-Packard's implementation of the UNIX® operating system. (UNIX is a registered trademark of UNIX System Laboratories in the U.S.A. and other countries.)

Like any operating system, HP-UX organizes and schedules processes for your computer system. However, HP-UX is a **multi-tasking** operating system — it can run several processes at the same time. HP-UX is also a **multi-user** operating system — it can handle several users at the same time.

For further information about the HP-UX environment, refer to your HP-UX documentation, starting with *A Beginner's Guide to HP-UX*.

BASIC/UX Runs "On Top Of" HP-UX

HP BASIC/UX allows you to work in the BASIC environment while taking advantage of HP-UX enhancements. HP BASIC/UX can be thought of as an application that runs on top of HP-UX. HP-UX controls the resources and coordinates the internal processes that HP BASIC/UX initiates.

Having HP-UX as the "core" allows multiple HP BASIC/UX processes — several people can run HP BASIC/UX on the same system at the same time.

HP BASIC/UX Compatibility With HP-UX

HP BASIC/UX 6.2 runs only on HP-UX version 8.0. **It will not run on previous HP-UX releases.** For details on running HP BASIC/UX with future releases of HP-UX, please contact your local HP Sales Representative.

The following table shows which version of BASIC/UX is compatible with each version of HP-UX, as of the time of publication:

BASIC/UX Compatibility With HP-UX

HP-UX Version	Compatible BASIC/UX Version
6.2	5.5
6.5	5.51
7.x	5.52
8.0	6.2

What's in the Box

The HP BASIC/UX 6.2 language system includes these components:

- An HP BASIC/UX 6.2 tape (unless you have a pre-installed system).
- The HP BASIC 6.2 manuals.
- The HP BASIC keyboard overlays for the ITF keyboard.
- The "Contents List" packed in the shipping carton lists all components. You should inventory these items. If any items are damaged or missing, please contact your HP Sales Representative.

Setting Up Your Computer System

If your computer system has not already been set up, you should unpack it and set it up now. Connect the display system, external disk drives (if present), and other peripherals to the computer SPU (System Processor Unit) following the directions included with each hardware component.

Now you are ready to install the software.

Do You Have a Pre-Installed BASIC/UX System?

If you have purchased an HP 9000 Model 382 or Model R/382 *bundled* BASIC/UX system, the software is pre-installed on the hard disk. The pre-installed software includes HP-UX 8.0, HP BASIC/UX 6.2, X11 Windows, and HP VUE. You can skip the software installation procedures in chapter 2 and 3, and go on to "Starting a BASIC/UX System With HP VUE" in chapter 4.

Caution Pre-installed BASIC/UX systems do not include a BASIC/UX tape. To protect against loss of software, you should back up your system as soon as possible after starting it.

Overview of HP-UX and BASIC/UX Installation

Before you can use HP-UX or BASIC/UX, you must *install* the software. Software installation is a process that copies the system from tape and "builds" a working system on the hard disk.

Note

If you have a pre-installed system, HP-UX 8.0 and HP BASIC/UX 6.2 are both installed on the hard disk. No software installation is necessary.

To install your BASIC/UX system, you will need to perform the following tasks:

1. Set up your hardware (as outlined previously).
2. Install HP-UX 8.0 from the tapes supplied with HP-UX (as outlined in chapter 2). If you are using the X Windows and HP VUE environments, configure them as part of the HP-UX installation.

Note

If HP-UX version 8.0 is already installed on your system, you can skip this step. However, if an earlier version is installed, you will have to update HP-UX by installing version 8.0 — HP BASIC/UX 6.2 will not run on HP-UX versions earlier than 8.0.

3. Install HP BASIC/UX 6.2 from the tape supplied with BASIC/UX (as outlined in chapter 3).
4. Test the installed system (as outlined in chapter 4).

Installing HP-UX is a time-consuming process because of the volume of the HP-UX system. Several hours may be required.

Where to Start the Installation

Where you start the installation depends on your current configuration:

- If you have pre-installed software (an HP 9000 Model 382 or Model R/382 "bundled" BASIC/UX system), HP-UX 8.0 and BASIC/UX 6.2 are already installed. Skip to chapter 4 and test the installation.
- If you have a new system with no software installed, go to chapter 2 and install HP-UX 8.0. Then go to chapter 3 and install BASIC/UX 6.2.
- If you have a system with HP Series 200/300 workstation BASIC (BASIC/WS) installed, go to chapter 2. Back up your files and then install HP-UX 8.0. Then go to chapter 3 and install BASIC/UX 6.2.
- If you have a system with a previous version of HP-UX already installed, go to chapter 2 and update HP-UX to version 8.0. Then go to chapter 3 and install BASIC/UX 6.2.
- If you have a system with HP-UX 8.0 already installed, skip to chapter 3 and install BASIC/UX 6.2.

Additional BASIC/UX Considerations

Disk Drive/Instrument Control Interface Conflicts

To avoid conflicts between disk drives and instrument-control devices, do not connect both classes of devices to the same interface (for example, an HP-IB card at select code 7).

Caution

Do not connect any disk drives to an interface card that is being used for instrument control. Disk I/O is unreliable under these circumstances. If the interface card is locked, the kernel CS80 drivers may cause the system to hang. It is sometimes possible to correct this situation by unlocking the interface. Otherwise, you will have to reboot HP-UX.

Installing BASIC/UX on Heterogeneous Clusters

BASIC/UX can be installed on a heterogeneous cluster (an HP 9000 Series 600, Series 700, or Series 800 server with HP 9000 Series 300 clients). However, you cannot *run* BASIC/UX from the Series 600/700/800 computer — only from a Series 300 client.

Handling of AUTOLOCK and AUTOBURST During BASIC/UX Boot

If another process has the HP-IB interface locked, AUTOLOCK, and AUTOBURST for HP-IB both timeout on power up and display:

```
AUTOLOCK failed
```

or

```
AUTOBURST failed
```

The most common instance is when multiple BASIC/UX processes are started with AUTOLOCK or AUTOBURST on an HP-IB interface.

Local and Global rmbrc File Access

BASIC/UX no longer attempts to lock the global and local `rmbrc` files. The issue of security is now left up to the user. It is suggested that the permissions be set to 644 or 444 and the global `rmbrc` file should be owned by root to prevent unauthorized access to these files during the boot up of BASIC/UX. The only requirement of permissions is that the user have read permission on the global and local `rmbrc` files.

2

Installing HP-UX

Before you install HP BASIC/UX 6.2, you will need to install HP-UX version 8.0. *HP BASIC/UX 6.2 will not run on previous versions of HP-UX.*

Note

If you have already installed HP-UX 8.0 on your hard disk, you can skip this chapter and install BASIC/UX 6.2 as described in chapter 3.

If you purchased a "bundled" BASIC/UX system with pre-installed software (e.g. the HP 9000 Model 382 or Model R/382 with the BASIC/UX option), you can skip this chapter and chapter 3. HP-UX 8.0, HP BASIC/UX 6.2, X11 Windows, and HP VUE have been pre-installed and pre-configured on such systems. Skip to "Starting a BASIC/UX System With HP VUE" in chapter 4.

This chapter identifies the steps you will need to perform to install HP-UX on to your hard disk. However, this chapter does not describe the process in detail — you will need to refer to the *Series 300 HP-UX Installation* manual for further information.

This chapter does provide information specific to the HP-UX configuration needed for BASIC/UX. For example, this chapter will help you select file sets (see "Selecting File Sets") and determine swap space (see "Determining Swap Space").

It is helpful to read this chapter *before* you start to install HP-UX.

Preparing to Install HP-UX

Before you start the installation process, there are several tasks you should complete that save time during installation.

Books You Need

The following manuals are your primary references for the installation process:

Manual	Type of Information It Contains
Series 300 HP-UX Installation manual	How to install HP-UX on to a disk, and how to install and configure HP-UX diskless clusters.
Series 300 HP-UX System Administration Tasks manual	How to select file sets for disks that can't hold the entire system; how to customize HP-UX; how to configure HP-UX diskless clusters.
This manual: *Installing and Maintaining BASIC/UX 6.2*	A guide to HP-UX installation and HP BASIC/UX installation.

How Long Does Installation Take?

HP-UX is a comprehensive set of computing tools and utilities that exceeds those provided by HP BASIC. This along with the size of disk required for HP-UX causes installation to be a time-consuming process. Allow several hours for the installation process. However, many of the processes won't require your full time attention. You may wish to read some of the references listed in "Books You Need" during such periods.

Use System Defaults, Except for Special Cases

The system provides **default** parameters which have been chosen as the optimum values for most cases. You should use the system defaults unless you require a special configuration. Setting up such a special configuration will require considerable knowledge of the effect of each parameter (refer to the *Series 300 HP-UX System Administration Tasks* manual). For example, when you select swap space, you should choose the system default (unless you are adding several applications in addition to HP BASIC/UX).

Setting Up Hardware

If you haven't set up your hardware yet (computer, disk and peripherals), you must do so at this time. See the *Series 300 HP-UX Installing Peripherals* manual or your hardware's installation manual. Be sure to record values such as each peripheral's select code and bus address. You need these when you configure the peripheral with HP-UX later (see the chapter "Maintaining the BASIC/UX System" for more information).

- If you are configuring an HP-UX diskless cluster, read the *Series 300 HP-UX Installation* manual.
- If you see unfamiliar terms in any of the manuals used during installation, read the *HP-UX System Administration Concepts* manual.

If Your Disk Has BASIC Workstation On It: Back Up Files

If you are planning to install HP BASIC/UX onto a disk that currently has HP BASIC Workstation installed on it, you must back up on tape any valuable files so you can transfer them to the new system.

1. Go to the "Using the BACKUP Utility" chapter in the *Installing and Maintaining BASIC/WS* manual for your version of HP Series 200/300 workstation BASIC.
2. Follow the steps in that chapter to run the BACKUP utility.
3. Select `Backup selected files` (described in the "Using BACKUP Utility Options" section).
4. After backing-up files, keep the media handy until after you install HP-UX and HP BASIC/UX.

Reading About Installation

If you haven't read the section "What Is Installation?" in chapter 1, "Before You Install," you might do so now. It describes the installation process.

Reading Overview of HP-UX Installation

For a brief discussion of the HP-UX installation procedure, read the *Series 300 HP-UX Installation* manual.

Selecting File Sets

The HP-UX system contains many tools and utilities, not all of which you might need on your system. When you select file sets during HP-UX installation you can choose *not* to install some of the optional file sets (thus saving disk space).

- If you are installing on a hard disk with at least 200 MB (megabytes) of capacity, you can install all of HP-UX by following the system defaults. (With an HP-UX 8.0 run-time system plus BASIC/UX 6.2 installed on a 200 MB hard disk, you will have about 40 – 45 MB of free capacity on the disk.) Go on to the next step (unless you wish to optimize).
- If you are installing on a hard disk with less than 200 MB, or you wish to select file sets to optimize disk space, you must read about file sets and choose the ones you wish to install. Read the *Series 300 HP-UX Installation* manual and the *Series 300 HP-UX System Administration Tasks* manual for information on selecting file sets.

Determining Swap Space

For HP BASIC/UX, the installation procedure's default swap space is adequate. Unless you are installing large applications along with HP BASIC/UX, setting up a diskless cluster, or have several users using the system simultaneously, you should follow the system default when asked for swap space.

What is Swap Space?

When a process is run in HP-UX, it does not all have to fit in the computer's memory (RAM): it is divided into pieces called pages. A mechanism called the

swapper swaps the process pages between memory and disk as needed. This way several processes can run at the same time and share system resources.

This is how HP-UX is able to multi-task (run processes at the same time), and how it can run processes larger than memory available. The only restriction is the amount of **swap space** available.

During HP-UX installation, you are asked to calculate the amount of swap space needed. Unless you are adding many applications in addition to BASIC/UX to the HP-UX system, you should use the installation procedure's defaults. Information for calculating swap space is found in the *Series 300 HP-UX Installation* manual.

Never select a swap space value less than the default without careful consideration of the processes or applications you will be running.

Installing HP-UX

After completing the tasks in "Preparing to Install HP-UX," you are ready to begin installing HP-UX. *If your hard disk already has a previous version of HP-UX installed, read the next section, "Updating HP-UX," for information on updating to HP-UX version 8.0.*

Note

You must install HP-UX 8.0, or update to it, before you install HP BASIC/UX 6.2. HP BASIC/UX 6.2 will run only on HP-UX version 8.0, not on previous versions.

To install HP-UX, follow the procedures given in the *Series 300 HP-UX Installation* manual and return when HP-UX is installed. As part of the HP-UX installation, you may wish to install and configure the X11 Windows and HP VUE environments.

Updating HP-UX

If your hard disk has a previous version of HP-UX installed on it, you will have to update HP-UX to version 8.0 before you can install HP BASIC/UX 6.2. *If you are installing HP-UX for the first time, read the previous section, "Installing HP-UX."*

To check the version of HP-UX currently installed, log in to the existing system and execute:

```
uname -r (Return)
```

If you do not have the correct version (HP-UX 8.0), follow the instructions in the "Read Me Before Installing or Updating HP-UX" document. Complete instructions for updating are found in the chapter "Updating HP-UX" in the *Series 300 HP-UX System Administration Tasks* manual.

After completing the update process, read the file `/tmp/update.log` for information about the results of the update process.

What's the Next Step?

After you install or update HP-UX you are ready to install HP BASIC/UX. For information on how to do this, refer to chapter 3, "Installing BASIC/UX on to HP-UX."

Installing BASIC/UX on to HP-UX

Once you have installed HP-UX 8.0, or you have updated to version 8.0 from a previous version, you can install HP BASIC/UX 6.2. You'll use the `update` application in HP-UX to install BASIC/UX.

Note

If you purchased a "bundled" BASIC/UX system with pre-installed software (e.g. the HP 9000 Model 382 or Model R/382 with the BASIC/UX option), you can skip this chapter. HP-UX 8.0, HP BASIC/UX 6.2, X11 Windows, and HP VUE have been pre-installed and pre-configured on such systems. Skip to "Starting a BASIC/UX System With HP VUE" in chapter 4.

Checking Your HP-UX Version

HP BASIC/UX 6.2 runs only on version 8.0 of HP-UX, not on previous versions. If you are installing HP BASIC/UX on to an existing HP-UX system, check your current version of HP-UX. Log in, and type:

```
uname -r (Return)
```

If the current version is not HP-UX 8.0, go to the section "Updating HP-UX" in the previous chapter and update to version 8.0 *before* you install BASIC/UX.

3

Prerequisites for Installing or Updating Your System

The following file sets are required for installing or updating to HP BASIC/UX 6.2: C-MIN and KERN-BLD. These file sets are used for reconfiguring the kernel drivers, system parameters, or swap space. See the *Series 300 HP-UX System Administration Tasks* manual if you need help in verifying the presence of these file sets. Note that the minimal AXE system does not have these file sets.

HP BASIC/UX 6.2 will run only on version 8.0 of HP-UX. *It will not run on previous versions of HP-UX*. For details on running HP BASIC/UX with future releases of HP-UX, please contact your local HP Sales Representative.

BASIC/UX File Sets

When you update HP-UX to install HP BASIC/UX, you add file sets to the system. This section describes the file sets and the disk space they require. Note the sizes shown are approximate.

Three file sets comprise HP BASIC/UX:

BASIC/UX File Sets

File Set	Size	Description
RMBUX-PRG	3.9 MB	Mandatory file set containing core `rmb` files such as: `rmb` `rmbconfig` `rmbclean` `ipcclean` `rmbtmr` `rmbxfr` `rmbkbd` `rmbhil` font libraries, and `nls` files.
RMBUX-CSUB	869 KB	Optional file set containing CSUB Utility and files such as: `librmb.a` `csubdecl.h` `rmbbuildc`
RMBUX-UTIL	184 KB	Optional file set containing utilities such as: `BPLOT` `GDUMP_R` `LIF_UTIL`
RMBUX-MAN	18 KB	Man pages
RMBUX-DEMO	453 KB	Demos, and manual examples

Total size of combined file sets: 5.4 MB.

For a current list of file sets installed on your system, run this command:

```
ls /etc/filesets
```

Installing BASIC/UX with HP-UX Update Utility

To install HP BASIC/UX, you must follow the steps for updating HP-UX found in the *Series 300 HP-UX System Administration Tasks* manual.

Prerequisites

HP-UX must match the HP BASIC/UX version (see Chapter 1).

Follow the Update Procedure

The update procedure is found in the chapter "Updating HP-UX" in the *Series 300 HP-UX System Administration Tasks* manual. When you have completed the update, HP BASIC/UX is installed and ready to use.

Problems You Might Encounter

If you have trouble determining your source device, see the *Series 300 HP-UX System Administration Tasks* manual.

After Installing Your System

After completing the installation process, it may be necessary to build a new HP-UX kernel. Some commands are provided to help you do this. The command `/usr/lib/rmb/rmbdfile` will scan the `dfile` used to build your kernel and tell you if it is sufficient for `rmb`. It can also produce a modified `dfile` for use with the `/usr/lib/rmb/rmbkernel` command. This command builds a new HP-UX kernel, and provides instructions on how to install it. See the HP-UX manual pages in Appendix A for details on `rmbdfile` and `rmbkernel`.

Customizing HP VUE for BASIC/UX

If you are using the HP VUE user interface, you can customize it for BASIC/UX. You can create your own custom configuration by following the instructions in your HP VUE documentation. However, you don't have to do this — we've provided a custom HP VUE configuration for BASIC/UX on your BASIC/UX tape. This custom configuration provides a special "BASIC/UX" icon on the HP VUE Workspace Manager. You can use the icon in the following ways:

- *As a "push button"* — just move the mouse to the icon and click the left mouse button to start BASIC.
- *As a "drop zone"* — you can "drag" programs from the HP VUE File Manager and "drop" them on the icon to start BASIC and run the program.

The icon is shown in the "Starting a BASIC/UX System With HP VUE" section of chapter 4.

Note

If you have a pre-installed BASIC/UX system, the custom configuration with the BASIC/UX icon is already set up. You won't need to do anything but log in to get the custom HP VUE screen.

If you do not have a pre-installed BASIC/UX system, you can set up the custom HP VUE configuration by following these steps:

1. Execute the following commands:

```
cp /etc/newconfig/rmb/sys.vuewmrc $HOME/.vue/.vuewmrc

xrdb -m /etc/newconfig/rmb/sys.resources
```

2. From the Workspace Menu, restart `Vuewm`.

For further information, refer to the *HP Visual User Environment User's Guide*.

4

After Installing BASIC/UX

This chapter describes what you should do after you finish installing HP BASIC/UX.

Testing BASIC/UX

Once you have installed HP-UX and BASIC/UX, you can test your system. The procedure for starting BASIC/UX depends on whether you are using HP VUE (Visual User Environment) or not. (If you have a pre-installed HP BASIC/UX 6.2 system, it is configured to use HP VUE.)

Starting a BASIC/UX System With HP VUE

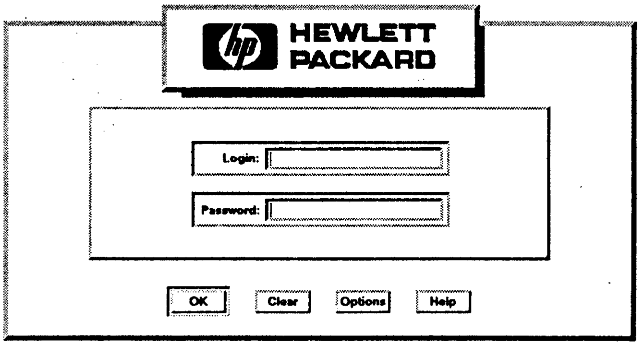
First, let's look at the procedure to test a pre-installed BASIC/UX system that you are turning on for the first time.

Pre-Installed BASIC/UX Systems

When you turn on your pre-installed BASIC/UX system (Model 382 or R/382) for the first time, it will prompt you to enter three things:

1. The **machine name**. Get your machine name from your system administrator (or refer to *HP-UX System Administration Tasks*). Type the name and press (Return).
2. The **internet address**. Get your internet address from your system administrator (or refer to *HP-UX System Administration Tasks*). Type the address and press (Return).
3. The **time zone**. Just follow the directions on the screen to select your time zone.

Once you have entered the requested data, the computer will reboot and the HP VUE login screen will appear:



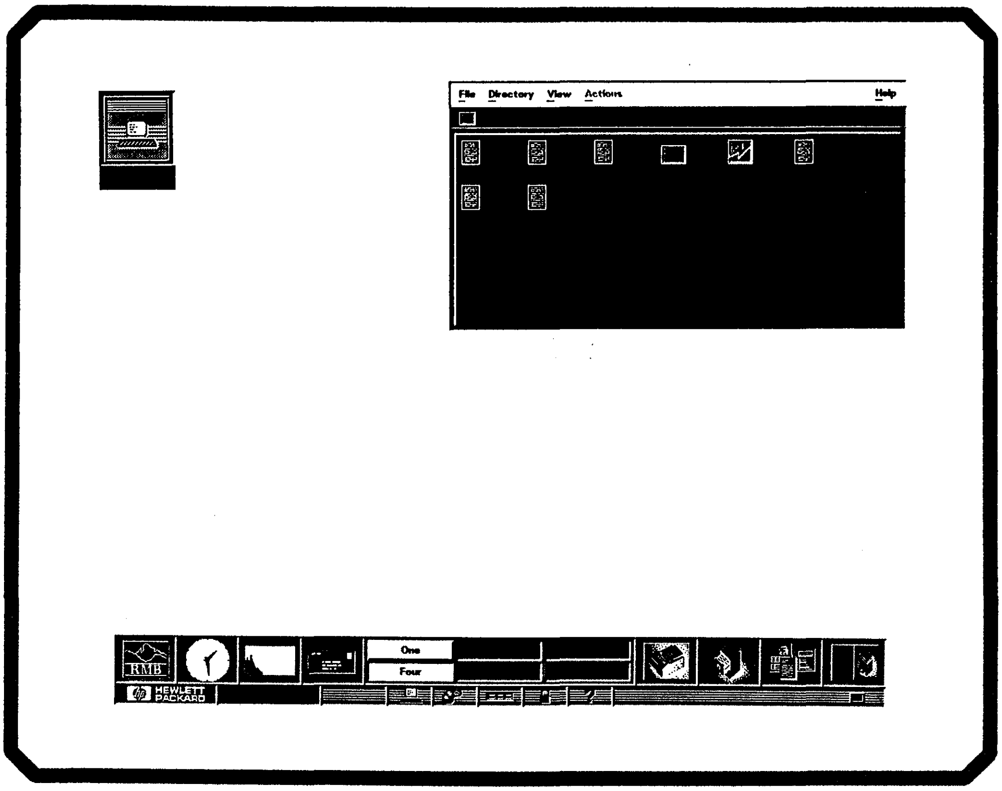
For now, you can log in as "root" and no password will be required.

Note To keep your system secure you should set a root password and establish your own login and password as soon as possible. See your system administrator or refer to the *HP Visual User Environment User's Guide*.

Type:

`root` (Return)

The following screen will appear:



To start BASIC/UX, just move the mouse cursor to the BASIC/UX icon at the left-hand side of the Workspace Manager and click the left mouse button. A BASIC/UX window will appear. Refer to *Using HP BASIC/UX 6.2* for further information.

User Installed BASIC/UX Systems With HP VUE

If you have installed HP-UX and BASIC/UX on your system, and you have configured your system to use the HP VUE user interface, the procedure for initial start up will be essentially the same as given in the previous section for a pre-installed system. However, the special icon for BASIC/UX will not appear unless you have set up HP VUE to use the custom configuration for BASIC/UX. Refer to "Customizing HP VUE for BASIC/UX" in chapter 3 for further information.

Starting a BASIC/UX System Without HP VUE

If you are using X11 Windows without HP VUE, you can start BASIC/UX manually from a window. If you are not using X11 Windows, just start BASIC/UX from the system prompt. The procedure is as follows:

Once HP-UX and BASIC/UX are installed, the system boots, and the system prompt ($) appears, type the following:

```
rmb (Return)
```

You should see the BASIC screen or window (as shown in *Using HP BASIC/UX 6.2*). If you do not, the update process did not work: try the procedure again (make sure you used the correct device file name for both the source and destination drives).

Before the systems are completely operational, you must complete some set-up tasks.

Follow the After Installing Instructions in HP-UX Installation

Go to the *Series 300 HP-UX Installation* manual and follow the instructions in the section "Tasks to Complete after Installation" in the chapter "Initial Configuration of HP-UX."

If You Backed-Up Files from a BASIC Workstation System: Restore Them

If you installed HP BASIC/UX on to a disk that was previously an HP BASIC Workstation system and backed-up files, follow these steps (from HP-UX):

1. Login as `root`.
2. Type:

   `cd /` (Return)
3. Check available space, by typing:

   `df` (Return) (1 block = 512 bytes).
4. Check space of files, by typing:

   `cpio -tc </dev/update.src` (Return)
5. If you have enough space, type:

   `cpio -ic </dev/update.src` (Return)

   If your drive has a different device file, substitute that device file name for `/dev/update.src`).

For More Information

See the *HP-UX Reference* for more on `cpio`. To see a list of device files:

```
ll -a /dev
```

Configuring a Diskless Node's Kernel

If you installed HP BASIC/UX on to a diskless cluster, a new kernel must be configured for each diskless node (cnode).

Prerequisites

Before You can configure a diskless node's kernel:

- You must be `root`.
- HP BASIC/UX must be installed on the cluster server.
- The system must be in the single-user state: `/etc/shutdown`.

Creating a Kernel Configuration File

Choose a dfile (a file with driver and configuration parameters), and run the configuration program `rmbdfile`:

1. Generate a list of possible dfiles: `ls /etc/conf/dfile*`
   - If you are at the server, you can choose `dfile.maxservr` (for a full HP-UX system) or `dfile.minservr` (for a minimal HP-UX system).
   - If at a node, you can choose `dfile.cnode`, `dfile.cnodemax`, or `dfile.cnodemin`.
2. Execute `rmbdfile` to determine if the `dfile` is adequate (this example uses `dfile.cnode`, substitute if you are using another dfile):

   ```
   /usr/lib/rmb/rmbdfile -v dfile.cnode
   ```

   If you receive the message:

   ```
   dfile configuration insufficient for rmb
   ```

   create a new `dfile` (this example uses `dfile.cnode` again):

   a. `mv dfile.cnode dfile.cnode.old`
   b. `/usr/lib/rmb/rmbdfile dfile.cnode.old > dfile.cnode`

Creating a New Kernel

To create a new kernel:

1. Run: `etc/config dfile.cnode`
2. Compile the resulting `config.mk` file: `make -f config.mk`
3. Move the new kernel into place, if *not at the server node*:

   ```
   mv ./hp-ux /hp-ux
   ```

   If at the server node (replace *cnodename* with the name of your cnode):

   `mv ./hp-ux /hp-ux+/`*cnodename*
4. Reboot the diskless node using the new kernel:

   ```
   /etc/reboot
   ```

For More Information

See Appendix A for the `rmbdfile` reference page. See also the chapter "Reconfiguring the Kernel Customizing the Kernel" in the *Series 300 HP-UX System Administration Tasks* manual.

5

Maintaining the BASIC/UX System

After HP-UX and HP BASIC/UX are installed, you are ready to use the system. As system administrator, you are responsible for:

- HP-UX system administration
- HP BASIC/UX system administration.

This chapter gives you an outline of the HP-UX system administration responsibilities and forwards you to the *Series 300 HP-UX System Administration Tasks* manual for details. HP BASIC/UX system administration is discussed in this book.

Some Differences Between BASIC/UX and BASIC Workstation

Two chapters in *HP BASIC 6.2 Porting and Globalization* discuss the differences between HP BASIC/UX and HP BASIC/WS ("BASIC/UX Differences and Enhancements" and "Porting BASIC/WS Programs to BASIC/UX"). Here are a few additional items:

- All system binaries for HP BASIC/UX are permanently loaded. You can not load binaries, nor can you SCRATCH BIN.
- System time for HP BASIC/UX is calculated as an offset of HP-UX system time. This means a SET TIMEDATE changes HP BASIC/UX time but not HP-UX time. You must be super-user (logged in as `root`) to change the HP-UX system time (for example, `date 0311120088`).
- HP BASIC/UX can take advantage of the HP-UX native language support for error messages.

Living Within the HP-UX Structure

Since HP BASIC/UX runs "on-top-of" HP-UX, you as system administrator must deal with certain aspects of HP-UX. The files that HP BASIC/UX uses are located in various directories within the HP-UX file structure.

Prerequisites

Have a clear understanding of files and directories (if not, see *Using HP BASIC/UX 6.2*.

Important BASIC/UX Files and Directories

The following table lists some important files and directories that make up the BASIC/UX system.

Short Descriptions of Some BASIC/UX Files and Directories

File Name/Directory Name	Description
`/usr/bin`	Directory containing executable `rmb` command and CSUB build utility.
`/usr/bin/rmb`	Core BASIC/UX executable file.
`/usr/bin/rmbbuildc`	CSUB creation utility.
`/usr/lib/rmb`	Directory containing all BASIC/UX utilities, global files, and CSUBs.
`/usr/lib/rmb/rmbconfig`	Gathers configuration data needed by `rmb` and creates device files.
`/usr/lib/rmb/rmbclean`	Cleans up shared memory.
`/usr/lib/rmb/ipcclean`	Cleans up lock files (used by `rmbclean`).
`/usr/lib/rmb/utils`	Directory containing CSUBs and utilities provided with BASIC/UX.
`/usr/lib/librmb.a`	Export symbol definitions for CSUBs.
`/dev/rmb`	Device directory for special file entries.
`/etc/newconfig/rmb/rmbrc`	Default global environment file.
`/etc/newconfig/rmb`	Directory which contains default template files (used when you customize): `rmbrc`, `d.x11start`, `d.Xdefaults`.

For More Information

For information on utilities provided by HP BASIC/UX, see chapter 6, "BASIC/UX Utilities."

HP-UX System Administration Tasks

The *Series 300 HP-UX System Administration Tasks* manual discusses in detail the tasks you need to perform for your HP-UX system. Start by reading the "Introduction to System Administration" chapter. This chapter outlines the tasks you need to perform. In addition, a checklist below is provided for your convenience.

HP-UX administration can be complicated at times. You should have a good understanding of HP-UX before attempting many of the tasks. See *A Beginner's Guide to HP-UX* for beginning HP-UX concepts.

The following table lists some system administration tasks and tells where to find information for each task in the *Series 300 HP-UX System Administration Tasks* manual.

HP-UX System Administration Tasks

Task	Where to Find Information *(HP-UX System Administration Tasks)*
Evaluating user's needs	"Evaluating User Needs and Configuring a System" in the "Constructing an HP-UX System" chapter
Configuring (optimizing) HP-UX	"Reconfiguring the Kernel Customizing the Kernel" chapter.
Adding Users to the system	"Adding Users" in the "Managing Users" chapter.
Adding and moving peripheral devices	"Managing Devices" chapter.
Monitoring the file system	"Managing the File System" chapter.
Updating HP-UX	"Updating HP-UX" chapter.
Backing-up, recovering, and restoring the system	"Backing Up and Restoring the System" chapter.
Detecting/correcting file system errors	"Managing the File System" chapter.

BASIC/UX System Administration Tasks

As system administrator, you are mostly concerned with HP-UX system administration. Other than tracking user errors (problems the users have with HP BASIC/UX), your HP BASIC/UX responsibilities are

- to customize the system.
- set up peripherals.
- back-up the file system periodically.
- run utilities `rmbconfig` and `rmbclean`.

Running the Configuration Utility: rmbconfig

`rmbconfig` (found in `/usr/lib/rmb`) is a utility that:

- gathers information about the kernel and system hardware, from HP-UX, needed for HP BASIC/UX to run.
- places the information in `/usr/lib/rmb/rmbbootinfo` (used by `rmb`).
- creates device files for installed interface cards in `/dev/rmb`. `rmbconfig` creates links to existing device files where possible; otherwise, it creates new device files and assigns permissions 000 to the device files. You need to change the permissions, taking into account any local security issues, before users may access instruments or peripherals on the interfaces. For example, to give all users access to the internal HP-IB interface at select code 7 execute:

  ```
  chmod 666 /dev/rmb/hpib7
  ```

`rmbconfig` runs *each* time you boot HP-UX (it has been added to the `/etc/rc` file). If you wish to run it other than at boot time, add the `-k` option to the `rmb` command (`rmb -k`). This runs `rmbconfig` when a HP BASIC/UX session is started.

Cleaning Up System Resources with rmbclean

Whenever HP BASIC/UX terminates abnormally (power is lost or HP BASIC/UX is killed by a user), system resources and lock files need to be cleaned up. `rmbclean` is a utility to clean up the system and insure smooth HP BASIC/UX operation.

To execute `rmbclean` type (in HP-UX):

```
/usr/lib/rmb/rmbclean Return
```

or use the -i option in the `rmb` command: `rmb -i`

If a user who is not `root` runs `rmbclean`, only the unused lock files of that user are cleaned up. HP BASIC/UX runs `rmbclean` automatically if the system runs out of resources.

5

Customizing Your BASIC/UX Session

When you start a BASIX/UX session, the system looks for a file

```
/usr/lib/rmb/rmbrc
```

to set the default environment (the environment consists of system variables that affect how the system performs some tasks).

A template environment file:

```
/etc/newconfig/rmb/rmbrc
```

is available for you to customize and move to the `/usr/lib/rmb` directory. If you change this file, *you change the environment for all users.* To change the environment for a particular user, see *Using HP BASIC/UX 6.2*.

What Variables Can Be In The Environment File?

More details about these variables follow the table.

Global Environment Variables

Name of Variable	Range of Values	Default Values	Description
`autostart`	*pathname*	n/a	Pathname of an autostart file.
`errormode`	`on`, `off`	`on`	Generate error messages for BASIC Workstation, BASIC/UX compatibility.
`graphics_buffer`	`on`, `off`	`on`	Turn on graphics buffering to speed up graphics processing time.
`heap_prealloc`	0 to space available	n/a	Preallocate heap space.
`hfs_buffer`	`on`, `off`	`on`	Turn on HFS file system buffering.
`plock`	`all`, `t`, `d`, `w`	n/a	Lock text area, data area, workspace or all of BASIC/UX into memory (any combination of `t`, `d`, and `w` is valid).
`term_control`	`on`, `off`	off	Provide access to the special terminal keyboard mappings.
`workspace`	64 KB to shmmax	1 MB	Size of BASIC/UX workspace (integer values only).

Here is more detail on the environment variables for HP BASIC/UX, as well as some additional statements you can add to `rmbrc`.

Running an Autostart Program (autostart)

If you wish an autostart program, you can specify the file with **autostart**. For example, using the program, **/users/leslie/AUTOST**, you would enter:

```
   50 !autostart=/users/leslie/AUTOST
```

into the **.rmbrc** file. The line number is arbitrary, and the example shows a file created using EDIT mode.

Generate Compatibility Error Messages (errormode)

If you port programs created on HP BASIC Workstation systems, you may have some errors with, for example, commands not supported on HP BASIC/UX (such as LOAD BIN).

`60 !errormode=on`	has error messages generated.
`60 !errormode=off`	does not print error messages.

The above examples show an arbitrary line number created using EDIT mode.

Graphics Buffering (graphics_buffer)

When using graphics, you can choose to have graphics buffering:

`70 !graphics_buffer=on`	turns on graphics buffering. The image is faster than when off, but it could be choppy when an image moves on the screen.
`70 !graphics_buffer=off`	is slower than when on, but the image is smoother when moving on the screen.

The above examples show an arbitrary line number created using EDIT mode.

Increasing the Heap Space (heap_prealloc)

Increase your heap space, for example:

```
   75 !heap_prealloc= additional_heap_space
```

The above example shows an arbitrary line number created using EDIT mode. If the *additional_heap_space* given in bytes is zero (the default value), then no additional heap space is allocated; however, if it is non-zero then the amount of heap space specified is preallocated.

Some Heap-consuming BASIC/UX Operations

The heap-consuming BASIC operations are listed below, as well as suggested amounts of heap space to add for each one if the need arises:

Heap-consuming BASIC/UX Operations

Operation	Heap Space Required
CREATE WINDOW	17K for the default window and buffer sizes
GLOAD/GSTORE	*width* × *height* of "from" device (if given) or "PLOTTER IS" device (if no parameters)
INITIALIZE *memory volume*	*number of sectors* × 256 bytes
CSUBS	size of stored CSUB file
BPLOT	*width* × *height* for given parameters (plus size of stored CSUB)
GDUMP_R	*width* × *height* of "from" device (plus size of stored CSUB)
DUMP GRAPHICS	*width* × *height* of "from" device or `PLOTTER IS` device if no parameter is given
mass memory	operations on HFS directories will at most use 20 KB
opening SRM file	each open file uses 48 bytes

HFS File System Buffering (hfs_buffer)

This variable determines how data is written to a disk:

`80 !hfs_buffer=on`	saves data in a buffer and writes to a disk periodically. This makes system operations faster, but could cause a greater amount of data to be lost if a power failure occurs, or if the system is improperly shut down.
`80 !hfs_buffer=off`	writes the data to the buffer, then immediately to the disk. This causes the performance of OUTPUT to be much slower.

The above examples show an arbitrary line number created using EDIT mode.

Locking BASIC/UX in Memory (plock)

To lock the text area, data area, workspace, or all of HP BASIC/UX into memory (disables swapping),

`90 !plock=all`	locks BASIC/UX into memory.
`90 !plock=t`	locks text area into memory.
`90 !plock=d`	locks data area into memory.
`90 !plock=w`	locks workspace into memory.

Any combination of `t`, `d`, or `w` is valid (for example, `plock=td`). The above examples show an arbitrary line number created using EDIT mode.

Setting Special Terminal Keyboard Mappings (term_control)

Set the terminal keyboard mapping mode, for example:

`115! term_control=on`	gives you access to the special terminal keyboard mappings, such as CTRL-R (Reset) and CTRL-L (Recall) that are documented in *Using HP BASIC/UX 6.2.*
`115! term_control=off`	turns off the special terminal keyboard mappings.

The above examples show an arbitrary line number created using EDIT mode.

Setting Size of BASIC/UX Workspace (workspace)

Set the size of HP BASIC/UX workspace, for example:

```
   120 !workspace=2m
```

The above example show an arbitrary line number created using EDIT mode. Your value should be dependent on the size of programs to be run. For example, a program with lots of subprograms, CSUBs, etc., needs more workspace than a small program.

Customizing the Kernel to Increase Workspace Size

If you need to set your workspace larger than six megabytes, you may encounter problems when trying to boot BASIC/UX. For example, you may see a message such as:

```
   rmb: shared memory size exceeds kernel limits
```

This is because the BASIC/UX workspace is exceeding its boundaries. The information covered in this section provides you with an explanation and solution to this problem.

BASIC/UX places its workspace in a shared memory segment. HP-UX, however, imposes some limits when dealing with shared memory segments. The kernel parameter **shmmax** defines the maximum size of any shared memory segment in an HP-UX system. The BASIC/UX workspace cannot be larger than this parameter. The default size of **shmmax** is six megabytes, and any changes must be made by reconfiguring the kernel.

The BASIC/UX Process Memory Map

The following diagram shows the memory layout of the BASIC/UX process:

```
           =============================  <--  Maximum process size is four
           |   HP-UX reserved space    |       gigabytes (0xffffffff)
           -----------------------------
           |       Process stack       |
           |             v             |
           -----------------------------
           |                           |
           |            ...            |
           |                           |
workspace  =============================  <--  TOP of BASIC/UX shmem/workspace
workspace  |      BASIC/UX Stack       |       (default workspace = 1 MB)
workspace  |             v             |
workspace  -----------------------------
workspace  |             ^             |
workspace  |   BASIC/UX growth area    |
workspace  -----------------------------
workspace  |   BASIC/UX static area    |
workspace  =============================  <--  BOTTOM, BASIC/UX shmem/workspace
           |            ...            |       (0x80000000)
           |                           |
           =============================  <--  TOP of Starbase memory
           |   Starbase Frame Buffer   |       (always at or below 0x80000000)
           |      (1.16-12.5 MB)       |
 starbase  -----------------------------  <--  TOP of Starbase shared memory
 starbase  |                           |       (automatically chosen by kernel)
 starbase  |                           |
 starbase  |   Grahics Shared Memory   |   *   The size of the GRM-managed area
 starbase  |      managed by GRM       |   *   of memory is determined by the
 starbase  | (default 2.0 MB, based on |   *   WMSHMSPC environment variable.
 starbase  | WMSHMSPC environment var) |   *
 starbase  |                           |
 starbase  =============================  <--  BOTTOM of Starbase shared memory
           |                           |       (always at or above 0x40000000)
           |            ...            |
           |                           |
           -----------------------------  <--  TOP of heap
           |             ^             |
           | Heap Space (8 KB + CSUBs)  |
code/data  =============================  <--  BOTTOM of heap (0x0039d000)
code/data  |     BASIC/UX bss data     |
code/data  -----------------------------
code/data  |       BASIC/UX data       |
code/data  -----------------------------  <--  0x00345000
code/data  |       BASIC/UX text       |
code/data  =============================  <--  0x00000000
```

Here are some notes on the BASIC/UX process memory map:

- The area marked `workspace` is the shared memory segment in which the BASIC/UX workspace is stored. The area marked `starbase` is the shared memory segment used by Starbase. The area marked `code/data` is portion of the process space in which BASIC/UX code and data reside.
- The `Starbase Frame Buffer` portion of the memory map corresponds to the control registers and screen of your display. This area may range from 1.16 MB to 12.5 MB or more. The maximum size of the frame buffer for a DIO-I display is just over 1 MB. The maximum size of the frame buffer for a DIO-II display is just over 12 MB. For more information on DIO-I and DIO-II displays, refer to your display's section in the *Starbase Device Drivers Library* manual.
- The `Graphics Shared Memory` portion of the memory map stores information pertinent to X11 Windows and Starbase shared use of the display (for example, windows with retained raster images, cursor information, fonts, etc.). This area of memory is managed by the Starbase Graphics Resource Manager or GRM daemon. The size of this area is determined by the WMSHMSPC environment variable. If the WMSHMSPC variable is not explicitly set, it will default to 2 MB, which is sufficient for most situations.
- It is very important to note, however, that the WMSHMSPC variable only determines the `Graphics Shared Memory` size in the environment in which the GRM is first started. When running X11 Windows, the GRM is started at the same time the X11 server is started, and runs all the time that the X11 server runs. In this case, the value of WMSHMSPC in the startup environment of the X11 server will be used in allocating shared memory for the GRM.
- When running BASIC/UX outside of X11 Windows (in console mode), the GRM is started at the same time BASIC/UX is started, and runs all the time that BASIC/UX runs. In this case, the value of WMSHMSPC in the startup environment of BASIC/UX will be used in allocating shared memory for the GRM.
- If you define or change WMSHMSPC before running the X11 server, you should ensure that all other graphics applications (including BASIC/UX) that run under the X11 server are given the same value of WMSHMSPC so

that they can correctly compute the virtual address at which to access the shared GRM memory.

- For more information about the interaction of Starbase, X11, and shared memory, see the topic "Shared Memory Usage" in the "Using Starbase in X11 Windows" chapter in the *Starbase Device Drivers Library* manual. You should most certainly read this section if you feel you need to change the WMSHMSPC environment variable.

Using SAM to Enlarge the Kernel's `shmmax` Parameter

This section assumes that you have already installed BASIC/UX and have modified your kernel (if necessary) to work with BASIC/UX. (See the section "After Installing/Updating Your System" in chapter 4 of this manual. You may also need to refer to the HP-UX manual pages in appendix A for details on using the utilities `rmbdfile` and `rmbkernel` found in `/usr/lib/rmb`.)

Before you execute the procedure that follows, please ensure that you have configured your kernel as necessary for BASIC/UX, and that you can run BASIC/UX with the default workspace size of one megabyte.

The procedure is as follows:

1. Log in as the super-user (`root`). If you are using a diskless cluster, log in on the node for which you are modifying the kernel.
2. Ask everyone to log out (give them enough time), bring the system down to single user mode, and run the System Administration Manager (SAM):

   ```
   % /etc/shutdown 5
   % sam
   ```

3. Choose `Kernel Configuration` from the main menu. In the dialog box that appears, type the name of your current kernel, or accept the default provided (usually `/hp-ux`).
4. Choose `Modify Operating System Parameters` from the Kernel Configuration menu.
5. Choose `Shared Memory Related Parameters ...` from the Modify Operating System Parameters menu.
6. Tab to the field labeled `Maximum number of bytes in a shared memory segment (shmmax)` and type the maximum number of bytes you will EVER

wish your BASIC/UX workspace size to be. The limit is 2147483647 bytes (or two gigabytes). SAM will not allow you to enter a value larger than two gigabytes.

7. Press the softkey labeled `Perform Task` (f4). You will see a dialog box which says `Task completed.`
8. Press the space bar to continue.

   Press the softkey labeled `Exit Task` (f8).
9. You will be back at the `Modify Operating System Parameters` menu. Press `Previous Menu` (softkey f8) to exit.
10. You will be back at the `Kernel Configuration` menu. Press `Previous Menu` (softkey f8) again to exit.
11. You will see a dialog box telling you that `Kernel configuration information has been modified.` You must decide at this point whether you wish to rebuild the kernel now, or exit SAM without making any changes. To rebuild the kernel now, type an `x` next to the choice `Regenerate the kernel with modified values.` and press the `Done` softkey (f4).
12. You will see a dialog box which says ... `This will take a few minutes.` while SAM works to rebuild the kernel. You will then see a dialog box which says `Done.` and another one which says `Kernel successfully rebuilt.` A final dialog box appears which says `Would you like to reboot NOW? (y or n).` Simply strike the `y` key.
13. You will see a dialog box which says ... `Moving new kernel to /hp-ux` followed by another dialog box which says `REBOOTING!!!.` Your machine will be automatically rebooted with the new kernel at this point.

Notes On Very Large BASIC/UX Workspaces

As discussed above, BASIC/UX places its workspace in a shared memory segment. The size of this shared memory segment is determined by the command line workspace argument (`-w`), or by the `WORKSPACE` parameter in your `.rmbrc` file. In HP-UX 8.0, the maximum size that *any* shared memory segment can be is two gigabytes (provided that you have configured your kernel by enlarging the kernel parameter `shmmax` from its default of six megabytes).

Note that while a two gigabyte BASIC/UX workspace sounds exciting, the total BASIC/UX process size (including code, data, heap, Starbase shared memory, BASIC/UX shared memory (WORKSPACE), and process stack) is in practice limited by the amount of swap space available on your system. Thus, very large BASIC/UX workspace sizes are essentially limited by available swap space.

The total size of all running processes must not exceed the available swap space. If you are running VUE, X11, and BASIC/UX, you might expect the maximum BASIC/UX workspace size to be roughly (total swap space) - 20 MB. On a system with 40 MB of swap space, you might expect to run a BASIC/UX process with a 20 MB workspace.

A Note On the RMB_SHMEM_ADDR Environment Variable (Now Obsolete)

In previous revisions of BASIC/UX, the environment variable `RMB_SHMEM_ADDR` was used to reserve space between the top of the BASIC/UX workspace and the maximum shared memory address. You might have used this variable to reserve space for CSUBs that accessed their own shared memory segments in that space. `RMB_SHMEM_ADDR` *is no longer supported* by BASIC/UX and will be *ignored* if you set it. With the new virtual memory system used in HP-UX 8.0, it is no longer necessary to reserve space for shared memory in this manner. There is ample space in the process memory map to create shared memory segments above the BASIC/UX workspace.

A Note On the SB_DISPLAY_ADDR Environment Variable

In previous revisions of BASIC/UX, you were instructed to change the Starbase environment variable `SB_DISPLAY_ADDR` in order to use very large BASIC/UX workspaces. This is no longer necessary since the shared memory segment is no longer bounded on the top by the (now obsolete) kernel parameter **shmmaxaddr**. Instead, the BASIC/UX shared memory segment used for the workspace simply begins above the Starbase frame buffers, and grows toward the top of the process space. The size of the workspace shared memory segment is limited *only* by the current setting of the kernel parameter **shmmax**, which can easily be changed as directed above.

Using Retained Windows With BASIC/UX

When you use the BASIC/UX `CREATE WINDOW` statement, you can specify that the window about to be created should be a *retained* graphics window. This means that any graphics you draw in that window will be automatically stored on your behalf by the X11 server. Thus, if you iconify or obscure that window, any graphics present will automatically be redrawn by the X11 server when you later de-iconify or un-obscure the window. Non-retained windows do not exhibit this feature. Any portion of a graphics image that is obscured will be lost when the non-retained window containing the graphics image is later un-obscured. The entire graphics image will be lost when the non-retained window containing that image is iconified and later de-iconified.

If you need to create a large number of *retained* graphics windows using the BASIC/UX `CREATE WINDOW` command, you may encounter problems with some of the windows not properly retaining their images. When you iconify or obscure one of these windows and later de-iconify or un-obscure it, the graphics image is not redisplayed by the X11 server. This is caused by the Starbase Graphics Resource Manager (GRM) daemon overrunning its shared memory segment. To alleviate this problem, you need to increase the size of the shared memory segment allocated to the GRM. During this discussion you may wish to refer to the BASIC/UX Process Memory Map shown on page 5-12 in the preceding section, "Customizing the Kernel to Increase Workspace Size."

On the BASIC/UX Process Memory Map, you will notice a section labeled `Graphics Shared Memory managed by GRM`. This section of memory stores information pertinent to X11 windows and Starbase shared use of the display (for retained windows, cursor information, fonts, etc.), and defaults to two megabytes. Creating a large number of retained windows, or creating several very large retained windows can "fill up" this section of memory.

The Starbase environment variable `WMSHMSPC` controls the size of the GRM shared memory area. You can enlarge the GRM shared memory area to accomodate more retained windows by setting the `WMSHMSPC` variable.

For example, if you wished to increase the size of `WMSHMSPC` from the default, two megabytes, to six megabytes, you would need to set `WMSHMSPC` to `0x600000`.

As noted in "Customizing the Kernel to Increase Workspace Size", the `WMSHMSPC` variable *must* be set in the environment in which the X11 server and

GRM are first started. The sections below discuss the appropriate place to set this variable.

Setting the WMSHMSPC Environment Variable in HP VUE

Most environment variables can be set in your `.profile` or `.login` file. Since `WMSHMSPC` must be set *before* the X11 server and GRM are started, however, you cannot set this variable in `.profile` or `.login`. When using HP VUE, the X11 server is *already running* by the time `.profile` or `.login` are encountered.

You must instead use the `Vuelogin` resource `environment`. This resource is added to the file `/usr/lib/X11/vue/Vuelogin/Xconfig`. For example, to increase `WMSHMSPC` to six megabytes, you would add the following line to `/usr/lib/X11/vue/Vuelogin/Xconfig`:

```
Vuelogin*environment:          WMSHMSPC=0x600000
```

Refer to the *HP VUE System Administration* manual for more information.

Setting WMSHMSPC Using Bourne or K Shell (Not using HP VUE)

If you are not using HP VUE and you use the Bourne shell or K shell, you can simply set `WMSHMSPC` by modifying `.profile` in your home directory. For example, to increase `WMSHMSPC` to six megabytes, you would add the following lines to `.profile`:

```
WMSHMSPC=0x600000
export WMSHMSPC
```

Setting WMSHMSPC Using C Shell (Not using HP VUE)

If you are not using HP VUE and you use the C shell, you can simply set `WMSHMSPC` by modifying the `.login` file in your home directory. For example, to increase `WMSHMSPC` to six megabytes, you would add the following line to `.login`:

```
setenv WMSHMSPC 0x600000
```

Information for SRX, Turbo SRX, Personal VRX, and Turbo VRX Displays

If you have one of the following four displays:

Personal VRX (PVRX):	HP 98704/05
SRX:	HP 98720/21
Turbo SRX (TSRX):	HP 98730/31
Turbo VRX (TVRX):	HP 98735/36

and you wish to increase the size of WMSHMSPC, you must still pay attention to the Starbase environment variable SB_DISPLAY_ADDR. SB_DISPLAY_ADDR tells Starbase where in the process memory map to place the *bottom* of the display frame buffer. The *top* of the GRM shared memory segment is placed at this same address.

You should only worry about SB_DISPLAY_ADDR if both of the following are true:

- you need to increase WMSHMSPC, and
- you are using one of the four displays listed above.

In all other cases, it is not necessary to set SB_DISPLAY_ADDR. In fact, for displays other than the four listed above, SB_DISPLAY_ADDR will explictly be *ignored*—the kernel automatically picks an appropriate address in the one gigabyte to two gigabyte range (0x40000000 to 0x80000000) at which to map the Starbase frame buffer and GRM shared memory segment.

For these four displays, the GRM shared memory space exists just below the location indicated by SB_DISPLAY_ADDR (which defaults to 11 megabytes, or 0xb00000 if not set). When you increase the size of WMSHMSPC, you should plan to raise SB_DISPLAY_ADDR by the same amount you increase WMSHMSPC.

For example, to increase WMSHMSPC from the default, two megabytes, to six megabytes, you would need to set WMSHMSPC to 0x600000. Since you have increased WMSHMSPC by four megabytes, you should also raise the value of SB_DISPLAY_ADDR by four megabytes from 11 megabytes to 15 megabytes (0xf00000).

In HP VUE, you would add the following line to **/usr/lib/X11/vue/Vuelogin/Xconfig**:

```
Vuelogin*environment:        WMSHMSPC=0x600000 \
                             SB_DISPLAY_ADDR=0xf00000
```

If you are not using HP VUE and you use the Bourne or K Shell, you would add the following lines to `.profile` in your home directory:

```
WMSHMSPC=0x600000
SB_DISPLAY_ADDR=0xf00000
export WMSHMSPC SB_DISPLAY_ADDR
```

If you are not using HP VUE and you use the C Shell, you would add the following lines to `.login` in your home directory:

```
setenv WMSHMSPC 0x600000
setenv SB_DISPLAY_ADDR 0xf00000
```

Setting Up Automatic Device File Locking and Mapping

Set up automatic locking, memory mapping, or set `io_burst` on an I/O interface. Note that `autolock` + `automap` is equal to `autoburst`.

5

Determine the select code of the interface, and include this in the `.rmbrc` file:

`interface` *select_code*; *option*

where *option* is one of the following:

- `autolock`
- `automap`
- `autoburst`
- `normal`.

Mapping BASIC Mass Storage Volume Specifiers to HFS Directories

If you have programs that access mass storage devices with the volume specifiers, you can map a volume specifier to an HFS directory with this entry:

`disk` *scba,volume,unit* = *directory name*

where

- *scba* is: select code * 100 + bus address
- *,volume* is the volume number (optional; use the comma if you include this)
- *,unit* is the unit number (optional; use the comma if you include this)
- *directory name* is an HFS directory. This could be the directory under which the disk is mounted.

For example, map a device on select code 7, bus address 2, volume 1 mounted under `/disk1`.

```
disk 702,1 = /disk1
```

How to Create Your Environment File

To make the `rmbrc` file compatible with HP-UX, you must create a new `rmbrc` file and place it in `/usr/lib/rmb`. There are two ways to format the file:

- Using an HP-UX editor (`vi`, for example).
- Using the HP BASIC/UX editor.

Using HP-UX Editor (HP-UX EDIT mode)

1. Login as `root` or become super-user (`su`).
2. Change directories: `cd /etc/newconfig/rmb` (Return)
3. Make a copy of the default file: `cp rmbrc rmbrc.dflt` (Return).
4. Modify the `rmbrc` file, for example: `vi rmbrc` (Return)
5. When finished modifying (see below for an example), move the customized `rmbrc` file to the correct directory:

   ```
   mv rmbrc /usr/lib/rmb/rmbrc   (Return)
   ```

Precede comments with the # character; *include statements without spaces between variables and values*. For example:

```
# This is a sample rmbrc file edited in HP-UX
interface 7;autolock
errormode=off
workspace=1m
# End of sample rmbrc file
```

Files created in this manner (without line numbers or "!" before entries) cannot be modified using the BASIC Editor.

Using the BASIC Editor (BASIC EDIT mode)

1. Login as `root` or become super-user (`su`).
2. Enter BASIC/UX: `rmb` (Return).
3. Change directories: `MSI "/etc/newconfig/rmb"` (Return)
4. Edit a new file: `EDIT` (Return)
5. Create your customized environment file (see below for an example).
6. SAVE to the `rmb` directory: `SAVE "/usr/lib/rmb/rmbrc"` (Return)

All lines must be preceded with line numbers and an exclamation mark. Precede comments with `!#` or `REM`. For example:

```
10 REM This is a sample rmbrc file edited in BASIC
20 !interface 7;autolock
30 !errormode=off
40 !workspace=1m
50 !# End of sample rmbrc file
```

How to Set the Time, Date, and Time Zone

HP-UX time is the base time for all processes. HP BASIC/UX is an offset (or local) time relative to the HP-UX time. Thus, BASIC users can change time within HP BASIC/UX without affecting the time of other processes in the system.

Prerequisites

You must be logged in as `root` or super-user (`su`) to change HP-UX time.

How to Change HP-UX Time, Date, and Time Zone

You can be either in HP-UX or HP BASIC/UX to perform these tasks. If you are in HP BASIC/UX, use the EXECUTE command with the following HP-UX command (for example, `EXECUTE "date 0311120088"`).

HP-UX Commands for Time, Date, and Time Zone Changes

Command	Description	Example
`date`	Changes time and date. The example changes the date to March 11 at 12:00 noon, 1988.	`date 0311120088`

How to Change the BASIC/UX Local Time

Each user can change the local HP BASIC/UX offset with these HP BASIC/UX commands:

BASIC/UX Commands for Time, Date, and Time Zone Changes

Command	Description	Example
`SET TIME`	Changes time. The example changes the time to 12:00 noon.	`SET TIME TIME("12:00:00")`
`SET TIMEDATE`	Changes time and date. The example changes the time to March 11 at 12:00 noon.	`SET TIMEDATE TIME("12:00:00") + DATE("11 Mar 1988")`
`TIMEZONE IS`	Changes timezone. The example changes the timezone to Mountain Standard.	`TIMEZONE IS -7*3600`

For More Information

- See the *HP BASIC 6.2 Language Reference* for more on SET TIME, SET TIMEDATE, and TIMEZONE IS.
- Read "The Real Time Clock" in the *HP BASIC/UX Programming Guide* for more on setting time.
- See the entry for `date` in the *HP-UX Reference* for changing HP-UX time.

Setting Up HFS File Systems

An HFS (Hierarchical File System) disk must be mounted on to the HP-UX file system, making it part of the file system.

Prerequisites

Determine what directory/path name you want as your mount point (the directory used to access the HFS disk). For example, **`/disk1`**.

Connecting the Disk

1. Go to the *Series 300 HP-UX Installing Peripherals* manual, "Adding Mass Storage Devices" chapter.
2. Read the introductory information for your disk. When you see "Turn your computer off," *you must shut down the system first, with* **`/etc/shutdown -h`**.
3. Follow the installation procedure to set the address.
4. Write down the select code, bus address, volume number, and unit number you used.

Collecting Vital Information

Before you can mount the disk, you must turn to "Worksheet Entries" in the same section of the *Series 300 HP-UX Installing Peripherals* manual for your HFS disk.

1. Write down these values from the table for "File Type" b (follow the instructions in the foot notes to replace any italicized letters in the values):

   a. Path Name

   b. Minor Number.

Mounting the HFS Disk

1. Login as root (or become super-user with su).
2. Check that there are no devices with the same path name as you recorded in "Collecting Vital Information":

   ```
   ll -a /dev Return
   ```

   If there is, make a new path name (for example, if /dev/dsk/1s0 already exists, use /dev/dsk/2s0).
3. Run the following command, substituting the values you collected in "Collecting Vital Information" for the words in italics below (note that a *major_number* of 0 represents a CS80 block device and a *major_number* of 7 represents a SCSI block device):

   /etc/mknod *path_name* b *major_number* *minor_number*

   For example, for an HP 7959 Disk Drive connected to a built-in HP-IB Interface at bus address 2:

   ```
   /etc/mknod /dev/dsk/1s0 b 0 0x070200
   ```

   For the Model 362/382 built-in SCSI hard disk at select code 14, address 6:

   ```
   /etc/mknod /dev/dsk/1s0 b 7 0x0e0600
   ```

4. Mount the disk to the mount point you selected in "Prerequisites":

```
/etc/mount path_name mount_point
```

For example, mount the above disk to **/disk1**:

```
/etc/mount /dev/dsk/1s0 /disk1
```

Accessing and Unmounting the Disk

To access the mounted HFS disk, simply change directory to the mount point path name. For example, in HP BASIC/UX:

```
MSI "/disk1"
```

To remove the disk from the system, run the **umount** command and supply the mount path name. For example,

```
/etc/umount /disk1
```

Problems You Might Encounter

If you have a disk that's not listed in the *Series 300 HP-UX Installing Peripherals* manual, you can calculate the Minor number by following the instructions in the section "Determining HP-UX Minor Numbers", in the chapter "Adding Mass Storage Devices."

If you have trouble converting to hexadecimal, here's a table to help:

Decimal to Hexadecimal Conversion

Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal
0	0	11	b	22	16
1	1	12	c	23	17
2	2	13	d	24	18
3	3	14	e	25	19
4	4	15	f	26	1a
5	5	16	10	27	1b
6	6	17	11	28	1c
7	7	18	12	29	1d
8	8	19	13	30	1e
9	9	20	14	31	1f
10	a	21	15	32	20

For More Information

- For details of this process, see the section "Mounting or Unmounting File System" in the chapter "Managing the File System" in the *Series 300 HP-UX System Administration Tasks* manual.
- The *Series 300 HP-UX Installing Peripherals* contains Minor numbers for most disks in the "Worksheet Entries" sections.
- You can find useful information in the *HP-UX Reference* for the **mount**, **umount**, and **mknod** commands.

Setting Up LIF Devices

LIF (Logical Interchange Format) disks can accessed by HP BASIC/UX only when a device special file is set up in the `/dev/rmb` directory. The device special file must be a character device with a major number corresponding to the device driver and a minor number corresponding to the device's select code, bus address, unit and volume specifiers.

Prerequisites

Determine the device special file name. This can be any name as long as it is in the `/dev/rmb` directory. For example,

```
/dev/rmb/lif1
```

Connecting the Device

1. Go to the *Series 300 HP-UX Installing Peripherals*, "Adding Mass Storage Devices" chapter, and locate your device.
2. Read the introductory information. When you see "Turn your computer off," *you must shut down the system first, with* `/etc/shutdown -h`.
3. Follow the installation procedure to set the address and hook up the device.
4. Write down the select code, bus address, volume number, and unit number you used.

Collecting Vital Information

Before you can set up the device file, you must turn to "Worksheet Entries" in the same section of the *Series 300 HP-UX Installing Peripherals* of your HFS disk.

1. Write down these values from the table with "File Type" `c` (follow the instructions in the foot notes to replace any italicized letters in the values):
   a. Major Number
   b. Minor Number.

Create a Device Special File

1. Login as `root` (or become super-user with `su`).
2. Run `mknod`, substituting the italicized words with the values you determined earlier:

   `mknod` *device_file* `c` *Major_Number Minor_Number*

   For example, a CS80 device (select code 7, bus address 3) might use this command:

   ```
   mknod /dev/rmb/lif1 c 4 0x070300
   ```

   An AMIGO device (select code 14, bus address 7, volume 1) might use this command:

   ```
   mknod /dev/rmb/lif2 c 11 0x0e0710
   ```

Accessing the LIF Device

To access the device, change directories to that *msvs* (mass storage volume specifier), for example:

```
MSI ":,701"
```

For information on the *msvs*, refer to *Using HP BASIC/UX 6.2*.

Problems You Might Encounter

If you have a device that's not listed in the *Series 300 HP-UX Installing Peripherals* manual, you can calculate the Minor number by following the instructions in the section "Determining HP-UX Minor Numbers", in the chapter "Adding Mass Storage Devices."

If you have trouble converting to hexadecimal, here's a table to help:

Decimal to Hexadecimal Conversion

Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal
0	0	11	b	22	16
1	1	12	c	23	17
2	2	13	d	24	18
3	3	14	e	25	19
4	4	15	f	26	1a
5	5	16	10	27	1b
6	6	17	11	28	1c
7	7	18	12	29	1d
8	8	19	13	30	1e
9	9	20	14	31	1f
10	a	21	15	32	20

If you are MSI'd to a LIF device, and wish to return to your HFS directory:

```
MSI ":HFS" Return
```

For More Information

- The *Series 300 HP-UX Installing Peripherals* contains Minor numbers for most disks in the "Worksheet Entries" sections.
- You can find useful information in the *HP-UX Reference* for the `mknod` command.
- See *Using HP BASIC/UX 6.2* for more on accessing LIF devices.

Setting Up Printers

You can set up printers as local devices or as **spooled** devices. Spooled devices nate problem of more than one person or process accessing the printer at me time.

HP-UX is a multi-user system, it is best to set up *only* spooled devices ever, both are discussed here).

erequisites

Determine the printer name. This can be any unique file name, for example:

```
printer1
```

Hooking Up the Printer

1. Read the instructions for your printer. When you see "Turn your computer off," *you must shut down the system first, with* `/etc/shutdown -h`.
2. Follow the installation procedure in your printer's instructions to set the select codes and hook up the printer.
3. Write down the select code and bus address you used (you'll need them later).

Collecting Vital Information

Turn to "Worksheet Entries" in the same section of the *Series 300 HP-UX Installing Peripherals* of your printer.

1. Write down these values from the table (follow the instructions in the foot notes to replace any italicized letters in the values):
   a. The printer's model number (for example, an HP 2682A *Laserjet* has model number 2682A).
   b. Major Number.

Setting Up Spooled Printers

This section describes the process for the `reconfig` utility. For a detailed description of this utility, see the chapter "Reconfiguring the Kernel Customizing the Kernel" in the *Series 300 HP-UX System Administration Tasks* manual.

After you've connected the printer to the system,

1. Login as `root` (or become super-user with `su`).
2. Type: `/usr/bin/sam` (Return)
3. Using the `Next` and `Select` softkeys, select: `Peripheral Devices`
4. Select: `Printers and Plotters`
5. Select: `Add a Local Printer`
6. Answer the following questions:
   a. `Printer name`
   b. `Printer model/interface`
   c. `Printer device file name`
   d. `Printer priority`
   e. `Default printer?`
   f. `Printer class`
7. Exit the menu by pressing the `Previous Menu` softkey.
8. Exit `sam` by pressing the `Exit SAM` softkey.

Setting Up Local Printers (Not Spooled)

- Connect the printer to the interface on the backplane. (For example, to a built-in HP-IB.)
- Run `/usr/lib/rmb/rmbconfig`.

`rmbconfig` creates a device special file in `/dev/rmb`. (For example, with a built-in HP-IB on select code 7, a device file called `/dev/rmb/hpib7` is created.)

`rmbconfig` assigns permissions 000 to the device file. You must change the permissions (see `chmod` in the *HP-UX Reference*) appropriate to whom you want to access the printer.

Accessing Printers

To access a spooled printer, run the command in HP BASIC/UX:

```
PRINTER IS "|lp"
```

(The | is an HP-UX feature called a "pipe." It allows HP BASIC/UX to send output to an HP-UX command, which in this case, sends output to the spooled printer.) No output is printed until you close the pipe by reassigning the PRINTER IS device with, for example:

```
PRINTER IS CRT
```

To access a local printer, use the `PRINTER IS` command the same as in BASIC 5.1. For example, with a built-in HP-IB at address 1:

```
PRINTER IS 701
```

To return the printing to the screen, type:

```
PRINTER IS CRT
```

Problems You Might Encounter

If you have trouble accessing a local printer, check that there is an appropriate file in the `/dev/rmb` directory and that you have read and write permission for the file. If not, run `rmbconfig` again.

Setting Up Plotters

You can set up plotters as local devices or as **spooled** devices. Spooled devices eliminate problem of two or more persons or processes accessing the plotter at the same time, and since HP-UX is a multi-user system, it is best to set up *only* spooled devices (however, both are discussed here).

Prerequisites

Determine a plotter name. This can be any file name. For example,

```
plotter1
```

You must be familiar with an HP-UX editor, such as `vi`.

Connecting the Plotter

1. Go to the chapter "Adding Plotters & Other Graphics Devices" in the *Series 300 HP-UX Installing Peripherals* manual and locate your plotter or graphics device.
2. Read the introductory information. When you see "Turn your computer off," *you must shut down the system first, with* `/etc/shutdown -h`.
3. Follow the installation procedure to set the select codes and hook up the plotter.
4. Write down the select code and bus address you used (you'll need them later).

Setting Up Spooled Plotters

After you've connected the plotter to the system,

1. Login as `root` or become super-user with `su`.
2. Change directories (in HP-UX): `cd /usr/spool/lp/model` Return
3. Edit a file (using an HP-UX editor like `vi`), with the plotter name as the file name. For example, for a 7440A plotter:

   ```
   vi plotter1
   ```

4. Add the following to the file (this file enables the plotter to be accessed as a spooled device by HP-UX):

   ```
   shift;shift;shift;shift;shift
   for file in $*
   do
     cat "$file"2 >/dev/null
   ```

```
    done
```

5. Save the file (in `vi`, type: `:wq` Return)
6. Login as `root` (or become super-user with `su`).
7. Type: `/usr/bin/sam` Return
8. Using the `Next` and `Select` softkeys, select: `Peripheral Devices`
9. Select: `Printers and Plotters`
10. Select: `Add a Local Plotter`
11. Answer the following questions:
    a. `Printer name`
    b. `Printer model/interface`
    c. `Printer device file name`
    d. `Printer priority`
    e. `Default Printer?`
    f. `Printer class`
12. Exit the menu by pressing the `Previous Menu` softkey.
13. Exit `sam` by pressing the `Exit SAM` softkey.

Setting Up Local Plotters (Not Spooled)

- Connect the plotter to the interface on the backplane. (For example, if there's a built-in HP-IB.)
- Run `/usr/lib/rmb/rmbconfig` before entering HP BASIC/UX.

`rmbconfig` creates a device special file in `/dev/rmb`. (For example, with a built-in HP-IB on select code 7, a device file called `/dev/rmb/hpib7` is created.)

`rmbconfig` assigns permissions 000 to the device file. You must change the permissions (see `chmod` in the *HP-UX Reference*) appropriate to whom you want to access the plotter. For example, to let anyone access the HP-IB interface:

```
chmod 777 /dev/rmb/hpib7
```

Accessing Plotters

To access spooled plotters, run the PLOTTER IS command through an HP-UX command `lp` (running the output from a HP BASIC/UX command into an HP-UX command is called a "pipe"):

```
PLOTTER IS "|lp -dplot","HPGL",xmin,xmax,ymin,ymax
```

where *xmin*,*xmax*,*ymin*,*ymax* are the limit parameters for PLOTTER IS. (Use limit parameters because HP BASIC/UX treats any HPGL output directed through a pipe as a file. Plotter hard clip limits are not found when sent to the plotter, and an error occurs or the figure is distorted.) The `-dplot` is a parameter to the HP-UX `lp` command. It gives instructions to send the output to the destination "`plot`."

You must close the pipe to have the output plotted. For example, run:

```
PLOTTER IS CRT,"INTERNAL"
```

To access a local plotter, run the PLOTTER IS command in HP BASIC/UX. For example, a plotter on the built-in HP-IB at address 5:

```
PLOTTER IS 705,"HPGL"
```

To return plotting to the screen:

```
PLOTTER IS CRT,"INTERNAL"
```

Problems You Might Encounter

If you have trouble accessing a local plotter, check that there is a device file for the interface in the `/dev/rmb` directory and that you have read and write permission for the file. If not, run `rmbconfig` again.

For More Information

Plotter hard clip limits can be found in plotter manuals for all HP plotters.

For more on the HP-UX `lp` command, see the *HP-UX Reference*.

Accessing Instruments

Instruments and other devices have device special files created for the interface cards when the `rmbconfig` utility is run. These device special files enable you to access the instruments and other devices in the same manner as HP BASIC 5.1.

Prerequisites

When you hook up your peripherals and use the *Series 300 HP-UX Installing Peripherals* manual, "Turn the computer off" means: Shut down the system with `/etc/shutdown -h` and *then* turn off the computer.

Your instruments must be connected to a supported card in the backplane. Have this information ready for accessing the instrument:

- Select code
- Bus address.

How to Access Instruments

After you connect the instrument and power up the computer, run `rmbconfig`. The necessary device special file to access the instrument is created by `rmbconfig`.

`rmbconfig` assigns permissions 000 to the device file. You must change the permissions (see `chmod` in the *HP-UX Reference*) appropriate to whom you want to access the instrument.

Use the ASSIGN command to access instruments. For example, with

- Select code = 7
- Bus address = 23

you would type:

```
ASSIGN @pathname TO 723
```

where *pathname* is an I/O path name.

Problems You Might Encounter

Make sure `rmbconfig` runs to create the necessary device special file, or you cannot access the instrument.

For More Information

See the ASSIGN statement in the *HP BASIC/UX 6.2 Language Reference* for more on I/O path names and other detailed information.

Backing-Up the System

To back-up your system, you must follow the instructions in the chapter "Backing Up and Restoring the System" in the *Series 300 HP-UX System Administration Tasks* manual.

For More Information

For backing-up individual files or groups of files:

- To copy individual files in HP BASIC/UX, see "Backing-Up Individual Files to Flexible Disks."
- To archive files and directories on tape, see the HP-UX command `tar` in the *HP-UX Reference*.
- See `tcio` and `cpio` in the *HP-UX Reference* on how to copy files to tape and retrieve files from tape.
- To restore files from a disk or tape created with the BASIC BACKUP utility, see the *HP-UX Reference* on how to use the `cpio` command with the `-c` option.

Backing-Up Individual Files to Flexible Disks

If you need to back-up files to a flexible disk or a tape, you must determine its **msvs** ("Mass Storage Volume Specifier"), then use BASIC commands to copy to that disk or tape. (Setting up the disk or tape can be found in "Setting Up HFS File Systems" and "Setting Up LIF Devices.")

Backing-Up Individual Files

See *Using HP BASIC/UX 6.2* to help determine the msvs for your disk.

Here are some example commands for using disks for back-ups:

Example Statements with msvs

Example	msvs	Explanation
`CAT ":,700"`	:,700	Catalog the volume ":CS80,700" (the CS80 can be omitted; it is not required)
`COPY "AFILE:,700" TO "AFILE"`	:,700	Copy the file AFILE from the volume :CS80,700 to the current directory (and volume)
`SYSTEM$("MSI")`	n/a	Tells you what your current mass storage device is

5

Some Problems You Might Encounter

COPY can copy entire volumes from one LIF device to another LIF device. Be aware that it destroys the current contents of the volume to which you are copying.

For More Information

For more about mass volume specifiers, refer to *Using HP BASIC/UX 6.2.*

Accessing SRM Systems

An SRM (Shared Resource Manager) system is easy to access from HP BASIC/UX. You will need to install an SRM interface and connect it to the SRM server. When the `rmbconfig` program is run, it creates a device special file for you.

Note BASIC/UX supports both SRM and SRM/UX. However, BASIC/UX supports only an SRM interface, not a LAN card, with SRM/UX. This is different from BASIC/WS, which supports either a LAN card or an SRM interface with SRM/UX.

HP BASIC/UX only supports one SRM card in the backplane — only one SRM system can be accessed. If more than one SRM card is present, the one with lowest select code value is selected.

The rmbconfig Utility Creates the Device Special File

When your SRM system is connected to the SRM interface card, run:

```
/usr/lib/rmb/rmbconfig
```

This utility creates a device special file in `/dev/rmb`.

Accessing the SRM

`rmbconfig` assigns permissions 000 to the device file. You must change the permissions (see `chmod` in the *HP-UX Reference*) appropriate to who you want to access the SRM.

Access the SRM in BASIC/UX in the same manner as BASIC/WS. For example,

```
MASS STORAGE IS ":REMOTE,22"
```

establishes access to the SRM connected at select code 22.

If You Use Both LAN and SRM, Change One Select Code

Both LAN (Local Area Network) and SRM cards are shipped with default select code 21. If you use both cards in your system, you must change the select code of one of them (preferably the SRM to, for example, 23).

For More Information

See "BASIC/UX System Administration Tasks" for more on the `rmbconfig` utility.

Using Template Files

There are several template files available for customizing your system.

Prerequisites

Change directory to: `/etc/newconfig/rmb`

Description of Template Files

Listing this directory shows some files you can use for customizing your system (note those which can and cannot be edited in the BASIC EDIT mode):

`rmbrc`	Template for the global `/usr/lib/rmb/rmbrc` file. You can copy this file to `/usr/lib/rmb` and customize it in the EDIT mode. See "Customizing Your BASIC/UX Session" for more details.
`d.rmbrc`	Template for a local `.rmbrc` file. A user can customize his/her own environment by copying this file into their $HOME directory, and using the EDIT mode. The file only affects HP BASIC/UX sessions of the specific user. See *Using the BASIC/UX System* for more information.
`d.profile`	An example HP-UX `.profile` script that initializes either a Bourne shell (/bin/sh) or Korn shell (/bin/ksh) when logging in. This version automatically starts the X Window System (if

	available) if the user logs in as **xrmb**. You can copy this file into a $HOME directory and customize it using an HP-UX editor.
`d.hpwmrc`	An example start-up default file for the X window manager **hpwm**. This file can be copied into a $HOME directory as `.hpwmrc` and can be customized with an HP-UX editor.
`d.mwmrc`	An example start-up default file for the X window manager **mwm**. This file can be copied into a $HOME directory as `.mwmrc` and can be customized with an HP-UX editor.
`d.x11start`	An example control file for starting the X Window System under the **mwm** window manager. This file can be copied into a $HOME directory as `.x11start` and can be customized with an HP-UX editor.
`d.Xdefaults`	An example default file for controlling the X Window System. This file can be copied into a $HOME directory as `.Xdefaults` and can be customized with an HP-UX editor.
xrmbcolormap	A file used to pre-allocate color map entries for using HP BASIC/UX in X. It is used by **xinitcolormap** to select appropriate default colors for HP BASIC/UX.
rgb.rmb	An example file to add custom HP BASIC/UX colors to the default the X Window System color database (see rgb.README).
rgb.readme	A file with instructions on how to modify the X Window System database.

For More Information

See *Using the X Window System* for more on X configuration files.

Selecting the BASIC/UX Fonts

This section explains how to change the default HP BASIC/UX fonts.

Selecting the Font You Want

You may wish to change the default fonts that HP BASIC/UX uses. There are two different approaches to this based on the environment in which you are running.

In the X environment, simply use the font resource in your `$HOME/.Xdefaults` file. For example, if you wanted katakana characters on a high-resolution display, you might have the following line in your `$HOME/.Xdefaults` file:

```
rmb*font:  kana.10x20
```

or, you can specify the font as an option on the `rmb` command line:

```
rmb -fn kana.10x20
```

Available fonts for use in the X environment can be found in the directory:

```
/usr/lib/X11/fonts
```

In the console environment, you must copy the desired font from one of the subdirectories located in `/usr/lib/raster` into the file `lp.8U.scf` located in one of the subdirectories under `/usr/lib/rm/fonts`. For example, to get katakana characters on a high-resolution display (as above) perform the following copy:

```
cp /usr/lib/raster/10x20/SWF/kana.8K.scf /usr/lib/rmb/fonts/10x20/lp.8U.scf
```

The sub-directories under `/usr/lib/rmb/fonts` correspond to the following screen resolutions:

Sub-Directories	Screen Resolution
L6x15	512 x 400
8x16	1024 x 768
10x20	1280 x 1024

For More Information

See the *Using the X Window System* manual for information on selecting fonts in the X environment.

6

BASIC/UX Utilities

This chapter describes the utilities provided with HP BASIC/UX.

Before You Use the Utilities

HP BASIC/UX utilities are found in the directory:

```
/usr/lib/rmb/utils
```

Some utilities are complete BASIC programs: to use these programs, you must LOAD them into HP BASIC/UX and type `RUN`.

Other utilities are CSUBs (Compiled SUBprograms), and must be called from within a BASIC program.

- Use the BASIC editor to view programs like LIF_UTIL to see how CALL and LOADSUB are used.
- *Developing CSUBs for HP BASIC/UX 6.2* or the keyword descriptions for CALL and LOADSUB in the *HP BASIC 6.2 Language Reference* contain more CSUB information.

Prerequisites

Utilities are for experienced HP BASIC programmers. You should be able to:

- Configure a hardware system.
- Boot HP BASIC/UX.
- Initialize disks and copy programs.
- Write BASIC programs or use applications programs.
- Handle SUBs and CSUBs within a conventional program.
- Interpret what the computer is doing, or ought to be doing.

It is helpful to have performed the tasks described in *Using HP BASIC/UX 6.2*, and the *HP BASIC 6.2 Programming Guide.*

Utilities for LIF Only

Be aware that several of the utilities use PHYREAD or PHYWRITE (see "(PHYREC) Physical Record CSUB") and *give an error if used with SRM or HFS devices.*

Utilities Overview

The table in this section provides a brief description of the utilities. A complete description of each utility can be found in the "Utilities Reference" section next in this chapter.

Utility	Short Description
BITMAP_S	Replaces BASIC/WS GDUMP_C utility: dump a raster image to a file in proper format for Starbase (so you can use Starbase utility **pcltrans**).
BPLOT	Store and load rectangular "blocks" of raster data from graphics frame buffers.
GDUMP_R	CSUB utility to dump graphics raster images to a printer and rotate them 90 degrees.
LEX_AID	Create your own lexical order tables.
LIF_UTIL	A combination of utilities to ■ Dump mass storage records ■ Create non-standard sized LIF directories ■ Restore files previously purged from a LIF directory ■ Repack a LIF disk to give you contiguous available space at the end of the disk ■ Verify if the current MSI device is a LIF disk ■ Clear the directory of a LIF disk
PHYREC	A CSUB to perform bit-by-bit copies of an integer array in memory to a physical record on a LIF mass storage media (and vice versa).
status utilities	Four programs to display a formatted listing of status registers for: ■ HP-IB interface ■ RS232 interface ■ GPIO interface ■ data files

Utilities Reference

This section contains detailed descriptions of each of the HP BASIC/UX Utilities.

(BITMAP_S) Bitmap Store Raster Dump CSUB Utility

BITMAP_S (found in the `/usr/lib/rmb/utils` directory) is a CSUB utility that dumps a screen to a file in the Starbase bitmap format (see the *Starbase Reference*). Use this utility to print dumps of the screen.

There is no correlation between

- The Starbase bitmap file format supported by this CSUB and the file format used by the BASIC system (GSTORE/GLOAD).
- This CSUB and the HP BASIC Workstation GDUMP_C utility, except that both offer a path to using the PaintJet color printer.

Calling the CSUB

Syntax for BITMAP_S:

`Bitmap_store(`*from_ds*`,"`*to_file*`")`

6

where:

from_ds	The device (screen) whose frame buffer you want to dump. You must supply a value (for example in X Windows: the window number). It does not have to be the current PLOTTER IS device.
to_file	is the name of a file or a pathname where the bitmap data is to be stored. It may also be a string beginning with a pipe, for example: `"| pcltrans -C	lp -oraw"`

Here is an example of the procedure to load the CSUB and run it:

1. Copy the CSUB into the current directory:

```
COPY "/usr/lib/rmb/utils/BITMAP_S" TO "BITMAP_S"
```

2. Include statements similar to (the following example uses window number 600 as the screen to be dumped, and "save.bits" as the name of the file to store the bitmap data):

   ```
   400  LOADSUB ALL FROM "BITMAP_S"
   410  !
   420  Bitmap_store(600,"save.bits")
   430 !
   ```

If save.bits is replaced with a pipe, (see description in *to_file* above) you can have the file dumped directly to the printer.

Sending the Bitmap File to the Printer

After dumping to a file, use the HP-UX pcltrans command to dump to a printer from a HP BASIC/UX process (save.bits is the example file name):

```
EXECUTE "pcltrans -C save.bits | lp -oraw"
```

See the PCL Formatter section of *Starbase Device Drivers Library* for a complete list of supported printers (all that support pcltrans, for example: the PaintJet, HP 3630A).

Calling the CSUB Interactively

Outside of a program, load the CSUB with LOADSUB, and call it with (/tmp/mypicture stores the bitmap data):

```
CALL Bitmap_store(CRT,"/tmp/mypicture")
```

Deleting the CSUB from Memory

Run this statement:

```
DELSUB Bitmap_store
```

For More Information

For tips on how to use the pcltrans command, see the *Starbase Reference*.

(BPLOT) Raster Store and Load

BPLOT is a CSUB utility to store (Bstore) and load (Bload) rectangular "blocks" of raster data (using numeric arrays; not supported on terminals). Bstore and Bload are similar to GSTORE and GLOAD, except:

- They affect *only* a specified portion of the frame buffer; GSTORE and GLOAD affect the entire buffer.
- They also let you to specify a replacement rule (logical operation such as AND, OR, EXCLUSIVE OR) to combine the source and destination pixels; GSTORE and GLOAD are always dominant (they overwrite any existing destination pixels).

Calling the CSUB

1. Copy the CSUB into the current directory:

```
COPY "/usr/lib/rmb/utils/BPLOT" TO "BPLOT"
```

2. Include Bstore, Bload in your program as needed. For example:

```
400  LOADSUB ALL FROM "BPLOT"
410  !
420  Bstore(Int_array(*),X_pixels,Y_pixels)
430  !
```

6

Using Bstore

`Bstore` stores a rectangular area of the frame buffer in an INTEGER array. For example:

```
Bstore(Int_array(*),X_pixels,Y_pixels,Rplcmt_rule,X_start,Y_start)
```

where:

`X_pixels`	is an INTEGER for width of the rectangular raster area (in pixels).
`Y_pixels`	is an INTEGER for height of the rectangular raster area (in pixels).
`Rplcmt_rule`	is an *optional* INTEGER parameter that specifies how the pixels are placed into the array.

`X_start`	is an *optional* REAL parameter that specifies the X coordinate of the upper left corner of the area to be stored: in *current graphics unit of measure* (not pixels).
`Y_start`	is an *optional* REAL parameter that specifies the Y coordinate of the upper left corner of the area to be stored: in *current graphics unit of measure* (not pixels).

Bstore Details. X_pixels and Y_pixels values are placed in the INTEGER array `Int_array` as one byte per pixel for all frame buffers. Displays with non-square pixels (medium-resolution Series 300 displays) require twice as much array variable space as their square-pixel counterparts for the HP BASIC Workstation.

If part of the area is outside the current clip limits, then only the portion of the frame buffer within these limits is stored in the array—the remainder of the array remains unchanged.

The optional parameter `Rplcmt_rule` specifies the replacement rule to use in combining (these rules correspond to the rules provided by CRT CONTROL register 14 for bit-mapped displays):

- Source bits (frame buffer with `Bstore`; array with `Bload`)
- Destination bits (current contents of the array with `Bstore`; frame buffer with `Bload`).

The following table shows the rule values you can use.

Parameter Value	Effect on Destination Bits
0	All bits set to 0
1	Source AND Destination
2	Source AND NOT Destination
3	Source (Default)
4	NOT Source AND Destination
5	Destination
6	Source EXOR Destination
7	Source OR Destination
8	Source NOR Destination
9	Source EXNOR Destination
10	NOT Destination
11	Source OR NOT Destination
12	NOT Source
13	NOT Source OR Destination
14	Source NAND Destination
15	All bits set to 1

If no replacement rule is specified, then rule 3 (the default) is used.

If X_start and Y_start are not specified, the current graphics position is used. If you want the source bits to be placed into the destination without modification, then specify a value of 3 for the replacement rule parameter.

Using Bload

Subroutine `Bload` loads a rectangular area of the frame buffer from an INTEGER array. For example:

```
Bload(Int_array(*),X_pixels,Y_pixels,Rplcmt_rule,X_start,Y_start)
```

The parameters are the same as Bstore, except that the statement *loads* a rectangular area on the frame buffer with the current contents of the INTEGER array `Int_array`.

Problems You Might Encounter

The array variables for Bstore (X_pixels, Y_pixels) must be of sufficient size to store the specified pixels, or an error is reported (error 16—improper dimensions).

(GDUMP_R) Rotated Graphics Dump

This CSUB utility dumps graphics raster images to a printer. It provides the same function as the DUMP GRAPHICS statement, except that it "rotates" the image 90 degrees before sending it to the printer.

Using GDUMP_R

1. Copy GDUMP_R into the current directory:

   ```
   COPY "/usr/lib/rmb/utils/GDUMP_R" TO "GDUMP_R"
   ```

2. Include the Gdump_r routine in your program, for example (dump the image from the display raster PLOTTER IS CRT,"INTERNAL" to a printer at device selector 701):

```
400  LOADSUB ALL FROM "GDUMP_R" !  Load the CSUB into memory.
410  !
420  Gdump_r(CRT,701)           !  Dump raster (select code 1) to
430                             !  HP-IB printer (select code 7,
440                             !                 address 1).
```

Here is another example. This time, the raster at select code 6 is sent to a printer at select code 9 (there is no address, since this is a serial interface).

6

```
560  Gdump_r(6,9)               !  Dump raster (select code 6) to
570                             !  serial printer (select code 9).
```

If you specify 0 as the printer select code, the current DUMP DEVICE IS device is used. This allows you to dump to a pipe or file. For example, dump the CRT to the lp spooler:

```
DUMP DEVICE IS "|lp -oraw"
...
...
Gdump_r(CRT,0)
```

Deleting the CSUB from Memory.

```
DELSUB "Gdump_r"
```

(LEX_AID) Creating a Lexical Order Table

The LEX_AID (found in `/usr/lib/rmb/utils` directory) program simplifies the creation of user-defined lexical tables. You may wish to make modifications to the program to suit your particular needs.

Steps in Creating a Table

1. Use the LEXICAL ORDER TABLE WORKSHEET in the "User-defined LEXICAL ORDER" section of the "String Manipulations" chapter of *BASIC/UX Programming Techniques*. To avoid confusion, always assign sequence numbers in ascending order. Also mark special cases ("1 for 2", "Don't Care", etc).
2. When you have completed your lexical order table, scan it for blocks of consecutive sequence number assignments. For example, the control characters (codes 0 through 31) generally retain their original sequence number. The LEX_AID program has a "FILL BLOCK" mode that will simplify assigning groups of sequence numbers.
3. Run the LEX_AID program to create your table. (Be sure to save your table when you are finished.)
4. Install the table created by this program.

The remainder of this section further expands steps 3 and 4.

(LEX_AID) Creating a Lexical Order Table

Running LEX_AID

LEX_AID uses the softkeys to provide menu selections. One of the following main menus appears on the screen when the program is run.

Key Labels for ITF Keyboards

```
Seq   Mode   Show  Show   Show   Fill   Get    Quit  Save   List
Num   Index  Seq   Mode   Table  Block  Table        Table  Table
```

The options work as follows:

`Seq Num`	Allows a sequence number to be assigned to a character. The program prompts for the character, and then for the sequence number to be assigned. Whenever the program is prompting for a character, either the character may be typed from the keyboard or its decimal value may be entered.
`ModeIndex`	Displays a sub-menu (shown later) from which the special mode entries can be selected.
`Show Seq`	Prompts for a character then displays the character, its value and its current sequence number.
`ShowMode`	Displays all of the currently defined mode table entries.
`ShowTable`	Displays the current assignments on the CRT. The mode type and mode index are displayed as a single value to save space.
`FillBlock`	Prompts for the beginning sequence number and the first and last characters of the block to be filled. Thus a group of characters can be assigned consecutive sequence numbers quickly.
`GetTable`	Loads a previously defined lexical order table from the disk.
`Quit`	Returns you to normal keyboard mode and terminates the program.

`SaveTable`	Saves the currently defined lexical order table to the disk.
`ListTable`	Prints the currently defined lexical order table to the current PRINTER IS device.

Requesting **`ModeIndex`** displays one of the following menus.

Key Labels for ITF Keyboards

```
Don't  1for2  2for1  Accent   Normal
Care
```

Note that the keys shown without labels may have previously defined "typing-aid" definitions and labels.

If you are not familiar with these mode types, you may wish to review the aforementioned "User-defined LEXICAL ORDER" section. With either keyboard, the options work as follows:

`Don'tCare`	Requests the character to be ignored for collating purposes.
`1 for 2`	Asks if a suitable entry already exists. If the current character can use a previously defined "1 for 2" entry, the same mode table entry can be used. Answering "Yes" to the question prompts for the new character and then prompts for the character which already has the proper entry. Answering "No" to the question prompts for the character and then asks for the number of secondary characters. Each character and its sequence number is then entered.
`2 for 1`	Asks if a suitable entry already exists. If several characters use the same secondary character (as in the GERMAN lexical order), the same mode table entry can be used for more than one character. Answering "Yes" to the question prompts for the new character and then prompts for the character which already has the proper entry. Answering "No" to the question prompts for the sequence number of the second character (both upper and lower case).

(LEX_AID) Creating a Lexical Order Table

`Accent`	Requests the priority (0 through 63) and then the character to be assigned the accent priority.
`Normal`	Allows you to cancel a special mode entry that was mistakenly assigned to a character. No provision is implemented in the LEX_AID program to actually delete the mode table entry.

With practice, you can create your own lexical order tables with LEX_AID. Be sure to save the table when you are finished.

Installing Your Lexical Order Table

Having created a table and stored it on disk, you can install it with the following BASIC program.

```
10    INTEGER Table(0:320)
20    ASSIGN @File TO "MYTABLE"  ! Open file
30    ENTER @File;Table(*)       ! Read file
40    ASSIGN @File TO *          ! Close file
50    LEXICAL ORDER IS Table(*)
60    !
70    END
```

You may need to replace the name "MYTABLE" with the name of the file in which you have stored the table you have created with LEX_AID.

The lexical order remains in effect until a SCRATCH A or another LEXICAL ORDER IS statement is executed, or the system is re-booted.

LEX_AID Example

This example uses LEX_AID to create a case-independent lexical order. Both upper- and lower-case characters collate together.

1. Load and run the LEX_AID program.
2. Type the `FillBlock` softkey. Enter 0 for the initial sequence number, 000 for the value of the first character, and 255 for the value of the last character. As each character is assigned a sequence number, the character will be flashed on the screen.
3. Type `FillBlock` again. Enter 65 for the starting sequence number, 097 for the value of the first character, and 122 for the value of the last character.
4. Type `SaveTable` and save the table under the name "NOCASE." The newly created table will be saved on the disk as a BDAT file.
5. Execute `SCRATCH`, then enter and run the following program:

```
10    INTEGER Table (320)
20    ASSIGN @File to "NOCASE"
30    ENTER @File;Table(*)
40    LEXICAL ORDER IS Table(*)
50    IF "Hello" = "hElLO" THEN
60      PRINT "NO DIFFERENCE"
70    ELSE
80      PRINT "LEX DOESN'T WORK"
90    END IF
100   END
```

The message—`NO DIFFERENCE`—is printed. Similar results could have been achieved by using the UPC$ and LWC$ functions.

LEX_AID Details

Applications needing specialized collating sequences can be simplified by the use of the LEXICAL ORDER IS statement.

It is often easier to separate lexical tables into two arrays, one for the sequence number and a second for the mode entry. This simplifies the calculations of the mode entry. The two arrays are then merged into a single INTEGER array. You can list the LEX_AID program to see the operation.

(LIF_UTIL) LIF Utility

LIF_UTIL is a utility that contains elements of previous HP BASIC Workstation utilities:

DUMP	dump mass storage records in a variety of formats.
INITIALIZE	create non-standard sized LIF directories.
REPACK	repack a LIF disk for contiguous available space at the end of the disk.
UNPURGE	restore files previously purged from a LIF directory.
VERIFY_LIF	verify if the current MSI device is a LIF disk.
ZAP	clear (destroy files and data) the directory of a LIF disk.

Loading and Running the Utility

1. `LOAD "/usr/lib/rmb/utils/LIF_UTIL"` Return
2. `RUN` Return

   or press the RUN key (f3 in the System menu).

Choosing Options in the Utility

6

1. Move the => cursor to point to the desired option by pressing the ▼ or ▲ cursor arrow keys.
2. Press Return to select the option.

You can also use the softkeys labeled `Next` or `Previous` to move the cursor, followed by the softkey labeled `Select` to choose an option.

Formatted Record Dump

This part of LIF_UTIL dumps mass storage records in a variety of formats. To understand this section you may need to have a knowledge of LIF (Logical Interchange Format) disks.

Steps to Dump the Contents of a LIF Volume Record

1. Select the `Dump contents of a LIF volume record` option in the main menu.
2. Insert the disk which contains a record to be dumped into the disk drive.
3. Type `CONTINUE` or press the `Continue` softkey (SYSTEM menu of an ITF keyboard). If your disk is *not* in the drive designated by the current *msvs*, enter an appropriate msvs before you continue.
4. Select an alternative (see description below).

Several alternatives for dumping a record are selected via softkeys. On an ITF keyboard, toggle between softkey menus to see all options. The default dump is hexadecimal.

`RECORD#`	This lets you select the record you wish to dump.
`Hex`	Dumps a record in hexadecimal format.
`Hex/ascii`	Dumps a record in hexadecimal/ASCII format, where control characters are masked out and replaced by periods.
`LIF sys`	Dumps the LIF volume label in the correct format.
`LIF dir`	Causes subsequent dumps to have a LIF directory format.
`CAT`	Displays a catalog of the disk.
`N+1, N-1`	Allows you to increment or decrement the current record number and dumps a record according to the current format.
`Integer`	Dumps a record in an integer format.
`Oct/ascii`	Dumps a record in an octal/ASCII format. Control characters are masked out and replaced by periods.
`Main Menu`	Returns you to the main menu.

If the program pauses after the fourth entry, type `CONTINUE` or press the `Continue` softkey (SYSTEM menu on an ITF keyboard) to print the rest of the record.

(LIF_UTIL) LIF Utility

Extended Mass Storage Media Initialize

When you select `Initialize a LIF disk`, you should only be creating non-standard-sized LIF volumes. Standard LIF directories should be created using the INITIALIZE keyword (see the *HP BASIC 6.2 Language Reference*).

1. On the prompt, enter the *msvs* for the device which contains the disk you wish to initialize, or press Return to accept the current msvs.
2. Insert the disk to be initialized and press CONTINUE (or the `Continue` softkey [SYSTEM menu of an ITF keyboard]).
3. Select the action with a softkey: `YES`, `NO`, or `EXIT`.
4. Wait while the initialization actions take place. *Do not interrupt the process.* If you inserted a previously initialized disk, some additional options are provided:
   - OKAY (to initialize).
   - RESTART (to begin this part of the utility again).
   - EXIT (to leave this part of the utility).
5. After initialization, enter a volume name (six alphanumeric characters, uppercase, alpha first character). The default is six blanks.
6. You then enter a directory length and type Return (or the `Continue` softkey. The default is 14 physical records.
   - Avoid creating a directory too large to fit onto your disk.
   - Keep directory lengths between 1 and the maximum allowable value for your disk.
   - If you selected the `NO` option, you still have an option to change the volume label within the limitations.

The program selects the optimum interleave factor for the given msvs.

Repack a LIF Disk

When you select `Repack a LIF disk` from the main menu, the currently MSI'd disk is repacked to give you contiguous available space at the end of the disk.

Pay attention to available softkey options.

1. You are prompted for the msvs. Type (Return) only for the current msvs.
2. When prompted, insert the disk to be repacked.
3. Type (CONTINUE) (or the `Continue` softkey on the System menu). Note the options via softkeys (User 1 menu) and proceed. Arrows in the extended directory catalog point to purged files that are lost during repacking.

Restore Purged Files

When you select `Restore purged files` from the main menu, an extended CAT is displayed.

1. Follow the prompts.
2. Choose the correct type of file (ASCII of BDAT).
3. Note what has and has not been purged during the extended CAT display. Arrows point to files previously purged.

Check Logical Interchange Format

1. Select: `Determine if current msvs is a LIF disk`
2. You are prompted to insert the disk to be checked.
3. Type (CONTINUE) or the `Continue` softkey (SYSTEM menu of an ITF keyboard).
4. Wait a moment. Then, the status of your disk is displayed. It either meets the LIF standard or it does not.

(LIF_UTIL) LIF Utility

Clear the Directory of a Disk

Clear the directory of a disk by placing a −1 in the file type field of the first directory entry to denote the logical end of the directory (similar to INITIALIZE).

1. Select `Remove all files and data from a LIF disk` option in the main menu.
2. When prompted, enter the msvs (CONTINUE accepts default).
3. Then insert the correct disk, and type the softkey for `YES` to ZAP the disk. *Note: This will destroy all files and data on the disk.* (You can also type the softkey for `NO` to terminate this function.)

6

(PHYREC) Physical Record CSUB

PHYREC copies a bit-by-bit memory integer array to a sector on a LIF mass storage media, and vice versa. PHYREC contains two CSUBs:

Phyread	Copies data from the current msvs into an integer array.
Phywrite	Copies data from the integer array to the specified msvs.

Indiscriminate use of **Phywrite** can cause loss of valuable data. Considerable programming and computer experience are required to effectively use PHYREC.

Using PHYREC

1. Copy the CSUB into the current directory:

```
COPY "/usr/lib/rmb/utils/PHYREC" TO "PHYREC"
```

2. Include Phyread, Phywrite in your program as needed. For example:

```
400  LOADSUB ALL FROM "PHYREC"
410  !
420  Phyread(Sector,INTEGER Int_array(*))
430  !
```

(PHYREC) Physical Record CSUB

Details of Phyread

Phyread copies data from the current msvs into an integer array of one to six dimensions. For the example,

```
Phyread(Sector,INTEGER Int_array(*))
```

Sector	Replace Sector with a numeric expression. It is evaluated to a real, and rounded to specify the sector (address) on the disk where the copy begins. See "How LIF Disk is Structured" for more information on sectors. A sector is assumed to contain 256 bytes. For example, a disk with 1024-byte physical sectors is accessed by four 256-byte logical sectors for each 1024-byte physical sector.
Int_array(*)	The disk is read, beginning with the starting sector, and data is copied into an integer array (Int_array) in a row-major order. Data is copied until each array element is occupied or until an attempt is made to read beyond the end of data on the media (in this case, an error is reported).

When using arrays, be sure to use 0 or 1 OPTION BASE as required by the nature of your data. See *HP BASIC 6.2 Programming Guide*.

Details of Phywrite

The Phywrite CSUB works similiar to Phyread, except data is copied from the array to the media (an error is reported on an attempt to write beyond the end of the media).

```
Phywrite(Sector,Int_array(*))
```

How LIF Disk is Structured

The following tables contain information about the contents of LIF disks. Use them to conceptualize how information is stored.

Structure of a LIF Disk

Sector Number	Description
0	Volume label
1	Empty (set to 0)
2-15	Directory. 14 sectors is the default, with 112 file entries (each sector is 256 bytes, 128 words, long).
16-xx	Actual space for files.

How a LIF Volume Label is Structured. In sector number 0, the volume label contains the following words:

Structure of a LIF Volume Label

Word Number (2 bytes)	Contents
0	LIF disk identifier (100000 Octal)
1,2,3	Volume label: 6 characters long
4,5	Directory start address (default is 2)
6	Reserved (10000 Octal)
7	0
8,9	Length of directory (default is 14 sectors)
10	Reserved
11	0
12-20	Information on disk hardware (level 1 extensions)
21-126	Reserved (0)
127	Reserved

How a LIF Directory is Structured. The following table shows how sectors relate to file entries on a LIF directory (each sector is 128 words and each file entry is 16 words):

Structure of a LIF Directory

Sector Number	Contents
2	File entries 1-8
3	File entries 9-16
4	File entries 17-24
...	...
15	File entries 105-112

How File Entries are Structured.

Structure of a LIF File Entry

Word Number	Contents
0-4	File name. Each byte is one character; 10 characters are allowed.
5	File type. Codes for each file type are found here. Those supported by BASIC include (type follows code): 1 ASCII 0 (empty) -1 Logical end of directory -5775 BIN -5791 BDAT -5808 PROG -5822 SYSTM -5813 HP-UX File type codes are also used to indicate end of directory and an empty file entry.
6-7	File address. Numeric entry (two words) specifies the file's contents and start on the disk.
8-9	File length. Numeric entry (two words) specifies the file's size.
10-12	Creation time. Used to store the time when the file was made (three words: 12 BCD digits in the form YYMMDDHHMMSS).
13	Volume number. Used for operating systems that support multiple volumes mapped logically (not used by BASIC).
14	Protect code. A two-character password used for the file (0 for ASCII; default is 2 blanks for other file types).
15	Defined record size. ■ 0 implies BDATfile with 1 byte record size ■ ignored for ASCII but set to 0 for other system compatibility ■ ignored for PROG but set to 128 for future compatibility (it may appear on the display as 256).

Comparing LIF Directory Values to a CAT List. The following table shows how LIF directory entries compare to those done in a CAT list.

Differences Between LIF Directory and CAT Values

Field Value	File Type	Directory Value	CAT
file type	ASCII	1	ASCII
	Empty entry	0	
	Logical end of directory	-1	
	BDAT	-5791	BDAT
	BIN	-5775	BIN
	PROG	-5808	PROG
	SYSTEM	-5822	SYSTM
	HP-UX	-5813	HP-UX
file address	BDAT	n	n + 1
	All others	n	n
file length	BDAT	Number of sectors	Number of sectors user sets at creation. Read words 4 and 5 of system sector. Stores as first sector of file.
protect code	ASCII	0	
	SYSTEM	Upper word start address	
	HP-UX	Upper word eof byte	
	Others	2 character code	

Differences Between LIF Directory and CAT Values (continued)

Field Value	File Type	Directory Value	CAT
defined record size	ASCII	0	256
	BIN	128	256
	PROG	128	256
	HP-UX	Lower word eof byte	
	BDAT	If n=0	1
		If n>0	2n
		(length in words)	(length in bytes)
	SYSTM	Lower word start address	256

The Status Utilities

Three utility programs (found in **/usr/lib/rmb/utils**) are available that let you see the contents of status registers: HP-IB Interface (HPIB_STAT), RS-232 Interface (RS232_STAT), and GPIO Interface (GPIO_STAT). Each program works in a similar manner. Each requires that you enter fundamental information about such things as device select codes.

The best way to learn how to use these programs is to load them and call assorted functions via softkeys. (Remember to toggle between the ITF-keyboard User 1 and User 2 menus.)

Pay attention to softkey labels. They are completely self-explanatory. Use them for calling functions. If an interface is not present, type Return with no file entry to exit. Load any of the three utilities as shown below and execute them:

`LOAD "HPIB_STAT"` (works *only* with an HP-IB interface)

or

```
LOAD "RS232_STAT"
```

or

```
LOAD "GPIO_STAT"
```

Unlike utilities which can be rather tedious to use, these three are simple and friendly. Yet they can provide valuable information about the contents of Status Registers.

Unsupported BASIC/WS Utilities

The following table lists some of the BASIC/WS utilities that are not supported on HP BASIC/UX. In many cases, the functionality of the utility is covered by HP BASIC/UX in another capacity.

Unsupported Utilities

BASIC/WS Utility	**Description**
BACKUP/RESTORE	Use HP-UX back-up. See "Backing-Up the System" in Chapter 5.
HFSCK	Use `fsck`. See the section "Checking File System Consistency Running the fsck Command" in the chapter "Managing the File System" in the *Series 300 HP-UX System Administration Tasks* manual.
MKHFS	See the section "Making a File System Creating a File System" in the chapter "Managing the File System" in the *Series 300 HP-UX System Administration Tasks* manual.
CAT	Use CAT TO statement. See *HP BASIC 6.2 Language Reference*.
CBACKUP	Not supported.

Unsupported Utilities (continued)

BASIC/WS Utility	Description
CREATE	See *HP BASIC 6.2 Language Reference* for statement usage.
FBACKUP	Not supported.
FONT_ED	Not supported.
INFO	Not supported.
INITIALIZE	See LIF_UTIL and *HP BASIC 6.2 Language Reference* entry for INITIALIZE.
LISTER	Functionality supported by: 1. Load file into BASIC/UX 2. Set: `PRINTER IS "|pr |lp"` (see *HP-UX Reference*) 3. Use the LIST statement (see *HP BASIC 6.2 Language Reference*).
MASS_STOR	FILE SIZER, PURGE, Change MSVS, Extended CAT, Change Volume Label are not supported. UNPURGE, REPACK and ZAP are a part of LIF_UTIL.
MEM_UTILS	Not supported.
TAPEBACKUP	Not supported.
82905DUMP	Not supported.
DISK_UTIL	Not supported. See separate entries for MKHFS, HFSCK, and BACKUP/RESTORE.
VME Driver	Not supported.
Loader Utility	Not supported.
CONFIGUR	Not supported.

A

HP-UX Command Reference

This appendix contains the HP-UX reference pages for:

- `rmb`
- `rmbbuildc`
- `rmbclean`
- `rmbconfig`
- `rmbdfile`
- `rmbkernel`
- `rmbkill`

Notes

NAME

rmb, rmbhil, rmbkbd, rmbtmr, rmbxfr - HP BASIC/UX environment

SYNOPSIS

rmb [**-beikntN**] [**-c** *file*] [**-l** *opt*] [**-r** *num*] [**-w** *num*[KM]] [*X Windows options*] [*autostart_file*]

Remarks:

This command requires installation of optional BASIC/UX software (not included with the standard HP-UX operating system) before it can be used.

The keywords **disk** and **disc** are parsed as equivalents by *rmb* and related commands and programs.

DESCRIPTION

This command invokes the HP BASIC/UX interpreter which can be used to execute BASIC commands or run BASIC programs.

The BASIC EXECUTE command is used to temporarily exit the BASIC environment and spawn a new Bourne shell from which any number of HP-UX commands can be executed. Ctrl-D terminates the shell and returns to BASIC.

Options

The command line options are:

-b This option causes *rmb* to print a screen on startup that resembles the bootrom screen. This screen contains information about hardware and software configurations. Several additional fields have been added to the standard bootrom screen:

Field	Contents
workspace	size of the *rmb* workspace
swap	information about swap device locations
mounted disks	information about mounted disk locations
HP-UX	information about the HP-UX version

In addition, interface card lines provide information about whether a swap device is on the interface (which precludes BURST I/O), and which device file (if any) is used to access the interface.

-c *file*

Specifies that the file *file* should be used as the configuration file rather than **$HOME/.rmbrc**.

-e Enables "ignore compatibility errors" mode. This causes incompatible statements from the BASIC workstation to be ignored rather than flagged as errors.

-i Causes *rmb* to run **/usr/lib/rmb/rmbclean** at startup time to reclaim orphaned lockfiles and IPC resources.

-k Causes *rmb* to run **/usr/lib/rmb/rmbconfig -b** at startup time to update the kernel configuration file (**/usr/lib/rmb/rmbbootinfo**).

-l *opt*

This option specifies for *rmb* to attempt to lock the indicated program segments into memory. This can result in an increase in program performance at the expense of other programs on the machine. The *opt* parameter can be any combination of the characters:

char	segment	size
t	text	2Mb
d	data	200Kb
w	workspace	configurable

The parameter **all** can also be used for *opt* to indicate locking all the segments. Note the approximate size of physical RAM consumed for each segment locked. Note also that each of the *rmb* daemons attempts to lock itself into memory. Program locking requires either superuser(root) capabilities, or that the user be a member of the privgrp MLOCK (see *setprivgrp*(1M)).

-n Specifies that the global configuration file **/usr/lib/rmb/rmbrc** is not to be read.

-r *num*
This option specifies for *rmb* to attempt to run at the real-time priority specified by *num*. Valid values for *num* are 0 to 127, where 0 is the highest priority. Note that each of the *rmb* daemons will also attempt to run at this priority. Real-time priority requires either superuser capabilities, or that the user be a member of the privgrp RTPIO (see *setprivgrp*(1M)).

-t Enables terminal keymappings as described in *Using HP BASIC/UX 6.2*.

-w *num*[KM]
Specifies the *rmb* workspace to be *num* bytes in size. The optional K suffix can be added to represent Kilobytes, or M for Megabytes. The default workspace size is 1Mb.

-N In **X Windows**, specifies that no window is to be created when running in background. This is only useful when running *rmb* with input redirected and output **not** redirected.

The following standard **X Windows** command line options along with some additional options are supported by *rmb* when running in X.

-bd *color*
This option specifies the color to use for the border of the window. The default is the foreground color used in the window. The corresponding resource name is **borderColor** (class **BorderColor**).

-bg *color*
This option specifies the color to use for the background of the window. The default is "black." The corresponding resource name is **background** (class **Background**).

-buf *number*
This option specifies the number of lines to save in the alpha buffer. The default is 52. The corresponding resource name is **bufferSize** (class **BufferSize**).

-bw *number*
This option specifies the width in pixels of the border surrounding the window. The corresponding resource name is **borderWidth** (class **BorderWidth**).

-cm *mode*
This option specifies the type of alpha cursor to be used in the window. The valid modes are "Underscore" (default) and "Block". The corresponding resource name is **cursorMode** (class **CursorMode**).

-display *display*
This option specifies that *display* is to be used as the X windows display server for the *rmb* window.

-fg *color*
This option specifies the color to use for displaying text. The default is "white." The corresponding resource name is **foreground** (class **Foreground**).

-fn *font*
This option specifies a font to be used when displaying alpha text. The corresponding resource name is **font** (class **Font**).

-geometry *geometry*
This option specifies the preferred size and position of the *rmb* window. The corresponding resource name is **geometry** (class **Geometry**).

-iconic
This option indicates that *rmb* should be placed on the display in icon form. The corresponding resource name is **iconic** (class **Iconic**).

+iconic
This option indicates that *rmb* should not be placed on the display in icon form. The corresponding resource name is **iconic** (class **Iconic**).

-retain
This option indicates that the *rmb* window should be retained. The corresponding resource name is **retainedWindow** (class **RetainedWindow**).

+retain
This option indicates that the *rmb* window should be not be retained. The corresponding resource name is **retainedWindow** (class **RetainedWindow**).

-title This option specifies a window title for the *rmb* window. This string may be used by the window manager when displaying the window or icon. The corresponding resource name is **windowName** (class **WindowName**).

-x *display*
This option specifies that *display* is to be used as the X windows display server for the *rmb* window. This option is provided for compatibility with previous versions of *rmb*. It may not be supported in future releases as the **-display** option accomplishes the same task.

-xrm *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

AUTHOR

Rmb was developed by HP

FILES

/dev/rmb/*	location of device files used by rmb
/usr/bin/rmb	rmb executable
/usr/bin/rmbbuildc	program for building CSUBs
/usr/include/csubdecl.h	include file for compiling CSUBs
/usr/lib/librmb.a	rmb library for linking CSUBs
/usr/lib/rmb/.lock/lock*	resource information lockfile
/usr/lib/rmb/demos/*	demonstration programs and manual examples
/usr/lib/rmb/fonts/*	default bitmapped character fonts
/usr/lib/rmb/ipcclean	ipcclean program
/usr/lib/rmb/newconfig/*	example configuration files
/usr/lib/rmb/rmbbootinfo	rmb configuration information file
/usr/lib/rmb/rmbclean	rmbclean script
/usr/lib/rmb/rmbconfig	rmbconfig program
/usr/lib/rmb/rmbdfile	kernel configuration file scanner
/usr/lib/rmb/rmbhil	HIL interface daemon
/usr/lib/rmb/rmbkill	program to kill EXECUTE processes after RESET
/usr/lib/rmb/rmbkbd	keyboard daemon
/usr/lib/rmb/rmbrc	global configuration file
/usr/lib/rmb/rmbtmr	timer daemon
/usr/lib/rmb/rmbxfr	TRANSFER statement daemon
/usr/lib/rmb/utils/*	rmb utilities and CSUBs
$HOME/.rmbrc	the local user configuration file

SEE ALSO

Using HP BASIC/UX 6.2, HP BASIC 6.2 Language Reference.

NAME

rmbbuildc - generate a CSUB library

SYNOPSIS

rmbbuildc [*memory_size*]

Remarks:

This command requires installation of optional BASIC/UX software (not included with the standard HP-UX operating system) before it can be used.

The keywords **disk** and **disc** are parsed as equivalents by *rmb* and related commands and programs.

DESCRIPTION

Rmbbuildc is a program which generates a CSUB library in a BASIC PROG file, which can then be loaded by *rmb*.

Rmbbuildc interactively prompts the user with the necessary information about the interface of each CSUB. Each prompt is explained in detail in the manual *Developing CSUBs for HP BASIC/UX 6.2.* The program takes an optional parameter which specifies the number of bytes to allocate at run-time. If this parameter is omitted, the program will allocate 1,000,000 bytes by default.

AUTHOR

Rmbbuildc was developed by HP.

SEE ALSO

Developing CSUBs for HP BASIC/UX 6.2.

NAME

rmbclean, ipcclean - clean up RMB lock files and IPC resources

SYNOPSIS

rmbclean

Remarks:

This command requires installation of optional BASIC/UX software (not included with the standard HP-UX operating system) before it can be used.

The keywords **disk** and **disc** are parsed as equivalents by *rmb* and related commands and programs.

DESCRIPTION

Rmbclean is a script to find orphaned *rmb* lock files and clean them up. This involves removing any IPC resources remaining for the process that created the lockfile, and removing the lockfile itself. *Rmbclean* does not touch lockfiles for which an *rmb* program is still running. *Rmbclean* calls the program *ipcclean* to do the actual reclamation of IPC resources and removal of the lockfile.

The lockfile is removed if the IPC resource reclamation is successful. It is also removed if it is not readable or if it is an out-of-date version. The only time it should not be removed is when resources are not reclaimed due to permissions violations.

The *rmb* installation process should add an invocation of *rmbclean* to the **/etc/rc** script. This is recommended as a means of cleaning up all lockfiles each time HP-UX is booted. *Rmbclean* can also be invoked manually from *rmb* with the **-i** option to *rmb*. *Rmb* also automatically invokes *rmbclean* to attempt to reclaim IPC resources when no more are available.

AUTHOR

Rmbclean was developed by HP

FILES

/usr/lib/rmb/.lock/lock*	rmb lock files
/usr/lib/rmb/ipcclean	a program to scan a lockfile for IPC resources
/usr/lib/rmb/rmbclean	the rmbclean script

NAME

rmbconfig - generate and view RMB configuration information

SYNOPSIS

rmbconfig [**-bfs**]

Remarks:

This command requires installation of optional BASIC/UX software (not included with the standard HP-UX operating system) before it can be used.

The keywords **disk** and **disc** are parsed as equivalents by *rmb* and related commands and programs.

DESCRIPTION

Rmbconfig is a program to extract system information from HP-UX and store this information in a file accessible to *rmb*. *Rmbconfig* also automatically generates device files for all cards found in the system that are supported by *rmb*.

Since *rmbconfig* creates device files and reads **/dev/kmem**, it usually requires root capabilities. It is recommended that *rmbconfig* be installed with *setuid root* capabilities. This allows *rmb* to call it for updated system information.

The *rmb* installation process adds an invocation of *rmbconfig* to the **/etc/rc** script. This way, system configuration information is updated each time HP-UX is booted. *Rmbconfig* can also be invoked manually from *rmb* with the **-k** option to *rmb*. *Rmb* also automatically invokes *rmbconfig* if the file **/usr/lib/rmb/rmbbootinfo** is not found or is the wrong revision.

Options

The command line options are:

-b This option specifies that device files are not to be created.

-f This option forces scanning of the computer back-plane. Normally *rmbconfig* does not scan the backplane if an HP98577 card is found.

-s Specifies that the extracted system information is to be displayed.

Displayed information

There are several types of information that *rmbconfig* extracts from the system. In order of display they are:

HP-UX Information about HP-UX is determined via the *uname* system call. This includes information on the node name, computer number, and operating system revision.

bootrom The bootrom is scanned through a temporary iomap device file to determine the bootrom revision.

hardware Hardware capabilities are determined from the kernel. These include processor, coprocessors and other assists. Recognized hardware includes:

Device	Description
MC68020	Model 320, 330, and 350 processors
MC68030	Model 332, 340, 360, 362, 370 and 375 processors
MC68040	Model 380, 382 and series 400 processors
HP98248A	Floating-point accelerator
MC68881	Floating-point coprocessor
MC68882	Floating-point coprocessor
HP98635	Floating-point card
HP98286A	DOS coprocessor
HP98620	DMA card

console The system console address and type are read from **/dev/kmem**. A device file for the console is created (or linked) as **/dev/rmb/crt**.

ram Information about the system RAM. This includes 3 values:

physical	amount of physical RAM
available	amount available for processes
free	currently unused RAM

swap The location and sizes of all swap devices configured into the kernel are determined. This includes those swap devices which have **not** been *swapon*'ed. This information is extracted from **/dev/kmem**.

disks The location of all mounted disks is read from **/dev/kmem**.

drivers The kernel is scanned for all recognized drivers. This includes the SYS V IPC code, disk drivers, interface drivers and card drivers. Recognized drivers (including permanent drivers) are:

Driver	Abbr.	Description
amigo	AMIGO	Amigo-protocol disk driver
ciper	CIPER	CIPER-protocol printer driver
console	CONS	Console driver
cs80	CS80	CS/80 disk driver
diskless	DSKLESS	diskless server driver
dos	DOS	HP98286 MS-DOS card driver
ether	ETHER	Ethernet driver
gpio	GPIO	HP98622 GPIO driver
graphics	-none-	Graphics driver
hil	HIL	HP-HIL loop driver
hpib	HPIB	HP-IB interface driver
ieee802	802	IEEE-802 (LAN) driver
iomap	IOMAP	Iomap file driver
mem	MEM	Mem/kmem driver
messages	-none-	System V messages
minifloppy	MF	Internal minifloppy (obsolete)
nimitz	-none-	HP9836-style-keyboard driver
nfs	-none-	Network File System driver
plotter	PLOT	Old plotter driver
printer	PR	Line printer driver
ptym	PTYM	Pty master driver
ptys	PTYS	Pty slave driver
ramdisk	RAMDISC	RAM disk driver
rdu	RDU	RDU driver
r8042	R8042	8042 keyboard driver
rfa	-none-	remote file access driver
rje	RJE	HP98641 RJE driver
scsi	SCSI	SCSI disk interface driver
semaphores	-none-	System V semaphores
shared memory	-none-	System V shared memory
sna	SNA	SNA link driver
srm	SRM	HP98629 SRM driver
stp	STP	streaming tape driver
stealth	STEALTH	HP98577 VME backplane driver
swap	SWAP	disk swap space driver
tp	TP	magnetic tape driver
tty	TTYsy	virtual (/dev/tty) driver
ttyxx	TTYxx	physical tty driver
vme	VME	HP98646 VME extender driver
98624	-none-	hpib card driver
98625	-none-	disk interface card driver
98626	-none-	RS-232 card driver
98628	-none-	datacomm card river
98642	-none-	RS-232 mux driver

variables Various configurable kernel variables of interest are extracted from **/dev/kmem**. These include:

Variable	Description
maxdsiz	Maximum program data size
maxssiz	Maximum program stack size
maxtsiz	Maximum program text size
maxuprc	Maximum number of user processes
msgtql	System message queue limit
nproc	System process limit
shmmin	Minimum shared memory segment size
shmmax	Maximum shared memory segment size

interfaces I/O interfaces are determined from two sources in **/dev/kmem**. First the kernel's internal device table is searched to find interfaces supported by the kernel. Then the physical I/O space of the machine is searched for interfaces not recognized by the kernel (unless a *STEALTH* VME card is found). Interfaces recognized by *rmbconfig* and the device files created (if any) include:

Interface	Dev_file	Description
HP50962	srm	Serial SRM link card
HP98253	eprom	EPROM programmer card
HP98259	bubble	Bubble memory card
HP98265	-none-	SCSI disk interface card
HP98287	gbox	Hi-res display controller interface
HP98577	vme	VME backplane adapter
HP98622	gpio	GPIO card
HP98623	bcd	BCD interface card
HP98624	hpib	HP-IB card
HP98625	-none-	HP-IB disk interface card
HP98626	serial	RS-232 card
HP98627	moon	RGB interface card
HP98628	serial	Datacomm interface card
HP98629	srm	SRM interface card
HP98633	double	Multi-programmer interface
HP98640	adc	ADC card
HP98641	rje	RJE card
HP98642	mux	RS-232 mux card
HP98643	-none-	LAN interface card
HP98644	serial	low-cost RS-232 card
HP98646	vme	VME extender card
HP98649	sna	SDLC card for IBM SNA
HP98691	pdi	programmable datacomm card
HP98695	3270.	3270 emulator
proto-sngl	iomap	prototype standard size card
proto-dbl	double	prototype double size card
proto-quad	quad	prototype quad size card

Device files are created as **/dev/rmb/***devfileSC* where *devfile* is the name specified in the above table, and *SC* is the select code at which the card is mapped. Interfaces with -none- indicated for their *devfile* are either not supported for direct access by *rmb,* or do not require device files for access (HP98248, HP98635). If an appropriate device file is found in **/dev**, a link is made to it in **/dev/rmb**. Otherwise a new device file is created. If the interface is supported by HP-UX, the device file is created with permissions 000 for security, and the system administrator is required to change them to allow access to the card.

The display of interfaces includes the card number (name), select code, card interrupt level, appropriate device file name (for *rmb*), and abbreviations for all drivers (from the driver table above) for which a device file was found for this select code. Device files are searched only in directories **/dev** and **/dev/rmb**.

HIL Files **/dev/hil*** are checked to determine the types of devices on the HP-HIL loop. Devices locked by running programs (such as window managers) cannot be determined and are reported as busy.

AUTHOR

Rmbconfig was developed by HP

FILES

/dev	searched for relevant device files
/dev/hil*	device files scanned for HIL info
/dev/kmem	device file used to access kernel info
/dev/rmb/*	location of created/linked device files
/tmp/rmbconf*	temp iomap file for accessing the bootrom
/usr/lib/rmb/rmbbootinfo	rmb configuration information file
/usr/lib/rmb/rmbconfig	the rmbconfig executable

NAME

rmbdfile - kernel configuration file scanner for rmb (HP BASIC/UX)

SYNOPSIS

rmbdfile [**-v**] [*dfile*]

Remarks:

This command requires installation of optional BASIC/UX software (not included with the standard HP-UX operating system) before it can be used.

The keywords **disk** and **disc** are parsed as equivalents by *rmb* and related commands and programs.

DESCRIPTION

Rmbdfile is a program to scan a kernel configuration file and modify it such that it will generate a kernel capable of running *rmb*. If *dfile* is specified, it is scanned. Otherwise the input is taken from **stdin**. The modified file is output to **stdout**.

Options

-v Specifies that *rmbdfile* scan a kernel configuration file and indicate whether or not the file is adequate for generating a kernel that can support running *rmb*. The modified file is not output.

Operation

rmbdfile examines two types of kernel parameters:

Drivers *Rmb* requires that the following drivers be configured into the kernel. If not present in the input file they are added by *rmbdfile*. Other drivers are passed through to the output.

Driver Name	Description
98624	HP 98624 HP-IB card driver
gpio	Device I/O Library GPIO driver
hpib	Device I/O Library HP-IB driver
mesg	message queues
sema	semaphores
shmem	shared memory

Variables *Rmb* requires a minimum or a maximum value for certain kernel variables. Some of these variables have an adequate default value; others are added by *rmbdfile* if not explicitly specified. The value of all specified variables are checked against the minimum (or maximum) value and if less (or greater), the variable is assigned this value. The table below indicates the scanned variables, whether the default value is adequate, whether a minimum or maximum is tested, and the boundary value. Variables not in the table are passed through unchanged.

Variable	Default OK	Requirement	Value
maxdsiz	yes	min	1310720 (1.25 Mbytes)
maxssiz	yes	min	524288 (.5 Mbytes)
maxtsiz	yes	min	2097152 (2 Mbytes)
maxuprc	no	min	64
msgtql	no	min	256
ndilbuffers	yes	min	5
nproc	no	min	128
shmmax	yes	min	1048576 (1 Mbytes)
shmmin	yes	max	65536 (64 Kbytes)

Any other kernel parameters are passed through to the output.

EXIT CODES

0 Normal execution

1 Input file not adequate (for -v option)

AUTHOR

Rmbdfile was developed by HP

NAME

rmbkernel - kernel building script for rmb (HP BASIC/UX)

SYNOPSIS

rmbkernel [*dfile*]

Remarks:

This command requires installation of optional BASIC/UX software (not included with the standard HP-UX operating system) before it can be used.

The keywords **disk** and **disc** are parsed as equivalents by *rmb* and related commands and programs.

DESCRIPTION

Rmbkernel is a script to generate a new kernel consistent with the requirements of *rmb*. If an input *dfile* is specified, then that file is used as the starting point for generating the new kernel. Otherwise, the file **/etc/conf/dfile** will be used. If **/etc/conf/dfile** does not exist, you should use the System Administration Manager (*sam*, see sam(1M)) to create a dfile. The dfile that is used to build the new kernel will be saved as **/etc/conf/dfile.rmb**. If an old version of this file exists it is saved as **/etc/conf/dfile.rmb.old**. The new dfile **/etc/conf/dfile.rmb** is also copied back to *dfile* if one is specified.

The kernel that is built by this script is left in **/etc/conf/hp-ux**. Instructions are then given for installing the new kernel and rebooting.

Root capabilities are required to run *rmbkernel*.

DIAGNOSTICS

Rmbkernel prints messages as it runs to indicate what it is doing. If any failures occur in the script an error message is printed and the script aborts.

AUTHOR

Rmbdfile was developed by HP

FILES

/etc/config	kernel configuration program
/etc/conf/dfile	used if no dfile specified
/etc/conf/dfile.rmb	dfile corresponding to the new kernel
/tmp/dfile*	temporary working copies of the dfiles
/usr/bin/sam	system administration manager
/usr/lib/rmb/rmbkernel	the rmbkernel script

SEE ALSO

sam(1M).

NAME

rmbkill - kill a process and all its descendents

SYNOPSIS

rmbkill [**-q**] [*-signo*] *process_number*

Remarks:

This command requires installation of optional BASIC/UX software (not included with the standard HP-UX operating system) before it can be used.

The keywords **disk** and **disc** are parsed as equivalents by *rmb* and related commands and programs.

DESCRIPTION

Rmbkill is a program that kills a process and all its descendents without them having to be in their own process group. This program is called by *rmb* to kill all EXECUTE processes during a RESET.

Rmbkill first reads the process list from the kernel through the device file **/dev/kmem**. It then searches the process list for the process specified by *process_number*. If *process_number* is not found, the program terminates with an error. If found, the process and all its descendents are sent the signal SIGHUP. The process number and status of each process is printed with indentation showing the process tree structure.

Rmbkill exits if process numbers 0, 1 or 2 are specified. The program runs with setuid root capabilities in order to read **/dev/kmem**, but changes identity to that of the real user before attempting to kill any processes so that user permissions are not violated.

Options

-q	Specifies quiet mode. The process tree structure with status is not printed.
-signo	Indicates that signal signo should be sent instead of SIGHUP. Values in the range 1 through 32 (inclusive) are valid. Some signals are not implemented on all systems and produce an error message if incorrect.

AUTHOR

Rmbkill was developed by HP

FILES

/dev/kmem	device file for accessing kernel data structures
/usr/lib/rmb/rmbkill	the rmbkill program

B

Running BASIC/UX as a Background Process

You may wish to run BASIC/UX as a background process to free up your invoking window when running in the X Window System (X11). This is accomplished by typing an "&" after the **rmb** command and options. For example, if you type:

```
rmb -w3M -fg yellow &
```

from an **hpterm** window in X11, **rmb** will start up with a 3 MB workspace size and a foreground color of yellow, and the **hpterm** window will be free for other HP-UX commands (you will see a shell prompt).

Using the X11 Environment

Running **rmb** as a background process has some side effects that you should be aware of. In the X11 environment, the BASIC/UX window will be created the same as if BASIC/UX were running in the foreground. Input and output will also behave similarly. The only difference is that the window from which you started **rmb** will return with a shell prompt so that you can continue with other HP-UX commands.

There may be some cases where you don't want the BASIC/UX window to appear when running **rmb** in the background in the X11 environment. The **-N** option (no window) prevents the BASIC/UX window from being created and **rmb** runs as if it was being run in the background on the console.

Using a Console or Terminal

Running rmb as a background process from either a console or a terminal differs from running it in the foreground. There is no BASIC display and no live keyboard. The following command:

```
rmb &
```

causes rmb to exit immediately. While this may seem useless, there is another feature called "redirected I/O" which greatly enhances its usefulness. When the rmb process starts, it automatically opens three files. These files are the standard input, the standard output, and the standard error output. Normally these are all connected to your current terminal device, but can be redirected to files or pipes. If you redirect standard input for the rmb process, BASIC/UX interprets the input as KBD line commands. This allows you to create a file of BASIC commands (a BASIC script) and use it as input to an rmb process. If you redirect standard output for the rmb process, BASIC/UX sends all output which is normally displayed in the Output Area to the standard output file. BASIC/UX output which is normally sent to the Display Line and the Message Results Line is sent to the standard error file.

For example, create a file called rmb.input with the following contents:

```
PRINT DATE$(TIMEDATE)&" "&TIME$(TIMEDATE)
PRINT "LOADing MYPROG"
LOAD "MYPROG"
PRINT "SAVEing as MYPROG.asc"
SAVE "MYPROG.asc"
PRINT "End of input"
```

This script tells BASIC/UX to print the date and time, then convert a PROG file (MYPROG) to an ASCII file (MYPROG.asc). Now type:

```
rmb < rmb.input > rmb.output &
```

The rmb process exits when it reaches the end of the standard input. When it finishes, rmb.output looks like this:

```
28 Aug 1989 23:03:08
LOADing MYPROG
SAVEing as MYPROG.asc
End of input
```

This functionality is useful if you want to run BASIC/UX from cron. Note that what is really causing the rmb process to exit when you specify rmb & is that the standard input file is being assigned to the null file /dev/null. A read from the null file returns 0 bytes, thus the rmb process thinks it has reached the end of the input and exits.

Specifying an autostart file on the BASIC/UX command line causes the rmb process to run the program, and then process any input. If there is no input, rmb will exit. For example,

```
rmb MY_AUTOST
```

will start BASIC/UX (in the foreground), run MY_AUTOST, and then wait for KBD input. If you run rmb as background process,

```
rmb MY_AUTOST &
```

then BASIC/UX will boot up and run MY_AUTOST. BASIC/UX exits immediately after running MY_AUTOST because it reads from the null file for input.

There are some additional considerations when rmb is running as a background process. If the autostart program PAUSEs for any reason (a PAUSE statement or a runtime error) or requests input from the keyboard (via INPUT, LINPUT, or ENTER KBD), the rmb process will try to read from standard input. If you have not provided input through redirection, the rmb process will exit. If a program requests KBD input (e.g., with an INPUT statement) and you know what the program is requesting and how you want to answer the request, you can redirect input to the rmb process. For example, suppose MY_AUTOST looks like this:

```
10   PRINT "Welcome to the System"
20   PRINT
30   INPUT "Press RETURN to start the MONITOR program".,C$
40   LOAD "MONITOR",1
50   END
```

If you know that you are always asked to:

```
Press RETURN to start the MONITOR program
```

you can create an input file (rmb.input) that contains just a carriage return. To run rmb as a background process, run the autostart program

(MY_AUTOST), and proceed to load and run MONITOR, use the following command:

```
rmb MY_AUTOST < rmb.input &
```

`rmb` will read the carriage return from `rmb.input` and continue to load MONITOR. If you did not redirect standard input, BASIC/UX would attempt to read from the `null` file to satisfy the INPUT request and exit upon finding no data. The `rmb` process would never get to line 40 of MY_AUTOST. Note that if you are running on a console or terminal, the PRINT statements in MY_AUTOST will be sent to the standard output file.

Using the EXECUTE Command

If you are running `rmb` as a background process or have redirected standard input and attempt to execute a shell or a command that requires input, the shell used to control the EXECUTE command will inherit its input from the `rmb`standard input stream.

Similarly, if you perform an EXECUTE command that produces output, the output will be sent to the `rmb` standard output stream. For example:

```
EXECUTE "ls"
```

normally produces a listing of all the files in the current directory and sends it to your terminal device. However, if you have redirected standard output for the `rmb` process, then the output from the EXECUTE command will be sent to the same place as the `rmb` standard output. You can specify that the standard output of the EXECUTE command be sent somewhere other than the `rmb` standard output stream by redirecting the EXECUTE command's output itself. For example,

```
EXECUTE "ls > rmb.ls"
```

would save the output of the `ls` command in `rmb.ls`.

Additional Information on Redirecting Standard Input

As mentioned above, standard input is interpreted as commands you type into the BASIC/UX KBD line. This allows you to automate procedures that you have to repeat many times by writing BASIC scripts. For example, a script to convert a PROG file into ASCII format would look like the following:

```
LOAD "PROG_FILE"
SAVE "ASCII_FILE"
```

This example can be expanded to convert all the files in a BASIC/UX application so that they can be used under source code control.

If only standard input is redirected and not standard output, the BASIC/UX display will come up as usual in all environments (console, X11, and terminal), but the keyboard will be dead. This scenario may be useful for demonstrations. However, you should be aware that if your demonstration programs contain statements that expect a live keyboard (e.g., ON KBD and ON KEY), you may need to recode your program so that it is not dependent on those statements. When the end of the input stream is reached, BASIC/UX will exit.

When standard input is redirected, BASIC/UX defaults to CAPS LOCK mode disabled. This allows literal strings to be interpreted correctly by BASIC/UX. For example, suppose the following line was part of the redirected input:

```
20  SYSTEM$("MSI")
```

If CAPS LOCK mode were enabled, BASIC/UX would interpret this as:

```
20  system$("msi")
```

The `system$` would be recognized as a BASIC/UX keyword (BASIC/UX automatically converts keywords to upper case), but the `msi` literal would cause an error.

Non-ASCII Character Sequences

The standard input stream can contain the 2-byte non-ASCII character sequences as listed in the *HP BASIC 6.2 Interface Reference*. You can send non-ASCII character sequences to BASIC/UX by using the escape character in the `vi` editor (i.e., CTRL-V Esc) which will appear as: `^[`. For example,

```
^[H^[>^[>^[>^[>^[>
```

will position the cursor at the beginning of the line (`^[H`) and then move it five spaces to the right (`^[>^[>^[>^[>^[>`). Also, if you have a program with literal strings containing the non-ASCII keycode sequences which was SAVEd by BASIC/UX , the keycode sequences will be interpreted correctly. These keycode sequences can be identified with an inverse-video K (character code 255) followed by an ASCII character.

Consequences of Serial Input

Redirecting standard input disables the live keyboard. The input stream is read serially. Thus, ON KBD, ON KEY, and similar events can never occur. If you have existing programs which depend on the asynchronous nature of these statements, they will need to be changed. However, this allows the standard input stream to contain data for INPUT, LINPUT, and ENTER KBD statements. Standard input is not read while **rmb** is in the Running state, but as soon as an `Input?` state occurs, **rmb** will read from the input stream. Note that as soon as a program pauses, **rmb** starts reading from standard input. Thus, if there is a PAUSE somewhere before an INPUT statement in the program, data will be lost. If you know that there is a PAUSE in the program, you can cause the program to continue by including the 2-byte key sequence for the `Continue` softkey (`^[C`) as part of your input file.

Debugging BASIC Scripts

If there is a syntax error in the input stream, a BASIC error message is generated. If the KBD buffer has not been cleared (as is the case with syntax errors), the next line of input that **rmb** reads will also cause an error. Thus, it is possible for one error in the BASIC script to cause a cascade of errors. Turning on PRINTALL mode (CONTROL KBD,1;1) at the beginning of your BASIC script will aid in debugging any problems you may encounter when redirecting input.

Additional Information on Redirecting Standard Output

If standard output is redirected, no graphics is permitted to the display (graphics to HPGL devices is still allowed). In X11, no windows are created and all window commands generate an error. The only output from `rmb` that appears in the standard output is output which would have appeared in the Output Area of the display. One exception to this is that the EDIT screen is not output. This does not mean that you cannot edit a program (via OUTPUT KBD or the non-ASCII character sequences mentioned above), only that the EDIT screen will not appear in the standard output.

If you run `rmb` in the foreground and redirect standard output but not standard input, the live keyboard will be active. However, since output is redirected, the soft keys will not appear and PRINT or OUTPUT,CRT output will be sent to standard out. If you have not redirected standard error, DISP, INPUT, and LINPUT strings and BASIC errors will appear on your terminal device.

If you run `rmb` in the background and redirect standard output but not standard input, the keyboard will not be active. The `rmb` process will attempt to get input from the `null` file and exit.

Information on Redirecting Standard Error

Any output from the `rmb` process which would have appeared in the Display Line or Message Results Line will be sent to the standard error file. If you turn on PRINTALL mode in your BASIC script, output to the Display Line and Message Results Line will also appear in the standard output file. The following is an example of redirecting standard input, standard output, and standard error:

```
   rmb < rmb.input > rmb.output 2> rmb.error &
```

New STATUS Register for Pseudo Select Code 32

A new STATUS register has been defined so that you can determine the foreground/background and redirected I/O status of the `rmb` process. This information can help you decide programmatically whether or not to perform certain functions, such as an EXECUTE.

STATUS Register 5

Background and standard I/O status

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	0	0	0	Back-ground Process	`stderr` Redirected	`stdout` Redirected	`stdin` Redirected
Value=128	Value=64	Value=32	Value=16	Value=8	Value=4	Value=2	Value=1

Foreground and Background Definition

Foreground Definition

Operation	Console Behavior	Terminal Behavior	X Windows Behavior
`rmb`	Operates as originally defined	Operates as originally defined	Operates as originally defined
`rmb` MYPROG	Operates as originally defined	Operates as originally defined	Operates as originally defined
`rmb` < MYKBD	Start `rmb` No active keyboard Use MYKBD as keyboard input then EXIT	Start `rmb` No active keyboard Use MYKBD as keyboard input then EXIT	Start `rmb` No active keyboard Use MYKBD as keyboard input then EXIT Retains possession of EXECUTE window
`rmb` MYPROG < MYKBD	Start `rmb` No active keyboard Run MYPROG Use MYKBD for keyboard input then EXIT	Start `rmb` No active keyboard Run MYPROG Use MYKBD for keyboard input then EXIT	Start `rmb` No active keyboard Run MYPROG Use MYKBD as keyboard input then EXIT Retains possession of EXECUTE window

Foreground Definition (continued)

Operation	Console Behavior	Terminal Behavior	X Windows Behavior
rmb &> OUTFILE	Start rmb No visible display No graphics Live keyboard active Printed output goes to OUTFILE EXIT with QUIT	Start rmb No visible display No graphics Live keyboard active Printed output goes to OUTFILE EXIT with QUIT	Start rmb No visible display No graphics No window commands Live keyboard active Printed output goes to OUTFILE EXIT with QUIT
rmb MYPROG &> OUTFILE	Start rmb No visible display No graphics Run MYPROG Live keyboard active Printed output goes to OUTFILE EXIT with QUIT	Start rmb No visible display No graphics Run MYPROG Live keyboard active Printed output goes to OUTFILE EXIT with QUIT	Start rmb No visible display No graphics No window commands Run MYPROG Live keyboard active Printed output goes to OUTFILE EXIT with QUIT

Foreground Definition (continued)

Operation	Console Behavior	Terminal Behavior	X Windows Behavior
`rmb` < MYKBD > OUTFILE	Start `rmb` No visible display No graphics No active keyboard Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD	Start `rmb` No visible display No graphics No active keyboard Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD	Start `rmb` No visible display No graphics No active keyboard Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD
`rmb` MYPROG < MYKBD > OUTFILE	Start `rmb` No visible display No graphics No active keyboard Run MYPROG Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD	Start `rmb` No visible display No graphics No active keyboard Run MYPROG Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD	Start `rmb` No visible display No graphics No window commands No active keyboard Run MYPROG Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD

Background Definition

Operation	Console Behavior	Terminal Behavior	X Windows Behavior
`rmb &`	EXIT	EXIT	Create full function BASIC/UX window and operate normally
`rmb MYPROG &`	Start **rmb** No visible display No graphics No active keyboard Run MYPROG Printed output goes to **stdout** then EXIT	Start **rmb** No visible display No graphics No active keyboard Run MYPROG Printed output goes to **stdout** then EXIT	Create full function BASIC/UX window RUN MYPROG then operate normally
`rmb` < MYKBD &	Start **rmb** No visible display No graphics No active keyboard Use MYKBD as keyboard input Printed output goes to **stdout** EXIT on end of MYKBD	Start **rmb** No visible display No graphics No active keyboard Use MYKBD as keyboard input Printed output goes to **stdout** EXIT on end of MYKBD	Create full function BASIC/UX window Use MYKBD as keyboard input then EXIT

Background Definition (continued)

Operation	Console Behavior	Terminal Behavior	X Windows Behavior
`rmb` MYPROG < MYKBD &	Start `rmb` No visible display No graphics No active keyboard Run MYPROG Use MYKBD as keyboard input Printed output goes to `stdout` EXIT on end of MYKBD	Start `rmb` No visible display No graphics No active keyboard Run MYPROG Use MYKBD as keyboard input Printed output goes to `stdout` EXIT on end of MYKBD	Create full function BASIC/UX window Run MYPROG Use MYKBD as keyboard input then EXIT
`rmb` > OUTFILE &	EXIT	EXIT	EXIT
`rmb` MYPROG > OUTFILE &	Start `rmb` No visible display No graphics No active keyboard Run MYPROG Printed output goes to file OUTFILE then EXIT	Start `rmb` No visible display No graphics No active keyboard Run MYPROG Printed output goes to file OUTFILE then EXIT	Start `rmb` No visible display No graphics No active keyboard Run MYPROG Printed output goes to file OUTFILE then EXIT

Background Definition (continued)

Operation	Console Behavior	Terminal Behavior	X Windows Behavior
rmb < MYKBD > OUTFILE &	Start **rmb** No visible display No graphics No active keyboard Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD	Start **rmb** No visible display No graphics No active keyboard Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD	Start **rmb** No visible display No graphics No active keyboard Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD
rmb MYPROG < MYKBD > OUTFILE &	Start **rmb** No visible display No graphics No active keyboard Run MYPROG Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD	Start **rmb** No visible display No graphics No active keyboard Run MYPROG Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD	Start **rmb** No visible display No graphics No active keyboard Run MYPROG Use MYKBD as keyboard input Printed output goes to file OUTFILE EXIT on end of MYKBD

Index

HEWLETT
PACKARD

HP Part Number
E2040-90002

Printed in U.S.A. E0891

E2040-90602 Manufacturing Number