PERQ™ System Software

Reference Manual

February 1982 Software Version D.6

Textual Material Prepared by
Three Rivers Computer Corporation
Pittsburgh, Pennsylvania

TABLE OF CONTENTS

INSTALLATION GUIDE

1.1 Unpacking Instructions.

To unpack base unit (the big box), refer to Drawing #0410-A.

1. Open top of largest box.
2. Turn box upside down by carefully "rolling" it over.
3. Open inner box (the side which is now up). Note that the feet are up.
4. "Roll" box over and lift it off.

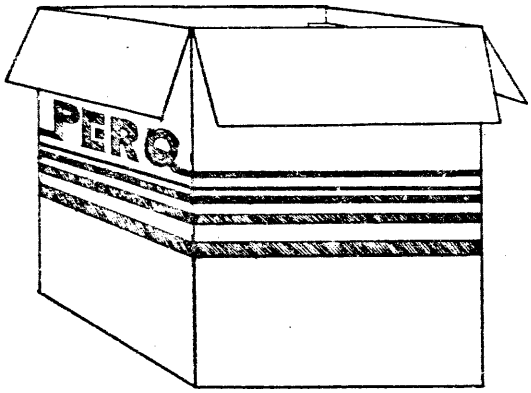
1.2 Unit Installation.

Be certain that the PERQ is NOT connected to a power source.

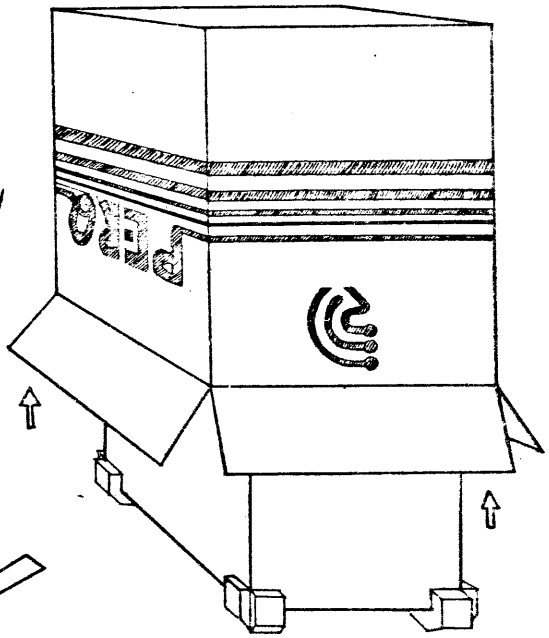
1. Position the PERQ in its desired location. Allow at least six inches between the PERQ and an obstruction such as a wall. The clearance is required to provide access to the screws securing the back or side covers.
2. Lower the four levelers 3/4 inch to permit airflow beneath the machine. Adjust the leverlers to compensate for uneven floors and make the PERQ stable.
3. Using a #2 Phillips screwdriver, remove the four (4) front cover screws. See drawing #0133-A. Remove the front cover.
4. Similarly, remove the rear cover. See drawing #0134-A.
5. When facing the front of the PERQ, the harddisk drive is located on the left hand side. Remove the two (2) screws securing the left side cover. Remove the left side cover. See drawing #0135-A.
6. Using the supplied Allen Wrench, remove disk shipping screw. See drawing #0137-B. Do not lose the shipping screw because it must be reinstalled before you move or ship the PERQ.
7. Remove shipping disk from floppy drive.
8. Plug in A.C. connector for disk. See Drawing #0137-B.

CAUTION - DURING THE FOLLOWING STEPS, HAZARDOUS VOLTAGES ARE PRESENT IN THE CHASSIS.

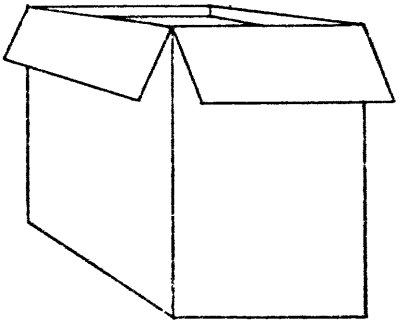
9. Plug line cord into appropriate power sources. See label on PERQ above line conncector for power requirements.



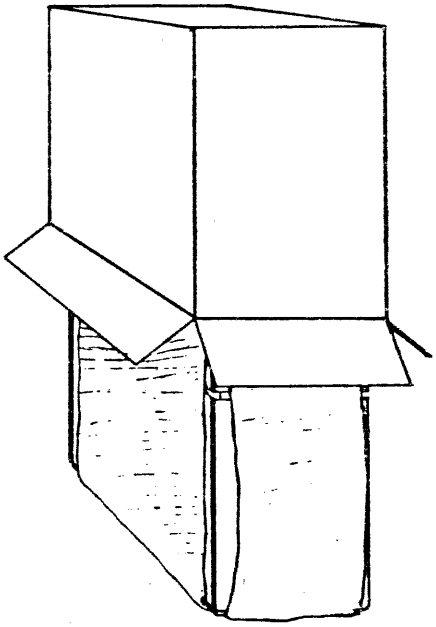
1



2



3



4

Three Rivers Computer	
PERQ UNPACKING ILLUSTRATION	PERQ
	PRQ-AD-0410-A

10. Pull front and rear "cheat" switches. Operate "On/Off" switch on front of PERQ. System will turn on. Wait 30 seconds.

11. Remove disk head locking clip ("A" Clip) by pulling in direction of arrow. See Drawing #0137-B. Do not lose this lock because it must be reinstalled before you move or ship the PERQ.

12. Operate "On/Off" switch to turn PERQ off.

13. Remove line cord from wall outlet and then from PERQ.

14. Re-install left side cover. (Note labels on cover.)

15. Re-install rear and front covers.

16. Plug in keyboard, display, and tablet (bit pad) to rear of PERQ. Connect the tablet cable between plug J7(GPIB), at the rear of the PERQ, and the back of the tablet. DO NOT USE PLUB J2 (tablet). Tighten the thumb screws on both plugs. Connect the power supply cable to the back of the tablet, then plug the power supply into the wall outlet.

17. Plug line cord into appropriate power source. See label on PERQ above line connector for power requirements.

NOTE: Shugart documentation regarding SA4000 (rigid disk) is enclosed for reference.

The Federal Communications Commission of the United States Government has published regulations which govern the allowable limits of emanation of radio frequency energy of computing devices and associated peripherals.

Those regulations are concerned with interference to radio communication, such as radio and TV.

The regulations require equipment for end use in the United States to be labeled and to be accompanied by the notice appearing in this instruction manual.

> Warning: This equipment generates, uses and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. As temporarily permitted by regulation it has not been tested for compliance with the limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

1.3 Reshipment Procedures

This section details the procedures to follow if you must move or

ship your PERQ. Sectins 1.3.1 and 1.3.2 describe the packing procedures.

1. Turn power off and disconnect all external sub-assemblies.

2. Using a #2 Phillips screwdriver, remove the four (4) front cover screws. See drawing #0133-A. Remove the front cover.

3. Similarly, remove the rear cover. See drawing #0134-A.

4. Remove the two (2) screws securing the left hand side cover (left side when facing the front of PERQ) and remove the left side cover.

5. Reconnect the monitor and keyboard, pull both "pull to cheat" switches, and power on the macine.

6. Boot the PERQ and Log in.

7. Type BYE WAIT. This moves the disk heads to the center of the disk.

8. Insert disk head locking clip ("A" clip). Push the "A" clip to install. Do NOT install the "A" clip unless the disk is spinning.

9. Turn power off.

10. Disconnect the monitor and keyboard. Disconnect the AC power plug from the hard disk (see drawing #0137-B).

11. When the disk has stopped spinning, turn the motor and align the hole for shipping screw. Insert shipping screw using Allen Wrench.

12. Replace left side cover. Replace front and rear covers.

1.3.1 Packing Procedures

1. Take the two longest pieces of white Micro-foam and lay them on the floor in the shape of a cross (+). Place the PERQ on top of the pieces and wrap the pieces over the PERQ.

2. Slide the box about the size of the PERQ over the foam wrapped machine.

3. Very carefully, turn the box over so that the open end of the box is up. Lower the four (4) levelers on the botoom of the PERQ. Tape the box shut.

4. Place the corner-shapes on the corners of the boxed PERQ.

5. Carefully, slide the large outer box over the boxed PERQ and turn the large outer box over so that the open end is up. Tape the outer box shut.

1.3.2 Monitor and Sub-assemblies Packing

1. Wrap the monitor with white Micro-foam. Find a box about the size of the monitor and slide it over the monitor. Tape the box shut.

2. Place white and green plastic corners on the monitor box. Place a larger outer box over the boxed monitor and tape shut.

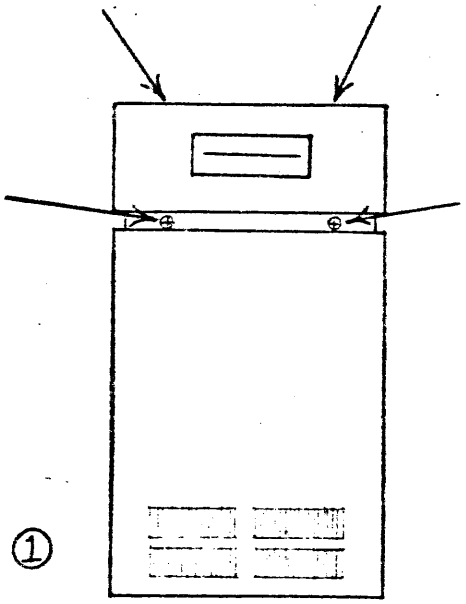
3. Find the box the size of the keyboard. Insert keyboard, close, and tape shut.

4. Insert the bit pad, bit pad magnet, and manual into the bit pad box and tape shut.

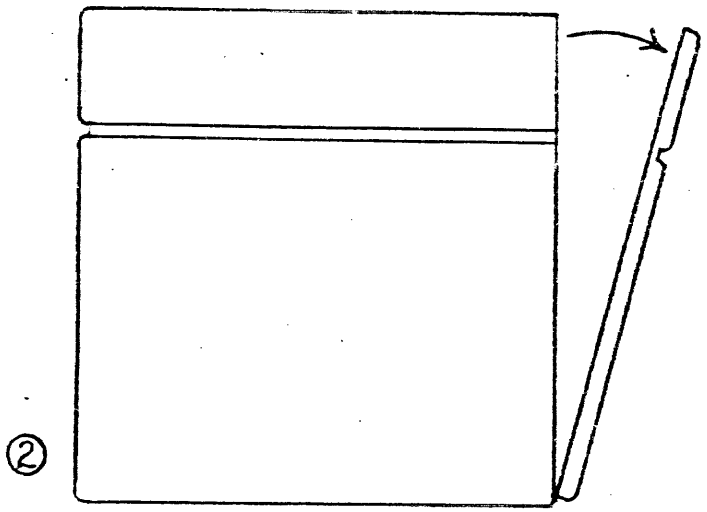
5. Box the cables and power cords and tape shut.

6. Take all boxes and place them in the larger outer box. Pack with foam to take up extra space. Close and tape.

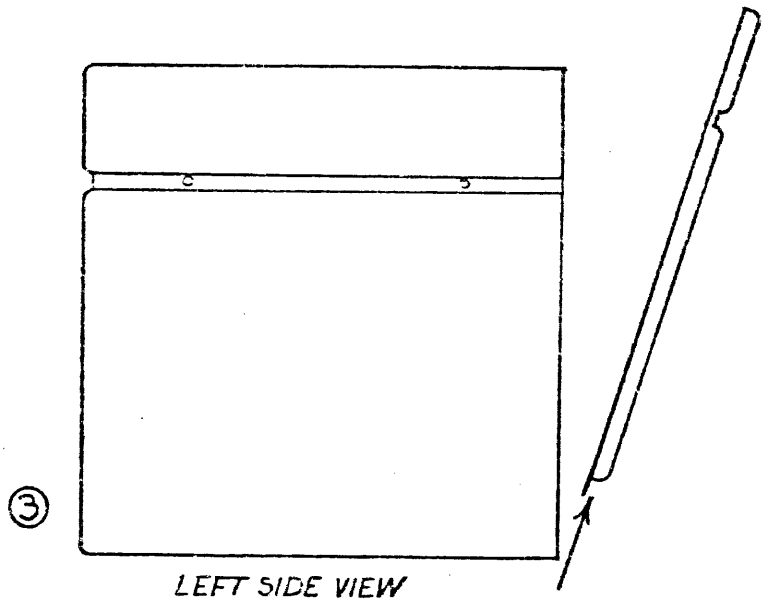
7. After packing and sealing the boxes, they should be palletized to eliminate shipping damage.



① FRONT VIEW

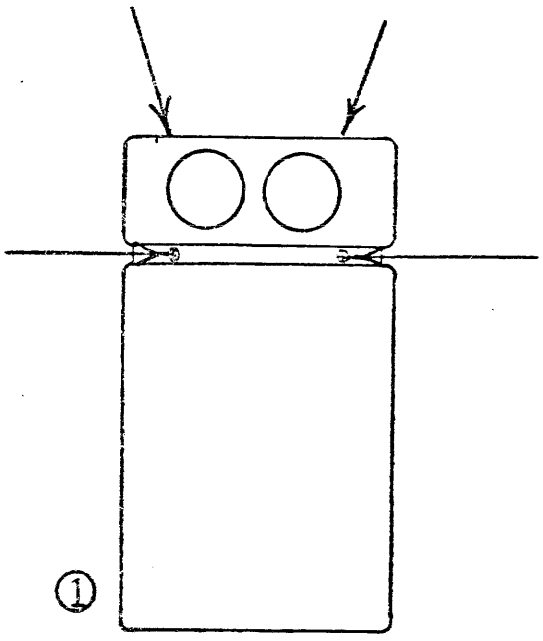


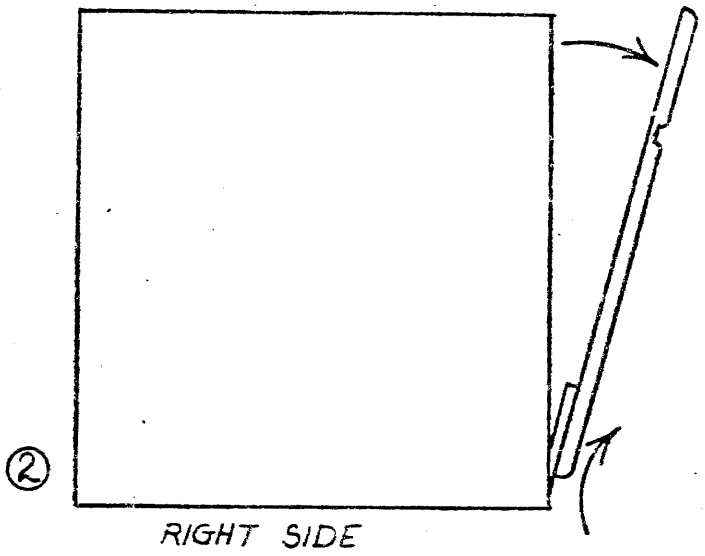
② LEFT SIDE VIEW

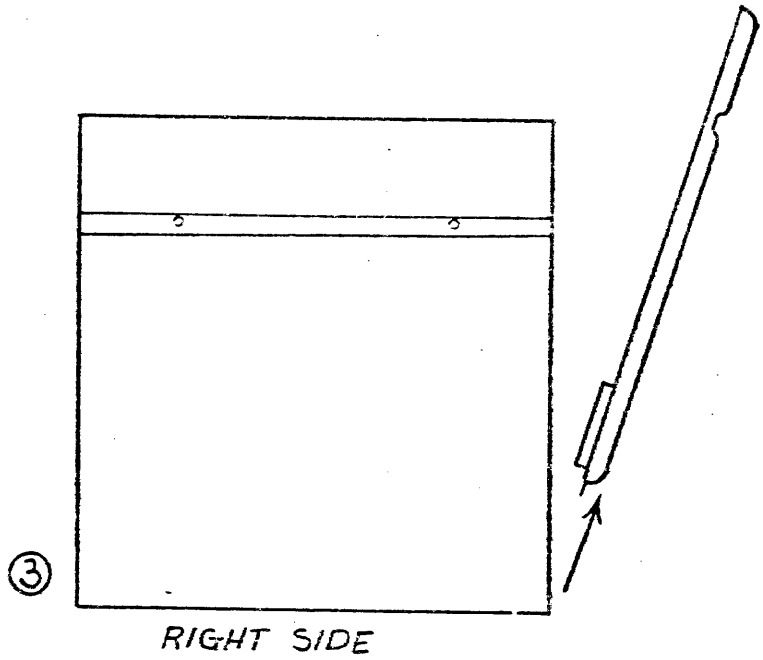


③ LEFT SIDE VIEW

PERQ BASE UNIT:
REMOVING THE FRONT COVER







PERQ BASE UNIT

REMOVAL OF REAR COVER

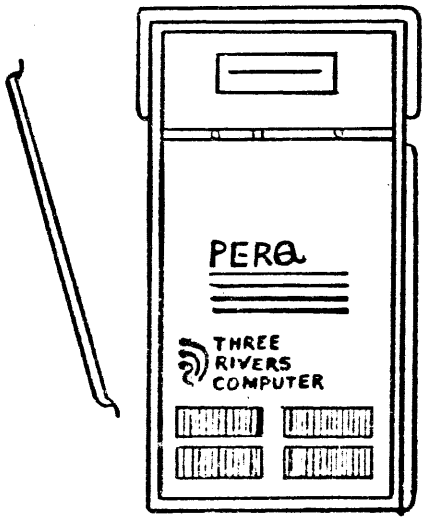
1 REMOVE SCREWS (SHOWN BY ARROWS)

2 TILT AND LIFT COVER SIMULTANEOUSLY

3 REMOVE COVER

REVISIONS

LTR	ECO NO.	DATE	APPV'D
B	00078	2-2-82	[illegible]



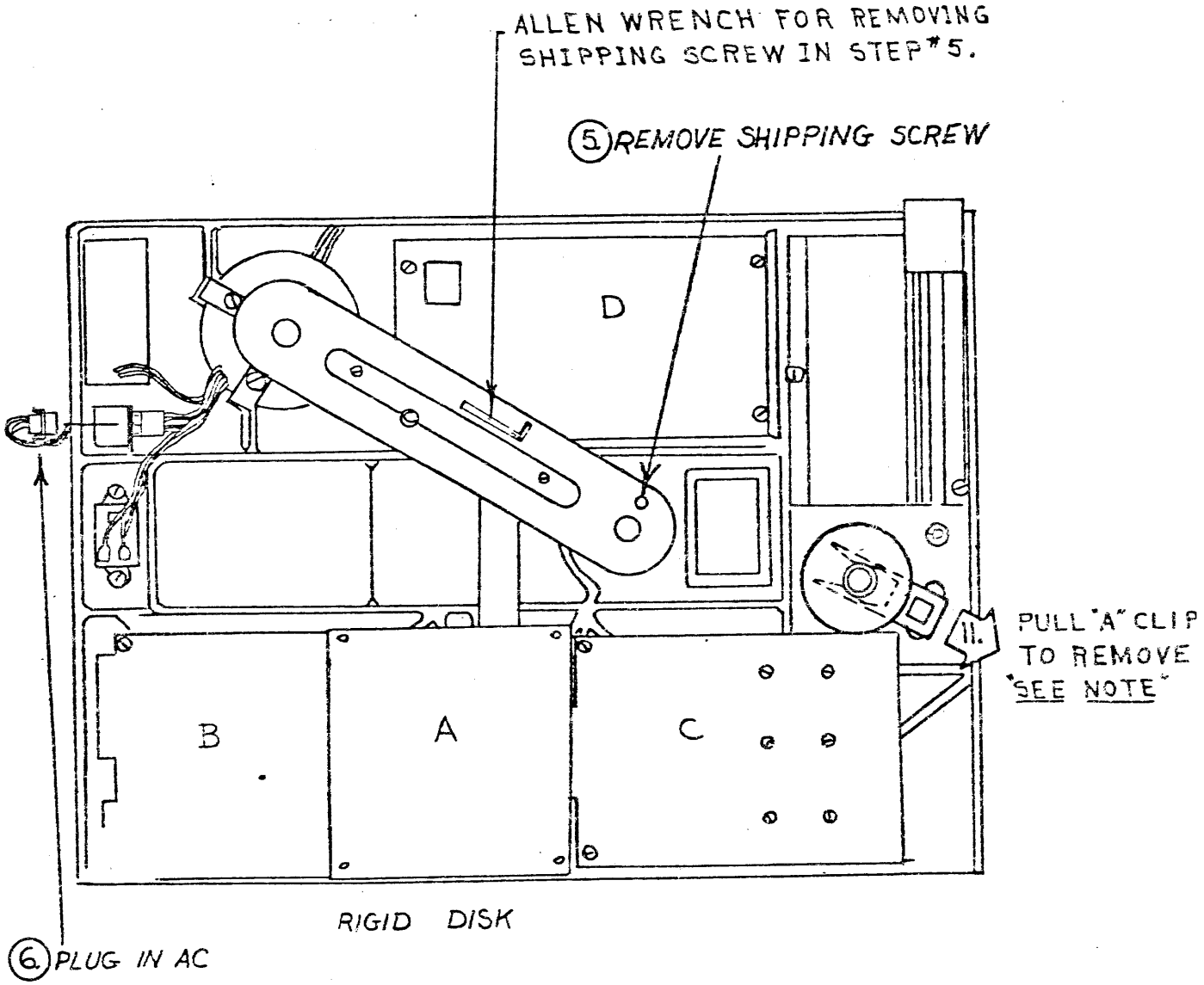
FRONT VIEW

NEXT ASSY PAK-001

TITLE LEFT SIDE COVER REMOVAL

Three Rivers Computer

			SIZE	CODE	IDENTIFICATION	VAR	REV
DRAWN	DIV	2-2-82	A	PRQ	AD-0135		B
CHECKED	[illegible]	2-8-82					
APPV'D	[illegible]	2-2-82	PROJ	PERQ		SHT	2 OF 4



NOTE: DO NOT REMOVE "A" CLIP, UNLESS PERQ IS "ON" AND DISK IS ROTATING. SEE INSTRUCTION 1.2.10 & 1.2.11

SA4000 UNPACKAGING INSTRUCTIONS

CAUTION: These directions **must** be carefully followed to insure the correct operation of the drive.

1. The **SPINDLE LOCKING SCREW must** be removed **before** applying AC power to the drive motor.

   IMPORTANT: Retain locking screw for re-installation prior to transporting drive.

2. **CAUTION: ROTATE SPINDLE IN DIRECTION OF ARROWS ONLY.**

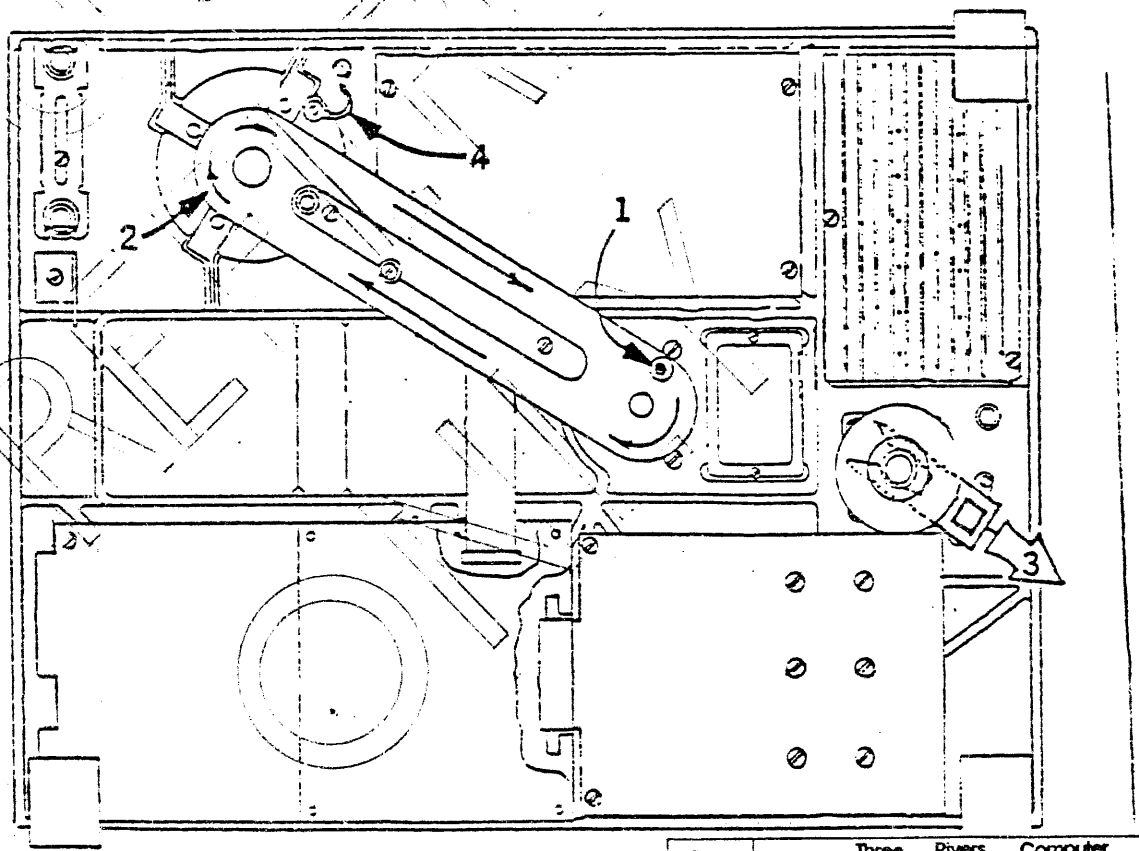
3. **Remove the ACTUATOR LOCK** in the following sequence:

   1. Energize AC power
   2. Withdraw the lock from the stepper assembly.

   IMPORTANT: Retain the lock for re-installation prior to transporting the drive.
   It is recommended the heads be moved to the extreme inside tracks prior to re-installing the actuator lock.

   CAUTION: It is recommended that the AC power be energized before removing or installing the actuator lock. Failure to do this may result in **media damage.**

4. OPTION — To isolate AC motor - remove GND strap.



Three Rivers Computer		
REFERENCE INST.	PROJECT PERQ	
DRAWN BY T.T.	DRAWING NO. PRQ-AD-0611-A	SHEET 1 OF 1
CHECKED BY		

PERQ Introductory User Manual

Donald A. Scelza

Diana Connan Forgy

Brad A. Myers

Bob Amber

This manual provides an introduction to the use of the Three Rivers Computer Corporation PERQ Computer.

Three Rivers Computer Corporation
720 Gross Street
Pittsburgh, PA 15224
(412) 621-6250

The information in this document is subject to change without notice and should not be construed as a commitment by Three Rivers Computer Corporation. The Company assumes no responsibility for any errors that may appear in this document.

Three Rivers Computer Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

1. Preface

This manual provides an introduction to the PERQ. The manual explains general concepts of the PERQ system; you should be familar with the concepts before using the PERQ. This manual also supplies information to boot the PERQ system.

2. Introduction.

PERQ is Three Rivers Computer Corporation's personal computer which provides an integrated computing system for a single user. All PERQs come with the following hardware features:

a) 16-bit micro programmed processor
b) High-resolution graphics using a 1024 X 768 bit mapped raster display
c) Standard keyboard with special function keys
d) 12-megabyte Winchester technology disk drive
e) RS232 & GPIB I/O interfaces
f) 512k bytes of main memory
g) Pointing tablet
h) Speech output

Optional features are:

a) 24 megabyte disk
b) 4k X 48-bit writable control store
c) 1 megabyte of main memory
d) 8 inch floppy disk drive

PERQ provides a large, 32-bit, segmented virtual address space. Virtual addresses are mapped into a 20-bit physical address.

The supplied software provides a functional program development environment. The system includes a text editor, a Pascal compiler, interactive stack dump for debugging, file management utilities, and support for micro program development.

3. Turning the PERQ on and off.

Refer to the Installation Guide before using your PERQ.

The PERQ's power switch is located on the front panel in the groove below the floppy disk drive. The switch is right of center and is labeled OFF/ON. Pushing the switch to the right powers on the PERQ. The fans start; if you do not hear them check to see that the machine is plugged in. When the PERQ is plugged in, the small neon light in the lower right hand side of the back glows. If the light is out, check the electrical outlet. If there is power at the outlet or the light is glowing and you still do not hear the fans then call your local service representative.

The PERQ takes about two minutes to boot after it is turned on allowing the disk to spin up to speed. At this point the Diagnostic Display (DDS) reads 999. Refer to Section 12 for boot procedures and information on the DDS.

If the machine has not booted after two or three minutes, try pressing the Boot button on the back of the keyboard. If this does not cause the machine to boot, call your local service representative.

After the PERQ boots, you will be asked to enter the time of day. The time and date is given in the form:

DD MMM YY HH:MM:SS

The time notation uses a 24-hour clock. If you were logging in on the 27th of March, 1982 at 2:30 in the afternoon you'd type

27 mar 82 14:30:00

The seconds are optional and the spaces in the date can be replaced by hyphens (27-Mar-82). There is usually a default date and time given at login. If the date is accurate, enter only the time or, type a carriage return to confirm both date and time. If it is not accurate or if there is no default, enter the correct date and time.

Next, you will be asked to login. You must supply a user name and password to be allowed access to the machine. After the operating system has been loaded onto a machine, the names defined are "Guest" and "" (the empty string; just press the Carriage Return key). See "UserControl" in the Utility Programs Manual for instructions on how to add other user names. After you have successfully logged in, the command interpreter, Shell, will be loaded and you will be able to execute commands.

Before turning the PERQ off, you should always log off using the Bye program. If you just power down, some temporary files on the disk will not be cleaned up. Type:

```
BYE OFF
```

to log off and turn the power off for the machine automatically. If this is not possible, you can always turn the machine off by moving the power switch to the left. See "BYE" in the Utility Programs Manual.

4. Control Characters and Special Keys.

The PERQ operating system recognizes a number of special control characters. These control characters perform simple input line editing and program control.

The valid control characters are:

BACK SPACE or ^H - erases the last character typed by the user.

^W or ^BACK SPACE - erases the last word typed by the user.

OOPS or ^U - erases the last line typed by the user.

^C - typed once causes a current program to abort the next time that it asks the operating system for input.

^C - typed twice causes the current program to abort immediately. However, this does not cause user command files to exit; see ^SHIFT C.

^S - causes program output to the screen to be suspended.

^Q - allows output to the screen to resume after a ^S.

^SHIFT C - causes a dump of the runtime stack and an immediate return to the Shell. If a command file was being executed, it is aborted and control is returned to the keyboard.

^SHIFT D - causes a dump of the runtime stack to be printed and the preliminary debugger (called Scrounge) can then be entered. If the debugger is not entered, the original program will resume execution. If the debugger is used, the user can request that the program be resumed after investigating the state. For a more complete description of the debugger, see the "PERQ Utility Programs Manual."

HELP - pressing the HELP key by itself displays a general help message. If you press the key after typing a command, the display contains specific help information for the command.

5. The Command Interpreter, The Shell.

The command interpreter for the PERQ operating system runs as a separate user program. It is possible for the user to replace the command interpreter with his own (see the section on "Login" in the PERQ Utility Programs Manual). The name of the command interpreter supplied with the system is the Shell.

The Shell takes commands from the user terminal or from a user command file and executes them. It does not distinguish between upper and lower case letters; you can use whichever you prefer. The commands in a user command file look exactly as if they were typed at the terminal. However, the cursor is a lighter shade so that you can distinguish command file execution from a typed command. The PERQ Utilities Programs Manual describes user command files and how to use them.

The general form of a Shell command line is a command or program name, followed by any number of optional parameters, followed by any number of optional switches. Some of the switches take parameters. For example:

```
Compile SourceProgram/Symbols=32
```

compiles a Pascal program in file SourceProgram using 32 symbol table blocks. All switches begin with a "/".

An example of a program that takes a number of parameters on the command line is the Copy program. The form is:

```
Copy SourceFile~DestinationFile
```

An example of a program that takes multiple switches is the Linker. A Linker command line might be:

```
Link SourceProgram/Verbose/StackSize=1024
```

When you supply a command line to the Shell, it extracts the first symbol on the line and does a unique substring lookup of the symbol against a small set of commonly used commands. You can get a list of these commands by typing "?" or the HELP key. If a match is found, the Shell executes the command. If no match is found, the Shell assumes the symbol is the name of an executable runfile (the output of the Linker), and attempts to execute it. Commands can also be invoked by means of a menu if you have booted from the harddisk. See the section on "PopUp Menus" below.

6. Specifying Commands and Arguments.

The Shell uses a default file name as the parameter to certain programs if no parameter is provided. This file name is the last file name typed to one of these programs. The Editor, Compiler, and Linker use and set the default file name. The TypeFile program uses the default file name but does not set it. The PERQ Utility Programs Manual provides more details on these programs.

Other programs require that you specify arguments. If a command requires an input and an output argument, you can specify the arguments as either

```
INPUT~OUTPUT
```

or

```
INPUT OUTPUT
```

You can separate the input and output arguments with multiple spaces; only the initial space delimiter is relevant.

However, if a command accepts multiple input or output arguments, you must separate like arguments with commas (,) and distinguish input from output with the tilda character (~). For example:

```
input1,input2,...inputn ~ output1.output2,...outputn
```

If a command accepts multiple input arguments and no output arguments, you must separate the arguments with a comma (,).

Switches modify the action of the command and therefore must follow the command specification. An exception is the Help switch (either type /HELP or press the HELP key); when specified before the command, the Help switch supplies general information and when specified after a command, the Help switch provides specific information. Switches always start with a slash (/) and are generally optional. If a switch accepts a parameter, specify the parameter after the switch, but preceded by an equal sign (=). For example:

```
/switch=parameter
```

The effect of a switch is global; regardless of where the switch appears on the command line, it has the same effect. A switch applies to every argument. If a command accepts multiple input or output arguments, no switch applies to one and not another argument. The PERQ Utilities Programs Manual provides details on the general syntax.

For most programs on the PERQ, arguments that have defaults will print the default answer in square brackets ("[]"). This answer can be chosen by simply typing a carriage return (thus providing an empty argument). To supply a different value, type the value followed by a carriage return. For example, if:

```
Delete FileName.Seg [No]:
```

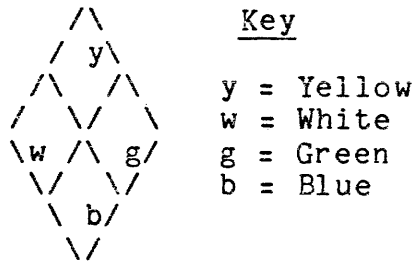
is the prompt for an argument, a carriage return means no. Of course, you could type "yes" or "no" as the argument. Most programs do not distinguish between upper and lower case for arguments or commands. In addition, you can abbreviate a command to the number of letters unique to other command names. However, this does not work for filenames.

The user's profile file specifies the set of commands recognized by the Shell. You can use a copy of the file DEFAULT.PROFILE initially. Later, you may wish to define commands of your own. Refer to the section on "Profiles" below.

7. The Pointing Device.

PERQs are supplied with a bit pad and pen or puck. In the normal mode, the cursor on the screen follows the movement of the pen. If the pen is in the upper-left corner of the tablet, the cursor will be in the upper-left corner of the screen. The PERQ can read the pen position when the pen is near the tablet surface. If you press down on the pen, a small switch closes. This is called a "press" of the pen. The editor and some other programs use the pen for pointing and drawing.

As an option, the pen can be exchanged for a four-button puck. The puck rests on the tablet and uses a circle with cross hairs for positioning. The four buttons are arranged in a diamond as shown below. In all cases, a press of the top (yellow) button on the puck functions exactly like the press on the pen. For programs which do not distinguish between the buttons, the other buttons also act like a press. The editor, however, assigns different functions to the other three buttons (see the "Editor User's Guide").



8. PopUp Menus.

The Shell and the FLOPPY and FTP utility programs allow their arguments to be entered by means of menus. (This only works, however, if you have booted from the harddisk.) The menu holds a list of all legal commands or arguments, and the pointing tablet can be used to specify the selection. Using a menu is sometimes more convenient than typing out the name of the command or argument desired. PopUp menus appear when requested. When the selection is made, the menu disappears and restores the screen space that was covered by the menu. Using PopUp menus, therefore, does not require sacrificing any screen area.

For the Shell, you can invoke a menu whenever the prompt (">:") is displayed and no characters have been typed. (The prompt is actually a dark grey, right pointing triangle.) To invoke a PopUp menu, simply press the pen or puck. The menu appears at the current cursor position. The cursor can then be moved up or down inside the menu to select the desired command. The selected command is highlighted by reverse video. A press over the selected command causes it to be invoked just as if the user had typed the command on the keyboard.

If the menu is not large enough to hold all the commands, a scrolling mechanism is provided. When scrolling is necessary, a "gauge" is displayed in a black border at the bottom left of the menu. When you move the cursor over this area, the cursor changes to a scroll. A press here allows scrolling. Moving the cursor to the right while pressing causes the menu text to scroll up and moving the cursor to the left while pressing causes the text to scroll down. The further the cursor is moved from the original press position, the faster the text scrolls. A line in the gauge shows how fast the menu is scrolling. Of course, you cannot scroll past either end of the menu. (Each end is signified by a row of "~"). Releasing the button causes scrolling to stop.

If you press in an illegal part of the menu, or if you try to invoke a menu and have typed some text, the PERQ beeps. If a menu is displayed and you press outside of the menu, the menu disappears, causing no side effects. In addition, if you type ^C or ^Shift-C while a menu is displayed, the menu disappears.

9. The "Lights".

The PERQ does not have a front panel full of lights like many computers. To show the occurrence of certain long operations, the PERQ simulates these lights by inverting small squares on the top of the PERQ screen. If the standard system window is displayed, the lights appear inside the title line.

Currently, the PERQ operating system uses three lights. The leftmost is used to show when the PERQ is doing a disk or floppy recalibration. This is done when the microcode is confused about where the heads are. Opening the floppy door during a disk operation usually causes this to happen. The light appears about 5/8 inches from the left margin.

The next light is used when the file system is attempting to fix up a file at run-time. This is needed occasionally when a file system operation has not completed normally. This light appears about 1-1/2 inches from the left margin.

The last light defined by the operating system is used for swapping. While a swapping operation (swap in or swap out) is in progress, this light is "lit." The swapping light appears about 2-1/4 inches from the left margin.

The Pascal compiler uses an additional light to indicate that the compiler is swapping its symbol table blocks between memory and the disk. (The compiler swaps symbol table blocks more frequently in systems with 256k bytes of main memory than in systems with 512k bytes or 1024k bytes of main memory. Therefore, the light is lit more often on minimum memory systems.) The compiler symbol table block swap light appears about 3 inches from the left margin.

10. Specifying a Filename.

The PERQ file system is described in detail in the PERQ File System Utilities Manual. This is a brief overview, a short introduction to path and filenames, a description of the various ways to specify a filename, and an explanation of the wildcard convention.

Overview of the PERQ File System

The PERQ file system has a hierarchical structure. This is reflected in the syntax of filenames. Files are stored on devices, which are divided into partitions. Each partition contains a number of files and directories; each directory may contain other directories and files.

Introduction to Path and File Names

The route that is traveled to reach a filename is called a PATH. A full path name specifies device, partition, directories, and filename, in that order. The syntax of a path name is:

device:partition>[directory>]filename

The brackets surrounding the "directory" part indicate that there may be zero to nine occurences of ">directory". Note that if you specify a directory, the brackets are NOT part of the syntax. To specify the current directory, or one of its subdirectories, you can omit the :partition> and the directory> syntax elements (see the description of the Path command in the PERQ Utilities Programs Manual).

1. Device names are usually assigned by the user. Examples of devices include floppy and hard disk. The syntax used to denote a device name is:

    device:

If you omit the device name, the system assumes the name of the device you booted from.

Each PERQ is given a name, which is usually the device name of the hard disk.

A device is accessable when it is mounted. See the description of "Mount" in the PERQ Utilities Programs Manual.

2. Partitions are set and named at device initialization time. You can modify partitions using the Partition program, described in the PERQ Utilities Programs Manual. Each device is divided into a fixed number of partitions;

the recommended size for a partition is 10,080 or fewer 256-word blocks. Files must fit entirely within a single partition as they cannot cross partition boundaries. Generally there are three partitions on a 12-megabyte disk and five on a 24-megabyte disk. Examples of partition names are BOOT and USER.

3. Directories are easy to create, destroy, rename, and move around. There can be multiple directories in a partition. These are denoted by the symbol > and can take names of up to 25 characters. There can be up to 9 directories listed in a full path name. The symbol:

..

is a convenient way of referring to a parent node in a tree. It goes up one node. An example of its use is:

SYS:BOOT>NEW>..>filename

This will look up filename in SYS:BOOT, rather than in SYS:BOOT>NEW>. The symbol

.

refers to the current directory.

4. Filename is the last thing specified in a file specification. A filename can have up to 25 characters.

The names of all of the directories and the filename cannot exceed 80 characters. You can include most special characters in a filename by surrounding the special characters with a single quote ('). For example, you could include the asterisk (*) in a filename by specifying '*'. Note that you cannot include the special and control characters described in section 4.

When you boot, the system takes as default device and partition the device and partition that you booted from. For example, the system might come up with a default of:

SYS:BOOT>

Ways to Specify a Filename

1. You can specify a full path name:

device:partition>[directory>]filename

the brackets, which are not part of the syntax, indicate that you can list up to nine directories. This is useful if you want to access a file on a different device from the one you're using.

2. You can use the default device that was determined at boot time:

:partition>[directory>]filename

Using this syntax enables you to look for a file in a different partition.

3. You can use the default device and partition that were set at boot time:

>[directory>]filename

The search then starts at the first directory specified.

4. You can just type:

filename

and the search begins at the current directory (which may be set by you, using Shell's PATH command). This can involve any device, any partition, and any directory in that partition.

Setting the default path doesn't affect the default device and partition used in forms 1, 2, and 3.

Along with the concept of a current directory, the file system provides a search list. This gives the user the ability to specify a set of directories to be searched, in a specific order, when a filename is specified. Refer to the SetSearch command description in the PERQ Utilities Programs Manual.

Wildcard Convention

A number of PERQ programs use a wildcard convention when looking up files. The wild cards are as follows:

* matches 0 or more characters.
& matches 1 or more characters.
# matches exactly 1 character.
'0 matches any digit.
'A or 'a matches any alphabetic.
'@ matches any non-alphanumeric.
'* matches *. Other wild cards can be quoted also.

There can be any number of wildcards in file specifications to programs that handle this convention. Examples of wildcard usage are:

Dir *.Pas

Dir &Boot*

The first command gives a directory of all files with a .pas extension. The second command gives a directory of

all files that have one or more characters followed by the characters "boot", followed by zero or more characters.

11. Default File Extensions.

An extension is a conventional sequence of characters that appears at the end of a filename. In the PERQ operating system there is nothing special about extensions. However, there are certain conventions that are used in the system. An extension is found by finding the last "." in the filename and taking the following symbols. Backup files conventionally have a "$" as the last character of their last extension; they take the form Name.Ext$. Following is a list of the standard extensions and how they are used.

.PAS - Pascal source files usually have this extension.

.SEG - The Pascal compiler produces a .SEG file when it compiles a .PAS file. .SEG files are used as input to the Linker and contain the code that will be executed when the program is run.

.RUN - Files produced by the Linker have this extension.

.MICRO - PERQ microcode files have the extension .MICRO. They can be used as input to the micro assembler, PRQMIC.

.BIN - The runnable version of PERQ microcode is contained in .BIN files. These files are produced by the micro placer PRQPlace.

.REL, .RSYM, .INT - These three file types are temporary files that are produced by the micro assembler and placer. They can be deleted after a .BIN file has been created.

.DFS - These files are used to communicate definitions between programs that may be in different languages, for example, between Pascal and microcode.

.CMD - A number of programs on the PERQ accept commands from a file as well as from the keyboard. Files that contain the commands usually have this extension.

.DR - In the PERQ file system directories are files. These files appear in a directory listing with the extension .DR.

.KST - Character set definitions are kept in files that have the extension .KST.

.MBOOT, .BOOT - When the PERQ is booted it reads the microcode interpreter and Pascal operating system from files that have the extension .MBOOT and .BOOT.

.ANIMATE, .CURSOR - Pictures used by utilities that are put into the cursor that follow the pen or puck.

.INDEX - An index to .HELP files.

.DOC - Formatted documentation has this extension.

.HELP - Files containing help about a subsystem use this extension.

12. Booting the Machine.

The PERQ can be booted in one of two ways. First, when the machine is powered up it will go through the boot sequence. Second, the machine can be booted by pressing the Boot button on the back of the keyboard. In either case the same sequence of events happens.

The boot sequence for PERQ has three steps. As each of these steps progresses, the Diagnostic Display, DDS, increments. The DDS is a three-digit number display. On earlier models of the PERQ, it is found on the inside of the machine behind the front cover; on later PERQs it's under the keyboard. If any step fails it is possible to look at the Diagnostic Display and see where in the boot sequence the failure occurred. The Fault Dictionary provides a list of the DDS values and explanations of their meanings.

In the first part of the boot sequence, microcode is executed out of a small ROM. This ROM covers the lower 2k of standard control store during this part of the boot sequence. This microcode runs a simple diagnostic on the processor and memory systems. If there are any errors, the microcode halts. The value in DDS gives the reason that the machine halted. Once these diagnostics have been passed, the microcode makes a decision about which device is to be used for booting. Currently, there are three possible boot devices.

1. The first alternative is booting from the hard disk. The microcode tries to boot from the hard disk.

2. The second choice of boot devices is a floppy disk. The microcode checks to see if there is a floppy in the floppy drive. If there is a floppy in the drive, PERQ will check to see if the floppy is a boot floppy. If so, the second part of the boot will be done from the floppy.

3. The third possible boot device is another PERQ. The microcode determines if there is a PERQ Link Board plugged into the I/O Option slot of the machine. If the board is plugged in, and there is another PERQ on the other end of the link, the booting PERQ will wait for commands from the link.

If all of these fail, then the DDS will contain an indication of what the error is. See the Fault Dictionary Manual for an explanation of the display number.

After the boot device has been chosen, the second part of the boot sequence can begin. In this part, the PERQ reads 3k words of microcode from the selected boot device. This microcode is in two sections, a more extensive diagnostic (VFY) and a system boot loader (SYSB). VFY attempts to verify that all of the CPU and Memory systems are working. Any failures that VFY discloses are displayed on the DDS.

If all went well, the microcode determines what set of interpreter microcode and system Pascal code is to be loaded. It does this by

checking to see if any key is being held down on the keyboard. If a key is being held down, that key specifies which boot is to be done. If no key is held down, the default boot will be done. The default is the same as holding down "a". Hold the key down until a pattern flashes on the screen. Any of the 26 alphabetic keys can be used to specify a boot. All lower-case characters cause a boot from the hard disk. Upper case characters cause a boot from floppy. If you type "Details/Boots" the Details program will provide you with a list of all of the valid boot characters. Once a boot has been chosen, the microcode interpreter and PERQ operating system are loaded into the PERQ. Control is then transferred to the third portion of the boot sequence.

In the third portion of the boot sequence, the interpreter microcode does any initialization that is needed and then starts to execute the PERQ operating system. The PERQ operating system also increments the DDS. If there were no errors during the boot sequence, the machine will be running and the DDS will read 999.

13. Run-time Errors.

When the operating system or a user program discovers an error condition, it raises an "exception" (see the "PERQ Pascal Extensions" manual for an explanation of exceptions). If an exception is not handled, it is given to the preliminary debugger, called Scrounge (see the "PERQ Utility Programs Manual"). First, a dump of the user state is produced and then the user is asked if he wants to debug. If not, the program is aborted and control returns to the Shell and any active command files are terminated. If the user decides to debug, the program can be continued after the point where the exception was raised. It is usually a bad idea to continue from uncaught exceptions. The user can also abort the program and return to the Shell.

14. Profiles.

Profiles can be used to tailor your PERQ. The profile is a text file which contains commands that define characteristics of certain utility programs. For example, a profile can direct the Login program to initialize the default path and searchlist. Each user of the system can have his own profile; the UserControl program can assign each user a profile file.

To create a profile file, copy DEFAULT.PROFILE to your own directory. For example:

```
COPY SYS:BOOT>DEFAULT.PROFILE SYS:USER>MYDIR>MYPROFILE
```

Now run UserControl and specify SYS:USER>MYDIR>MYPROFILE for the profile file. You can then edit the file and establish your specific conventions.

Each entry in the profile file begins with a number sign (#), followed by the name of the subsystem. For example, #LOGIN. The subsequent lines contain switches or data for the subsystem. The general format of a profile file is as follows:

```
#program1 <switches or input for program1>
          <more data for program1>
#program2 <switches for program2>
  ...
```

For example:

```
#Login /Path=Sys:User>Mydir
       /SetSearch=Sys:Boot>Library
       /CursorFunction=7
#RandomUtility /MaxSize=100
```

The format of each list of entries in the profile is defined by the utility program that uses the profile. You should read the documentation for a particular utility to determine whether it reads the profile, and if so, what entries can be included in the profile. You may use the profile in your own programs. See the Profile module in the Operating System Interface manual.

The Shell commands are specified in the #ShellCommands section of the profile. The format for each line in this section is:

<implementation><useDefault><setDefault><screensize><cmdname>

Where <implementation> is the command line Shell issues to execute the command. <useDefault> tells whether to use the default file name if no file name argument is specified for the command. <setDefault> specifies whether to set the default file name if an argument is provided. <screensize> can limit the screen to less than full size. <cmdname> is the name that will be used to invoke the command. This name and the rest of the line is printed when a

"?" is typed so the additional comments are provided to explain the command.

PERQ Utility Programs Manual

Diana Connan Forgy

Donald A. Scelza

Brad A. Myers

Bob Amber

This manual provides an introduction to the use of the PERQ Utility Programs.

Three Rivers Computer Corporation
720 Gross Street
Pittsburgh, PA 15224
(412) 621-6250

The information in this document is subject to change without notice and should not be construed as a commitment by Three Rivers Computer Corporation. The Company assumes no responsibility for any errors that may appear in this document.

Three Rivers Computer Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

1. Preface: Notation Conventions.

This manual describes the PERQ Utility Programs and commands to the Shell. The command descriptions are in alphabetical order regardless of whether the command is implemented as a Pascal file or through the Shell directly (The Introductory User Manual describes the Shell.)

The descriptions of the commands given in this manual observe the following notational conventions:

- Lowercase text indicates a variable whose actual value is determined when the command is entered
- Square brackets ([]) indicate optional entries in a command line. Note that when an option is used, the brackets are not part of the syntax.
- The circumflex (^) indicates the control key
- CR indicates carriage return
- SHIFT indicates the shift key

2. Introduction

This manual describes the PERQ Utility Programs and explains their use. As required, the manual references online or other Three Rivers Computer Corporation publications.

Most of the commands described in this manual are implemented as Pascal files, but some are implemented directly by the Shell.

To issue a command, type a command line in response to the default PERQ prompt. You can also use a pop-up menu, as described in the PERQ Introductory User Manual, to issue Shell commands.

If you elect to use a pop-up menu, the menu displays all of the valid commands. Use the pointing tablet to specify the selection. The FLOPPY and FTP utilities also permit you to enter their respective commands through pop-up menus; a press of the pointing tablet in response to either of the prompts for these utilities displays a menu for the utility commands.

If you type a command line, the line consists of a command name, input and output arguments as needed, optional switches, and a line terminator.

The command name describes the action the system performs or the name of the utility that performs the action. When submitting a command that is implemented in the Shell, you need not type the entire command name; you can abbreviate the command name to the number of letters unique to other command names. For example, the Mount command can be abbreviated M because it is the only command within the Shell beginning with that character. You can abbreviate the Path command to Pat, but not Pa because the Shell includes the Pause command. The question mark command (see section 3) lists all commands. You can modify the list by changing your profile file.

Input and output arguments further define the command action. The input and output arguments are usually file specifiers. Some commands require arguments as part of the command line. Other commands use the default file as the argument if you do not supply one.

If you neglect to supply a required argument, the utility prompts with a few words indicating the general nature of the missing argument. For example, the Rename command performs as follows:

```
>RENAME
File to rename:  OLDFILE
Rename OLDFILE to:    NEWFILE
```

The single line format for the above command is:

```
RENAME OLDFILE ~ NEWFILE
```

or

RENAME OLDFILE NEWFILE

You can mix formats; the utility prompts for whatever you omit. For example:

```
RENAME OLDFILE
Rename OLDFILE to:  NEWFILE
```

There are no defaults for prompts. You must supply a response. However, your response to the prompt can request help (type /HELP or press the HELP key). If you request HELP, the utility displays specific information and then exits to the Shell.

If you neglect to supply an argument to a utility that uses the default file (for example, the Editor or Linker), the utility appends the default file name to your command. For example, if the default file is sys:user>myfile, typing the command:

EDIT

is the same as typing:

Edit sys:user>myfile

If a command requires an input and an output argument, you can specify the arguments as either

INPUT~OUTPUT

or

INPUT OUTPUT

You can separate the input and output arguments with multiple spaces; only the initial space delimiter is relevant.

However, if a command accepts multiple input or output arguments, you must separate like arguments with commas (,) and distinguish input from output with the tilda character (~). For example:

input1,input2,...inputn ~ output1.output2,...outputn

If a command accepts multiple input arguments and no output arguments, you must separate the arguments with a comma (,).

Switches modify the action of the command and therefore must follow the command specification. An exception is the Help switch (either type /HELP or press the HELP key); when specified before the command, the Help switch supplies general information and when specified after a command, the Help switch provides specific information. Switches always start with a slash (/) and are generally optional. If a switch accepts a parameter, specify the parameter after the switch, but preceded by an equal sign (=). For

example:

```
/switch=parameter
```

The effect of a switch is global; regardless of where the switch appears on the command line, it has the same effect. A switch applies to every argument. If a command accepts multiple input or output arguments, no switch applies to one and not another argument. Some switches are mutually exclusive (for example, /ASK and /NOASK). If you specify a switch that conflicts with a previously specified switch, the last occurence has precedence. Likewise, if you change parameters by specifying a switch multiple times, only the last occurence has an effect.

All of the commands and utilities described in this manual accept the /HELP switch (either type /HELP or press the HELP key) to provide general information. Note that /HELP overrides all other switches; the utility displays specific help and then exits.

Carriage return is the line terminator for all commands.

Rather than typing a command line to initiate and direct a utility, you can use a user command file. A user command file is a text file containing a series of commands interpretable by the various utilities.

A user command file is a sequential file containing a list of utility specific commands. Rather than typing commonly used command sequences, you can type the sequence once and store it in a file. The user command file is specified in place of the command line(s) normally submitted to the utility.

To inititate user command files, replace the command line for a utility with a file specifier, preceded by an at sign (@). The utility requesting input then accesses the specified file and starts to read and respond to the commands contained within it. For example, to initiate a file of FLOPPY commands, type the following in response to the FLOPPY prompt:

```
FLOPPY>@FLOPPY.CMD
```

The FLOPPY utility accesses the file and then executes the commands contained within the file FLOPPY.CMD. The default file type for user command files is .CMD. Thus the above command line could also be typed as follows:

```
FLOPPY>@FLOPPY
```

You can nest user command files by simply specifying @file within the user command file. Also, the last line of a user command file can invoke another command file and recursive use is permitted.

To comment a user command file, start the comment line with an exclamation mark (!). The exclamation mark can appear anywhere on a line in a user command file, but remember that the system ignores all characters after the exclamation mark.

3. Question Mark: ?

The question mark command lists all the commands implemented in the Shell. Additionally, the command provides some information on what each command does and how it is called.

4. Append

Append copies one or more files to the end of an existing file. The command accepts a list of input files, separated by commas, and puts each file on the end of the first. For example, the command line:

```
APPEND file1,file2
```

puts file2 on to the end of file1. The append operation is successive. For example, the command line:

```
APPEND file1,file2,file3
```

first puts file2 on to the end of file1 and then puts file3 on to the end of file1.

5. Bye

Bye logs you off the PERQ. Type:

```
BYE
```

and it types your name, the date, and time of logoff. Login then loads and you or some other authorized user can log back on.

The command

```
BYE OFF
```

logs you off and turns off your machine. OFF may be disabled on your PERQ; if so, call Three Rivers Computer Corporation for details on reenabling it. OFF requires some special microcode which should be available on your machine. If not, Bye will log off and request that you power down the machine by hand. In this case, ^C will cause Login to run.

The command

```
BYE WAIT
```

logs you off the machine and sends the hard disk heads to the center of the disk (the highest disk address). Bye then prints a message and waits for power down or ^C. When shipping a PERQ, always type BYE WAIT to position the disk heads at the center of the disk.

The command

```
BYE HELP
```

describes the options available for Bye. You can specify all of the options (OFF, WAIT, and HELP) as switches.

It is very important to log off using BYE before powering down the machine. If you do not, certain temporary disk files will not be deleted and they will stay around using disk space until you run the Scavenger (see below).

6. Chatter

Chatter allows a PERQ to act as a terminal, using an RS232 line for communication. It is invoked by typing:

CHATTER

While in Chatter, some special functions can be invoked by typing ^R, the function's code letter, and CR. These include

S to save everything that comes from the remote computer in a file.

T to transmit a file across RS232, as if it were being typed at the keyboard.

C to close a file after doing a Save.

B to change the RS232 baud rate. The default when Chatter comes up is 4800 baud.

Q to quit Chatter and return to the Shell. This doesn't log off or disconnect the remote host.

This menu will appear at the very top of your screen to prompt you after you've typed ^R.

While in Chatter, if you find that the characters you type are not being echoed properly, one of two things may be wrong: the baud rate may be inappropriate or your RS232 cable might not be connected properly.

Chatter cannot be used as a half duplex terminal since it does not echo characters locally.

7. PERQ Pascal Compiler

The PERQ Pascal compiler translates your Pascal source program into a .Seg file that can be linked and run. There are several ways to invoke the compiler and several options that you can use with it. The command line takes the form:

```
COMPILE [inputfile] [~] [outputfile] [/switch(es)]
```

Examples of legitimate compiler calls include:

```
COMPILE Program.pas

COM ProgramX
    (Note that the .pas extension is implicit;
     if ProgramX does not exist, the compiler
     looks for ProgramX.PAS.)

COMP Program2~Program1
    (creates the output file Program1.SEG)

COM
    (compiles the default file)

COM /symbol=32
    (compiles the default file with 32 symbol
     table blocks)
```

If you want to compile a program immediately after editing it, you need not specify its name since the Shell remembers the last file edited, compiled, linked, or run.

Certain switches may be included in the source program text. See the PERQ Pascal Extensions Manual for more detailed information on switches and other compiler features.

8. Copy

Copy creates a new file identical to the specified source. Its command syntax is:

COPY SourceFile[~]DestinationFile

Copying works across devices and partitions (see the PERQ Introductory User Manual for details on devices and partitions). You can also specify the non-file-structured devices CONSOLE: and RS:. If you copy a file to the RS232 interface, by default the interface is driven at 9600 baud. To change the baud rate, run the SetBaud program (see below).

The source file for Copy may contain wild cards (for a description of the wild cards, see the PERQ Introductory User Manual). If the source contains wild cards, the destination must contain the same wild cards in the same order. In this case, Copy matches all files in the directory with the source pattern. For each match, the part of the source file name that matches each wild card replaces the corresponding wild card in the destination. As an example, for the command:

COPY foo*.abc# anotherdir>*baz.rmn#z

the input file "FOOZAP.ABCD" would be copied into a new file named "anotherdir>ZAPbaz.rmnDz".

If wild cards are used, Copy asks for verification of each file copied. This can be disabled with the switch "/NOASK" or enabled with "/ASK". The latter is the default. Copy also requests confirmation before overwriting an existing file. You can override this action with the /NOCONFIRM switch. (Note that the /NOCONFIRM switch implies the /NOASK switch.)

Wild cards are not allowed in the directory part of the source file. However, Copy uses the search list to try to find the source. Note that this is different from Rename and Delete which always look in only one directory.

When the source file name contains no wild cards, the destination file name may contain, at most, one occurance of the wild card "*". In this case, the non-directory part of the source replaces the "*" in the destination. For example,

COPY sys:Boot>newOS>myprog.Pas dir3>new.*

would copy the file "sys:Boot>newOS>myprog.Pas" into a new file named "dir3>new.myProg.Pas". This is most useful when you want to copy a file from one directory to another with the same name. For example,

COPY dir1>prog.Pas *

copies prog.pas from the directory "dir1" into the current directory.

If there are no wild cards in the source, an attempt to include in the destination name other wild card characters, besides the single "*" discussed above, may lead to problems later. These extra wild cards will be treated as simple literal characters. Because a file name with wild cards in it is hard to specify, the Copy program requires confirmation before creating a file with wild cards in the name.

Copy is a useful command if you'd like to edit your own copy of a file without changing the original one. (The Editor provides you with the ability to do this too, but copying can give you an additional safeguard against unintentional changes to the source.)

The names of SourceFile and DestinationFile may be identical; however if they are identical a new file will not be created. If you want two files you should use nonidentical names.

If the file you are copying to already exists, Copy requests confirmation before overwriting it. This can be disabled by using the switch "/NOCONFIRM" or enabled using the switch "/CONFIRM". NOCONFIRM also sets NOASK. If an error is discovered and wild cards were used, Copy asks the user whether to continue processing the rest of the files that match the input. This confirmation is required no matter what switches were specified.

A final switch is "/HELP" which describes the function of Copy and the various switches.

9. The Preliminary Debugger

The current operating system includes a simple debugger. When the user types ^SHIFT-D or an uncaught exception is discovered, a dump of the user stack is shown. This has the form:

```
Control-shift-D dump
Debug at      108 in routine 7 in IO_PRIVA.
Called from   214 in routine RANDOMWI (8) in WIPEWIN.
Called from   395 in routine WIPEWIN (0) in WIPEWIN.
Called from   149 in routine 0 in LOADER.
Called from   222 in routine 1 in SYSTEM.
Called from   520 in routine 0 in SYSTEM.
```

First, the reason for taking the dump is shown. Next is a trace of all the procedures on the stack. Each line shows the location in the code, the routine that location is in, and the module which contains that routine. The location is the QCode offset from the beginning of the procedure. You can use QDis to try to associate this with the corresponding place in the source. When the debugger can find the procedure name, it is printed followed by the procedure number. At other times, only the routine number will be printed. You can count the procedure (and exception and function) headers in the module source file to determine which procedure it is.

When counting procedures, start with one for programs and zero for modules. The main body of a program is its procedure zero. Exported and forward procedures are counted only once, where the name first appears. Internal (nested) procedures are counted exactly like other procedures.

A note about procedure names: These names are always truncated to eight characters and converted to all uppercase. To get the names, the debugger examines the Seg file for the module. If the Seg file found is not the one that was loaded, the procedure names will be wrong. The procedure numbers will always be correct, however. For the procedure names of system modules, the system run file is checked to find the Seg file name. This Seg file is used to get the procedure names. If the system run file or the system Seg files are not accessible, the debugger will not be able to print the procedure names for system routines.

After the dump is printed, you will be asked if you want to debug. (If a dump is printed due to a ^SHIFT-C, no debugging will be allowed). If you answer no, the program will be continued if it was called from ^SHIFT-D, otherwise it will be aborted and control will go back to the Shell. If you decide to debug, the debugger will print something like:

```
Scrounge, V0.10

  Now at routine KEYINTR (7) in IO_PRIVA
  There are 5 local words, 0 argument words, and 0 result words
```

```
Debug>
```

Now you can use the debugger's commands to investigate your program. Notice that the debugger goes to more effort to find the procedure names than the original dump, so that if a procedure name was not printed at first, going into the debugger may get it displayed.

The debugger does not know the types or sizes of variables, but it does know the number of words allocated for locals, arguments and results. Note that the compiler may generate temporary variables which are included in the local count.

When a debugger command is invoked which takes an offset, you can type a number which is the offset of the word to print. Zero is the first word. If you type -1, all the words in the current context will be printed. For example, to an "a" command, all the arguments will be printed for -1. If you type -2, the debugger will request the first and the last offsets to print. In this manner, a range of values can be printed. No checking is done to make sure that the offsets typed are in range; if a number is out of range, some random data will be printed. Data is printed in the form:

```
[ 7] ( 5053^) = 6
```

where 7 is the offset in the current procedure, 5053 is the offset from the bottom of the entire stack and 6 is the value in that location.

A note on counting variables: Imagine that your procedure was defined with the following variables:

```
    var a,b,c: Integer;
        d: Char;
```

When a list of variables are defined in the same statement, they are allocated in reverse order so the first word is the variable "c". The second word is "b", the third "a" and the fourth is "d". This is true for local and global variables and for records. Note that if you had declared the variables as:

```
    var a: Integer;
        b: Integer;
        c: Integer;
        d: Char;
```

then "a" would be the first word, "b" is the second one, etc. This does not hold for procedure parameters, however, where the variables are always stored in exactly the order declared.

The debugger's commands are:

? Print a list of all the commands.

x Set the radix. All integers are normally printed in signed decimal. With this command, you can specify any radix from 2 to 36. If the radix is negative then all output will be unsigned. If it is positive then output will be signed. Note that this radix is only for output; all input is still in signed decimal.

> Up level. Move up one level in the stack towards the top of the stack. To investigate the variables of a procedure, move up or down the stack until that procedure is reached and then it can be investigated.

< Down level. Move down one level towards the bottom of the stack. When entering the debugger, the current procedure is set at the top so this command has to be used first.

^ Dereference. Dereference any pointer in memory. This takes a segment number and an offset. For a variable parameter or pointer variable, the offset is first and then the segment number.

t Top of Stack. Move to the top of the stack.

b Bottom of Stack. Move to the bottom of the stack.

c Current. Shows number of words for arguments, returns and locals for the current procedure.

d Display Stack. This command reprints the original dump. Some additional procedure names may be printed. In addition, the procedure where you are debugging is marked with "<**>".

l Local. Examine the local data. The debugger reprints the number of local words. You can type the offset of the word you want to see.

a Argument. Examine the arguments to a procedure or function.

e Exception. Examine the arguments (parameters) to an exception.

r Returns. Examine the return values from a procedure.

g Globals. Examine the globals for the module or program that the procedure is in. When the g command is given, the module or program name is printed. If it is a program, the debugger asks if you want to skip input and output. These are the two file variables that are defined for every program and take up space at the top. If you answer yes to this question, you do not have to worry about the space for them when counting variables. Unfortunately, there is no way to examine data in modules that do not have procedures on the stack.

m Mode. The debugger cannot know the type of data, but if you know, you can tell the debugger. The mode command lets you specify the output mode for data. When you type the Mode command, it prints the current mode and asks for a new one. If you type "?" at this point, a list of the options is printed. They are: i=integer, s=string, c=char, B=Boolean, b=byte. Notice the case sensitivity of the arguments. When the mode is string, the debugger cannot print a range or all data since it cannot know how much memory was allocated to hold the first string. In this case, if the -1 argument is

given, the offset is assumed to be zero. When printing strings, the length is printed first. When printing bytes, the radix specified still holds (although they will always be unsigned). For bytes and characters, offsets are still in terms of words; the debugger prints both bytes in the word specified.

s Stack. This command permits display of words anywhere on the program stack. Detailed knowledge of the compiler's memory allocation is necessary to utilize this command so it is generally not useful.

q Quit. Exits the debugger and aborts the program that was running. This returns control back to the Shell. It requires confirmation.

p Proceed. Exits the debugger and resumes the program executing. Note that this command allows you to resume from uncaught exceptions but this is not recommended. In this case, confirmation is required. If the debugger was entered through ^SHIFT-D, then no confirmation is required.

If an exception is raised inside the debugger, the debugger aborts immediately and exits to the Shell. In the debugger, ^C and ^SHIFT-C both cause immediate exit to the Shell also. ^SHIFT-D is disabled while inside the debugger. Also, the HELP key does not work while in the debugger; use the debugger question mark (?) command.

10. Delete

The command:

DELETE FileSpecification[,file2,...filen][/switch]

irrevocably destroys the specified file(s). It deletes the file's name from the directory and places the blocks it occupied on the free list, thus making those blocks available for use by other files.

Wildcards may be used in the file specification, but not in a directory part. See the section entitled "Specifying a file name" in the PERQ Introductory User Manual for details on PERQ wildcard conventions.

The valid switches for Delete are:

/CONFIRM
asks for verification before deleting a file. It's the default when you use a wildcard unless you use a PopUp menu (see below).

/NOCONFIRM
is the default when you specify only a filename without a wildcard.

/HELP
provides some online documentation.

Delete also allows you to select the files you want to delete by using a PopUp menu (see the section on "PopUp Menus" in the PERQ Introductory Users Manual). To get a PopUp menu, type Delete followed by a carriage return (no arguments). Delete will prompt with

File to delete or press for Menu:

at this point you can simply press the pen or puck for a menu. You can also type a file name followed by switches. For example, you might type

:boot>myDir>*.TMP/confirm

and then press down on the pen or puck. You should not type a carriage return before pressing.

The menu displayed when you press contains the files that match the file pattern. If no pattern is typed, all the files in the current directory are listed. Simply select in the menu the files that should be deleted. Unlike the menu for the Shell, the Delete menu allows you to select multiple files. All selected files are marked by reverse-video. These are the files that will be deleted. You can de-select a selected file by simply pressing on it again. Since the number of files that can be displayed is limited, scrolling is

provided when the the number of files matching the file specification is large. Use of the scrolling feature is described in the PERQ Introductory Users Manual.

Once you have selected all the files you wish to delete, move the cursor to the lower right corner of the menu to the spot with the "x" in it. The cursor should change to a large exclamation point. When you press here, all the deletes will take place. If, however, you press outside of the menu before pressing here, no deletes will take place and the program aborts.

When using a PopUp menu for deletion, the default is NOVERIFY. Thus all selected files are deleted when you press the exclamation point. For added safety, you can still specify the /VERIFY switch (as shown above) to cause the system to ask for confirmation on each of the selected files before deleting it.

11. Details

The Details command provides some information about the current state of your PERQ. Typing DETAILS /HELP will get you online documentation. The Details command line is of the form:

DETAILS [/switch]

Valid switches are:

/USERNAME	Name of current user
/USERID	ID of current user
/MEMORYSIZE	The size of memory
/PARITYERRORS	Parity error information
/PROFILENAME	Name of profile file for the current user
/PARTITION	Names of all devices and partitions known (includes the number of free blocks in each partition)
/LOADEDPROFILE	Profile information that is cached in memory
/SEARCH	Prints the current search list.
/SHELLNAME	Name of current Shell runfile
/SHELLINFO	Shell specific information
/DISKSIZE	Size of hard disk
/TIME	Gives current date and time
/PATH	Gives current path, default partition name and default device name
/LASTFILE	Default file for Edit and Compile
/BOOTCHAR	Character used for booting
/BOOTS	Valid boot characters
/SWAP	Whether swapping is enabled or not and to where
/IOERRORS	A count of how many times each of the IO errors occurred since the last boot
/ALL	Displays all of the above information
/HELP	For online documentation

These may be abbreviated to as many characters as are unique. If you don't specify a switch, a selection of the available information is typed. If you specify *, all information is typed.

"Details /Partition" is very useful since it tells how much free space there is in all the partitions.

See the PERQ Introductory User Manual and the PERQ File System Utilities Manual for details on Partitions, Search lists, Shell, Path, and Boots.

12. Directory

The Directory command provides an alphabetical list of files in a directory. You can display the directory listing at your terminal or you can specify that the Directory command write the listing to a file. The form for the command line is:

DIRECTORY [dirSpec>][fileSpec]{/switch}[~][outputfile]

If it is invoked without a switch, all files in the current directory will be listed. To write the output of a Directory listing to a file, specify an output filename.

If there are wild cards, the dirSpec part is matched against all directories and the fileSpec part is matched against all files in the directories that matched dirSpec. Wild cards are described in the section "Specifying a file name" in the PERQ Introductory User Manual. Wild cards are not allowed in partition or device names.

Examples of usage include:

DIR
lists every file in the current directory

DIR *>*
shows all files in all directories starting with the current directory and including all subdirectories.

DIR/HELP
gives online documentation

DIRECTORY :BOOT>x*>*.run~run.list
looks in the Boot partition for all the run files in directories whose names start with "x" and writes all of these names into the file "run.list".

DIRECTORY Program*
tells you what files beginning with "Program" are in the current directory, e.g., Program.pas, Program.seg, Program.run.

DIRECT *zing*
lists all files with "zing" in their names.

DIR Program*/SIZE
lists files beginning with "Program" and tells how much disk space each occupies.

The following are the switches available for use with this program:

/HELP
types online documentation.

/FAST

prints a short directory. This is the default.

/SIZE

tells you how many blocks and bits are in each file.

/ALL

provides the following information about each file:

Number of Blocks
Number of Bits
Kind of file
Creation date
Last Update date
Last Access date

/LISTDIRECTORIES

When doing a multi-directory listing, only the directories that have valid matches for the fileSpec are printed. This switch tells Direct to print all directories that match the dirSpec even if they do not contain any matches for fileSpec.

/ONECOLUMN

tells Direct to print all files in one column. This is the default when the output goes to a file.

/MULTICOLUMN

tells Direct to print files in four columns. This is the default when doing a FAST directory to the screen. This switch does nothing if SIZE or ALL is specified.

/DELIMITER

when used in conjunction with an output file specification, writes filenames as

name | name

into a file. This is useful for creating command files.

/PARTITIONS

gives information about all partitions after the files are listed.

/SORT=option

specifies the method in which the directory

is sorted and displayed. This switch accepts the following options:

NOSORT - do not sort the directory. In this case, all the files are listed in essentially random order. This switch is useful when swapping is turned off and Direct does not have enough memory to sort the file (for example, when running from the floppy).

NAME - sort by the name of the file. This is the default.

SIZE - sort by file size. This operation lists files in decreasing order, with the largest file first.

CREATEDATE - sort by creation date. The most recent file is listed first.

ACCESSDATE - sort by last access. For this function, access is defined as the last read operation performed on the file.

UPDATEDATE - sort by last update. For this switch, update is defined as the last write operation performed on the file.

13. DirTree

DirTree gives a graphic representation of the filesystem's tree structure. It erases the screen and then displays a tree of all the directories starting from the root directory on the left. Any directory can be specified as the root of the tree. The default starting place is the default device. In this case, DirTree displays all the partitions, and then all the directories in each partition, and then all the subdirectories, etc. Lines are drawn from each directory to its parent. If a directory is supplied, DirTree simply starts the search from that directory.

If the specified, or default, root directory contains the current directory, DirTree highlights the current directory with reverse video. You can then change the path by moving the cursor to the desired directory and pressing the pen or puck. This is equivalent to issuing a Path command; DirTree permits you to change your current directory.

Typing any character or pressing in an area that does not contain a directory exits DirTree.

If the directory is too deep to fit on the screen, DirTree puts an asterisk (*) on the right of the parent. You can reinvoke DirTree with this directory as the root to see more of the tree structure.

DirTree accepts three switches: /WAIT, the default, enables pressing to select a new path; /NOWAIT disables pressing to select a new path (DirTree simply displays the tree structure); and /HELP for online documentation.

14. Dismount

The Dismount command detaches devices from the filesystem. The argument to Dismount is HARDDISK (H) or FLOPPY (F). Note that these names are used no matter what name the device was given when it was partitioned.

Once a device has been dismounted, it can no longer be accessed until it has been mounted again. It is very important to Dismount filesystem floppies before removing them from the drive.

See the section on "Mount" for more information.

15. Editor

The Editor is used to create or alter any text file on the PERQ. You will probably use it very frequently, so it's a good idea to become familiar with it as soon as you can. It has its own online documentation (run Edit and press the HELP key) and manual (The Editor User's Guide), so this description is brief. Three common uses of the editor are discussed here. The command line for Edit is:

EDITOR FileSpecification

or

EDITOR/replay

If the switch is left out Edit will assume that you want to edit the default file name remembered by the Shell. The /REPLAY switch is useful when disaster occurs during an edit session; you specify the switch, redo the edit session, and stop just before the disaster. Refer to the Editor User's Guide for details.

The Editor does extension completion on the file name specified. If the file to edit is FOO.PAS, it is only necessary to type FOO. The extensions that the Editor knows about, in order tried, are: Pas, Micro, Cmd, and Dfs.

The Editor signals the end of a file with a solid, left pointing triangle.

Three common uses of the Editor are:

1. To create a new file: Invoke the editor with the name of your new file. The screen is cleared to give you a blank page to write on. Type I to insert the text that you want to type in. When you're finished,

   i. Press the INS key (upper lefthand corner of keyboard). This is important; it's the only way that what you typed in will be saved.

   ii. Type Q. The screen is cleared again and you are prompted with a list of alternatives. See the Editor documentation for details on these.

2. To make changes to an existing file: Invoke the editor with the name of an existing file. Work with the Editor User's Guide at your side until you're comfortable with the functions available.

   If you find that you've made changes to a file that isn't yours or that you've done irreparable damage to one that is, don't panic - if, after typing Q you type E, all of the changes you've made will be ignored.

If you'd like to save your changes but don't want to alter the source file, you can type W after Q to make a new file.

3. To read a file at your leisure: you can EDIT it, reading and scrolling at your own pace. To safeguard against your having made any accidental changes to the file, type E after Q.

16. ExpandTabs

ExpandTabs simulates tabs in every 8th column by replacing tabs in the input file with the correct number of spaces. ExpandTabs is used when the input file was written for another system and put onto a PERQ, which does not support tabs. Its command line takes the form:

EXPANDTABS SourceFile DestinationFile

Note that the ExpandTabs command does not accept the Help switch.

17. FindString

The FindString command searches through a number of files for a particular string. There are two modes: /CONTEXT; and /NOCONTEXT. In /CONTEXT mode, FindString prints leading and trailing characters for each occurrence of the specified string. In /NOCONTEXT mode, FindString prints only the first occurrence of the specified string and does not print leading or trailing characters. The default mode is to print leading and trailing characters for each occurrence (/CONTEXT).

The first argument to FindString is the string to search for. If you want to include a space, comma (,), or slash (/) in the search string, you must precede it with a single quote ('). The next argument is the file pattern to match files against. The remaining arguments are optional. You can direct FindString to write the occurrence(s) to a file by specifying an output file. You can also specify a switch to show context, ignore upper and lower case, or request help. Thus, an example command line is:

```
FindString screen, :boot>os>*.pas~screen.users/nocontext
```

The above command directs FindString to search all files with a .PAS extension in the OS directory of the BOOT partition for an occurrence of the string screen. FindString writes the output to the file "screen.users". By default, case is not significant (in the example above, Screen matches screen). You can force FindString to match upper case characters by specifying the /CASESENSITIVE switch. FindString also accepts the /HELP switch.

18. Floppy

The FLOPPY utility formats, tests, reads, and writes RT-11 format and filesystem floppy disks. FLOPPY can also format filesystem floppies. You can use FLOPPY to transfer files between the hard disk and the floppy disk. The command line is:

FLOPPY [command][/switch(es)]

To execute a single FLOPPY command and return control to the Shell, enter a command on the command line.

To execute multiple FLOPPY functions, type FLOPPY and press return. In this case, the utility prompts with FLOPPY>. You can then enter commands or use the pop-up menu.

Some FLOPPY commands require confirmation. This confirmation comes from the keyboard even if a user command file is in use. If you use the FAST command (see below), no confirmation is required. The Zero and Format commands, however, require an explicit /NOCONFIRM switch to override the confirmation request.

While FLOPPY is processing a command, a "hand" cursor moves down the right margin of the screen. When it has reached the bottom, your operation is complete. (With small files, the cursor does not reach the bottom of the screen.)

The wild card handling in FLOPPY is more restrictive than the operating system's. FLOPPY only accepts wildcards for the DELETE and DIRECTORY commands (see the respective command descriptions). In addition, the only wild card available is the asterisk (*) and it can only be used by itself in either the filename or extension part of a file specification (or both).

Current FLOPPY commands and their switches are:

COMPRESS: Coalesce free space on the floppy. This moves files so that all the unused blocks are at the end of the floppy. COMPRESS does not accept input or output arguments, but has a switch which turns verification on or off. The default is verify; if this is on, COMPRESS checks every transfer to assure there are no errors. To COMPRESS in verify mode, you need not specify the switch, since /VERIFY is the default. The /NOVERIFY switch overrides the default.
NOTE: This command takes a long time and cannot be interrupted once it starts.

DELETE: This command deletes a file or multiple files on the floppy. To delete multiple

files, you must separate the filenames with a comma (,). By default, the DELETE command requests confirmation before deleting a file. You can override this by specifying the /NOCONFIRM switch. Wildcards are allowed.

DIRECTORY: This command lists the files contained on a floppy and optionally writes the directory listing to a file. If you specify the DIRECTORY command with no arguments, it lists all the files contained on the floppy. If you specify an input argument, DIRECTORY lists those files that match the specified filename. If you specify an output argument, DIRECTORY writes the listing to a disk file with that name, but does not display the filenames on the screen. By default, the DIRECTORY command prints (or writes) a full listing. To override this and print only the file names, specify the /SHORT switch.

DUPLICATE: This command copies the contents of one floppy onto another. The command creates a set of scratch files on the hard disk, copies the scratch files back to the new floppy, and then deletes the scratch files from the hard disk. To retain the scratch files on the hard disk, specify the /NODELETE switch. By default, the Duplicate command creates a double-sided floppy. To override this and create a single-sided floppy, you must specify the /SINGLESIDED switch. Remember that the blank floppy to be duplicated must have been formatted prior to invoking Duplicate.

FLOPPYGET: This command copies the contents of a floppy disk to the hard disk. You can optionally specify a hard disk file name. The Floppyget command accepts the /SINGLESIDED switch to permit you to specify a single-sided floppy.

FLOPPYPUT: This command copies the disk files created by Floppyget onto a floppy disk. By default, Floppyput deletes the disk files after copying them to the floppy. You can specify the /NODELETE switch to override this action. For single-sided floppies, you must use the /SINGLESIDED switch.

FAST: When issued as a command, FAST turns off requests for confirmation for all subsequent commands except Format and Zero. Use

the FAST command only in conjunction with command files.

FORMAT: This command formats a floppy disk and destroys its current contents. The FORMAT command accepts the following switches:

/DblDensity=Yes or No (default is no)
/Noconfirm (override request)
/Singlesided=Yes or No (default is no)
/Test=Yes or No (default is no)

GET: This command copies one or more floppy files to the hard disk. In its simplest form, you specify only an input filename to copy from the floppy to the hard disk. GET then copies the floppy file to the hard disk using the same filename. If the filename already exists on the hard disk, GET requests confirmation before overwriting it. Other forms of the command permit you to name the hard disk file. For example:

```
GET floppyfile~harddiskfile
```

To specify multiple files, use the following construct:

```
GET f1,f2,...fn~h1,h2,...hn
```

Like the COMPRESS command, GET takes the /VERIFY and /NOVERIFY switches. The default is /VERIFY.

The GET command requires confirmation before overwriting a file on the hard disk. You can specify the /NOCONFIRM switch to override this action.

HELP: When issued as a command, HELP provides general information for the FLOPPY utility. You can get specific help by using the /HELP switch. Note that HELP and /HELP override any other commands or switches; FLOPPY displays the requested help and then reprompts.

PUT: This command parallels the GET command, but transfers a file or files from the hard disk to the floppy disk. PUT also requires

confirmation before overwriting an existing file on the floppy (override this with /NOCONFIRM) and, like GET, accepts the /VERIFY and /NOVERIFY switches.

QUIT: Exits the FLOPPY utility.

RENAME: Changes the name of a floppy file. If the new filename already exists, RENAME asks for confirmation before overwriting it. You can override this with the /NOCONFIRM switch.

DENSITY: This command tells the user whether the floppy is single or double density.

TYPE: This command types a floppy file on the screen. The "hand" cursor tells how much of file has been typed. When the Type command finds a formfeed character (^L) in the file, it waits after displaying a screenful of text. This enables the user to read the page before the screen is erased and the next page displayed. To continue reading, type ^Q. You can disable this wait feature by specifying the /NOWAIT switch. The Type command displays a solid, left pointing triangle when it reaches the end of the file.

COMPARE: This command takes a disk file and a floppy file (in that order) and ensures that all bytes are identical. You can specify multiple disk files (input arguments) and multiple floppy files (output arguments) for the comparison. If you specify multiple arguments, you must separate like arguments by a comma (,) and delimit input form output arguments with an equal sign (=). COMPARE prints a message for every block that contains a difference.

ZERO: This command creates a new directory on the floppy. By default, the directory matches the number of sides on the floppy. You can override this by specifying the /SINGLE-SIDED switch. The Zero command always requests confirmation before creating the directory. You can override this only by specifying the /NOCONFIRM switch. In the process of creating the new directory, the ZERO command destroys the contents of the current floppy. Use the ZERO command after formatting a floppy (see the FLOPPY FORMAT command description in this section) or whenever you wish to destroy the current

content of a floppy.

Note that when you issue the ZERO command, you irrevocably destroy the current contents.

19. FTP

The FTP utility copies files across the RS232 link to another PERQ or any other computer that supports the FTP protocol. You can also use FTP to transfer files across an ETHERNET connection. The command line is:

FTP [command][/switch(es)]

To use the RS232 line, the two machines must be connected by an RS232 cable, which goes into their RS232 ports (located between the cables for the display and keyboard in the lower lefthand corner of the back of the PERQ).

To use an ETHERNET connection, you must first assign the Ethernet addresses in the file SYS:BOOT>ETHERNET.NAMES. The format of this file is the node name followed by a unique six byte address. The first three bytes are the Ethernet address blocks for Three Rivers Computer Corporation (02 1C 7C in hexidecimal) and the second three bytes identify the individual nodes. You can include up to ten nodes in the ETHERNET.NAMES file. The basic, and recommended format follows:

```
PERQ1     540  31744  1
PERQ2     540  31744  2
      .
      .
      .
PERQ10    540  31744  10
```

You can then issue the FTP Address command and specify the local and remote names. If the specified names exist in ETHERNET.NAMES, the connection completes.

Once the connection or addresses are established, each machine must run the FTP program. Its prompt is FTP>. You can enter commands directly to FTP or use the pop-up menu.

The transfer may be performed with either machine taking the active role; the alternate scenarios are described below:

1. PERQ #1 takes the active role and PERQ #2 is passive. They'll follow this script:

   PERQ #2: Runs FTP and starts polling
   PERQ #1: Runs FTP and then issues the next command
   PERQ #1: PUT SourceFile[~][DestinationFile]
   Both PERQs: #!#!#!#!#!...#!#!
   (this appears on the screen as the file is being transferred and a "hand cursor" moves from top to bottom to show percentage complete. When the hand cursor reaches the bottom, the transfer

is complete.)

2. PERQ #2 has the active role:

```
PERQ #1: Runs FTP and starts polling
PERQ #2: Runs FTP and then issues the next command
PERQ #2: GET DestinationFile[~][SourceFile]
Both PERQs: #!#!#!#!...#!
            (same as above)
```

The passive PERQ polls while the active machine processes the transfer.

The valid commands for FTP are:

```
GET  SourceFile[~][DestinationFile]
PUT  SourceFile[~][DestinationFile]
MODE mode (can be PERQ, VAX-11, or PDP-11;
           default is PERQ)
BAUD baud rate (initially set to 9600)
DEVICE line (either RS232 or ETHERNET)
ADDRESS localname[~]remotename
```

You can specify HELP whenever FTP requests input.

You can call FTP with one command line. For example, you can specify:

```
FTP GET DestinationFile
```

Control is returned to the command interpreter Shell after that one command is executed. Additionally, you can specify the MODE, BAUD, and DEVICE commands as switches to the GET or PUT commands. For example:

```
FTP GET file/Mode=PDP-11/BAUD=1200/DEVICE=RS232
```

The FTP command also accepts a switch that permits you to specify whether or not the file to transfer is a text file; specify /TEXT for text files and /BINARY for other files.

Sometimes the BitPad and pen can cause problems with file transfers. If you encounter any problems, unplug the pen or move it away from the BitPad.

If an FTP transfer fails, the current transfer aborts and FTP waits for another command (or exits if invoked with a command line). However, if FTP was invoked from a command file and the transfer fails, it restarts the transfer and repeatedly tries again until it succeeds.

20. Help

The Help command is used to display helpful information about other commands and the PERQ operating system. When issued without an argument, a message describing the PERQ and the use of PopUp menus is displayed. When an argument is included, information about the program specified by the argument will be displayed. The HELP key or the Help command without an argument gets you to the help utility, if available. The help utility erases the screen and then prints a list of all commands for which help exists. Type one of these names (or use the pop-up menu to specify one). The specific help is then printed for that command.

21. Linker

The Linker takes compiler-generated .Seg files as its input and produces a runfile with a .Run extension. The runfile is created by linking together all of the separately compiled modules that make up a program. The command line takes the form:

```
LINK Source{,Source}[~runfile]{/switch}
```

Examples of valid command lines include

```
LINK Program
```
where Program.Seg is the output of a compilation

```
LINK Program1, Program2
```
where Program1.seg and Program2.seg are compiler output files. The output will be Program1.Run.

```
LINK Program1, Program2~Program0/VERBOSE
```
where Program1.Seg and Program2.Seg are compiled .Seg files and Program0.run is to be the runfile. The VERBOSE switch specifies that the name of each import module is to be listed.

Any number of source files can be listed separated by commas. The switches currently available are:

/MAP=name	-creates a map file. If you don't specify a name after the equal sign, the MAP file will have the same name as the runfile with a .Map extension. A map file contains a listing showing placement and size of the code and data segments of the program as well as other linkage details. Thus, map files are useful in debugging user programs.
/USER	-this is the default. It specifies that the program being linked is a user program. The opposite is /SYSTEM.
/VERBOSE	-lists the imports of each module as the module is being processed.
/SYSTEM	-specifies that this is a system program.
/VERSION=nn	-links the program with the version of the system specified by "nn". If the run file name ends in a number (e.g., "LOGIN.42"), then the Linker uses the number as the version number. This can be overridden by using /VERSION=nn switch.

A word about versions. Every system has a version number. When you create a new system, you also should use a new system version

number. You then have to relink all the runfiles. This insures that the programs execute correctly with the new system. In order to prevent confusion, we have put the system version number into the name of certain critical system runfiles: Link, Shell, Login, and System (e.g. Login.5.run). When you link a file, it looks up the system run file based on the version in use. If no version is specified, it uses the current system version (which is displayed at the top of the screen when in the Shell). If a version is specified, a system run file with that number is used to resolve references to system routines. If the /SYSTEM switch is used, however, no run file is looked for since a new system run file is created. The syntax for creating a system run file is:

```
LINK SYSTEM~SYSTEM.NewNumber/SYSTEM/VERSION=NewNumber
```

where the /VERSION switch is optional.

To create runfiles for a new system, you link using the VERSION switch after making the system runfile. It's done like this:

```
LINK FileSpecification/VERSION=NewNumber
```

22. Login

Login initiates a user's session at a PERQ. It is called automatically when the PERQ is booted or when a user logs off using the BYE command. It can also be called explicitly by a user.

At boot time, Login asks for the date, time, your name, and password. When you invoke Login, it asks for your name and password. Login prompts you with the format it wants these in.

It can be called explicitly with the command line:

```
LOGIN [UserName][/switch(es)]
```

and will prompt you with the information it wants.

Login searches the System.Users file for the given user-name and validates the password. The UserControl program (see below) maintains the System.Users file. If the user has a profile, it is read to find parameters to set up the PERQ for the user.

The Login command line may include switches. You can also include Login switches in a #Login section of your profile. The following describes the valid Login switches:

/PATH=pathname

Sets the default path to pathname. The /PATH switch is not cumulative, if you specify the switch multiple times, only the last one has an effect.

/SETSEARCH=pathname

Pushes pathname onto the search list. The effect of this switch is cumulative; if you specify the switch multiple times, you add additional items to the search list. If the argument is a minus sign (-), the last path is popped from the list. Note that the last item specified is the head of the searchlist and the first item specified is the end of the list.

/CURSORFUNCTION=n

Sets the default cursor function. Since the cursor function determines the screen color also, this switch can be used to set the default screen color.

Valid cursor functions are the integers 0 through 7 inclusive.
The integers signify the following:

0 - screen is all white

1 - only cursor displays
2 - white on black, cursor is large square
3 - black on white, cursor is large square
4 - black on white, black cursor hides image
5 - white on black, white cursor hides image
6 - black on white, cursor inverts
7 - white on black, cursor inverts

If you like white letters on a black background, use CURSORFUNCTION 5.

The default value is 4.

/SCREENBOTTOM=option

Sets the default parameters for the bottom of the screen when you change the screen size with a ScreenSize command (see below).

Valid options for this command are:

ON - displays bits stored in area
OFF - bottom is all one color
WHITE - bottom matches background
BLACK - bottom is opposite color of background

/COMMAND=string

Execute the specified string as the first Shell command after login and before accepting commands from the keyboard.

/SHELL=filename

Specify an alternate command interpreter to run instead of the Shell. This is an aid in debugging a new or user written command line interpreter.

/PROFILE=filename

Specifies a file (filename) to execute as the profile file.

/HELP

Displays a brief explanation of the Login Utility.

A sample profile for login might be:

```
#Login  /Path=Sys:User>MyName>
        /SetSearch=Sys:Boot>Library>
        /SetSearch=Sys:Part3>Alt7>
        /CursorFunction=4
```

Note that the command names in the profile can be abbreviated. Additional documentation for Login can be found in the section "Turning on the PERQ" in the PERQ Introductory User Manual.

23. MakeBoot

MakeBoot writes boot files onto hard disk and floppy disk. It is self-prompting. Its command line is:

MAKEBOOT [RunFile]

See the section "Booting the Machine" in the PERQ Introductory User Manual for more details on booting. The manual How to Make a New System gives complete description of the MakeBoot program.

24. MakeDir

MakeDir creates a new empty directory. Its command line is:

```
MAKEDIR [FileSpecification]
```

where FileSpecification is the name for the new directory. If you do not specify the name you will be prompted for it. All directories have a .Dr extension; MakeDir will provide this automatically if you do not specify it.

MakeDir will not accept as its parameter the name ROOT or the name of any extant file that has a .Dr extension; it will print an error message if you pass it such a name.

See the PERQ Introductory User Manual for more information on files and directories.

25. Mount

The Mount command is used to attach devices to the filesystem. Devices must be mounted before files on them can be accessed by the fileSystem. The two Mount commands are:

```
MOUNT H[ARDDISK]
```

and

```
MOUNT F[LOPPY]
```

Note that these are the forms to use no matter what the name devices were given when partitioned. The device that was booted from is mounted automatically by the system.

Notes on Mount and Dismount:

1. It is imperative that you dismount FileSystem (as opposed to FLOPPY) floppies before you remove them from the floppy drive. If you fail to do this, the next floppy put into the drive may be overwritten and the fileSystem floppy is also likely to be corrupted.

2. Floppies written with the FLOPPY program cannot be mounted or dismounted.

3. A device may be mounted more than once.

4. It's a bad idea to dismount the device you're running from. If you do, the system will not be able to find the Shell.

26. ODTPRQ

ODTPRQ is a simple debugger for microcode and new operating systems. ODTPRQ runs on a PERQ other than the one that is being debugged. It communicates through a Link board that is plugged into the I/O option slot in the card cage. ODTPRQ has an online help facility. The ODTPRQ command line is:

```
ODTPRQ [StateFileName]
```

where StateFileName is the name of a State File.

27. Partition

Partition creates new partitions on a device (such as hard disk or floppy). It can also be used to modify the names and sizes of existing partitions. Creating a new partition destroys all old data in the area where the partition is made. The device should first be formatted before using Partition. After formatting, Partition is the first program to run. Its command line is:

```
PARTITION
```

The program is self-prompting. Refer to "Partitions and the Partition Program" in PERQ File System Utilities Manual before using it.

28. Patch

Patch allows you to examine and modify the contents of a disk file. To run it, type:

```
PATCH [filename]
```

If you have not specified a filename or if the specified file does not exist, PATCH prompts for the filename. When it has a valid file, it prompts with:

```
Read Block [0]?
```

Type return to look at the block number in brackets or type a new block number. You can also type HELP for some online documentation that describes the commands you can use.

When you ask to read a block, Patch displays it byte by byte or word by word on your screen in 32-rows. You can reference each byte with the indices 0 to 511. Patch permits you to make temporary or permanent changes to your file.

To access all hard disk blocks, you can patch the SYS: file.

29. Path

Your "path" is the directory you are currently using. To find named files, the system will first look in this directory and then in directories specified in your search list. (See SetSearch below). New files are created in this directory if a different one is not specified. The Path command changes the current directory. The command line is:

PATH [pathname]

Path does not affect the search list. If Path is called without an argument, it prints the current path. A new path can then be typed or Carriage Return to exit without changing the path. The final ">" of a directory name is optional for the path command so "Path Foo" changes the path to the directory foo.Dr. Path will print an error message if you try to Path to a nonexistant directory. If you attempt to set a Path from which the Shell cannot be accessed (which can happen if you change the SearchList), Path requires confirmation.

30. Pause

Pause can be used to suspend the execution of a command file and wait for user confirmation before proceeding. It takes a message as a parameter, prints it on the screen, and then waits for a carriage return on the keyboard before continuing. This command can be given by the user, but it is most useful in command files when some user action is required before proceeding, e.g., changing floppies.

31. PERQ.Files

Part of the documentation for each PERQ operating system is a file called PERQ.Files which describes all the files in the system and states which part of the system they are in. The PERQ.Files program is used to list portions of the PERQ.Files text file and to make command files. Run the program by typing:

```
PERQ.Files
```

and type "Help" when you are prompted. The program types out a comprehensive description on how to use the program.

32. Print

Print sends files to a printer (the GPIB address for Print is one). If the file does not begin with a form-feed character, Print supplies it. Also, Print supplies a form-feed at the end of every printed file. Its command line is:

```
PRINT [Filename][,filename2,...filenamen][/switch(es)]
```

For printers connected to the RS232 port, Print requires Z80 PROMs version 8.5 (or higher) for proper operation. You can find these two PROMs on the "IO" board (red color-coded) labeled with the version number. For printers connected to the GPIB (IEEE-488) port, the PROM versions are not relevant. Contact Three Rivers field service to exchange earlier PROMs.

The Print command accepts the following switches:

/TALL	tall characters (six lines per inch)
/SHORT	short characters (eight lines per inch) /SHORT is the default
/WIDE	wide characters (10 characters per inch)
/NARROW	narrow characters (16.5 characters per inch) /NARROW is the default
/SHIFT=n	shifts the listing n spaces to the right /SHIFT=0 is the default
/COPIES=n	specifies the number of listings /COPIES=1 is the default
/START=n	specifies the page number of the document to begin printing /START=1 is the default
/STOP=n	specifies the last page number of the document to print /STOP=lastpagenum is the default
/BAUD=n	sets the baud rate for printing /BAUD=9600 is the default
/TABS=n	tab stops every n characters /TABS=8 is the default
/TITLE	prints a title line at the top of each page /TITLE is the default for files with other than a DOC extension
/NOTITLE	omits the title line /NOTITLE is the default for *.DOC files
/BREAK	places a blank page between each page of the listing
/NOBREAK	omits the blank page /NOBREAK is the default
/HP	initializes to use the Hewlett-Packard 7310A
/LINEPRINTER	initializes to use the TI 810 (or similar) printer
/DIABLO	initializes to use the Diablo 630 daisy printer
/PLAIN	no initialization; use with generic printers /PLAIN is the default
/HELP	supplies online documentation

33.PRQMic

PRQMic is the PERQ microcode assembler. It takes a microcode source program (whose extension is .MICRO), and produces output that can be used as input to the microplacer, PRQPlace. The PERQ Microprogrammers Guide has details on how to write microprograms and how to use PRQMic.

34.PRQPlace

PRQPlace is the PERQ microcode placer. It takes the output from the microassembler, PRQMic, and produces a file (with extension .BIN) that can be loaded into the PERQ microstore. The PERQ Microprogrammers Guide has details on the use of PRQPlace.

35. QDis

QDis is a disassembler for Q-Code. It decodes a .Seg file into Q-code. The command line:

QDIS

causes the following sequence:

1. The program identifies itself as QCode Disassembler.

2. You are then prompted for an input file. Type the name of the program or module you want disassembled. (Since the input to QDis is a .Seg file, the input file must have been compiled.) You needn't specify .Seg but must specify extension if it is anything else.

3. Next, you are prompted for an output file. The default is CONSOLE: .

4. QDis displays a list of the program's routines and the following information about each:

   Routine name
   Routine number
   Lexical level
   Parameter size
   Result + Parameter size
   Local + Temporary size
   Entry address
   Exit address

5. QDis next will ask you which routine you'd like to see disassembled. Type in a routine number; the program then types a listing of that routine's Q-code translation.

Step 5 can be repeated indefinitely.

36. Rename

Typing the command line:

RENAME SourceFile[~]DestinationFile

will change the name of SourceFile to DestinationFile. You can rename a file from one directory to another (move the file) as long as both directories are in the same partition.

Rename prompts for any missing arguments.

You may rename a .Run file, but if you rename a .Seg file you must re-link the programs which use it.

The source file for Rename may contain wild cards (for a description of the wild cards, see the "PERQ Introductory User Manual"). If the source contains wild cards, the destination must contain the same wild cards in the same order. In this case, Rename finds all files in the directory which match the source pattern. For these files, the part of the source file name that matched each wild card is used to replace the corresponding wild card in the destination. As an example, for the command:

RENAME foo*.abc# anotherdir>*baz.rmn#z

The input file "FOOZAP.ABCD" would be renamed to the new file "anotherdir>ZAPbaz.rmnDz".

If wild cards are used, Rename asks for verification of each file renamed. This can be disabled with the switch "/NOASK" or enabled with the switch "/ASK." The default is enabled.

Wild cards are not allowed in the directory part of the source file.

When the source file name contains no wild cards, the destination file name may contain, at most, one occurance of the wild card "*". In this case, the non-directory part of the source replaces the "*" in the destination. For example,

RENAME sys:Boot>newOS>myprog.Pas dir3>new.*

would rename the file "sys:Boot>newOS>myprog.Pas" to a new file named "dir3>new.myProg.Pas". This is most useful when you want to rename a file from one directory to another with the same name. For example,

RENAME dir1>prog.Pas *

moves prog.pas from the directory "dir1" into the current directory.

If there are no wild cards in the source, an attempt to include in the destination name other wild card characters, besides the single "*" discussed above, may lead to problems later. These extra wild

cards will be treated as simple literal characters. Because a file name with wild cards in it is hard to specify, Rename requires confirmation before creating a file with wild cards in the name.

If the the destination file already exists, Rename requests confirmation before deleting. This can be disabled by using the switch "/NOCONFIRM" or enabled using the switch "/CONFIRM". /NOCONFIRM also sets /NOASK. If an error is discovered and wild cards were used, Rename asks the user whether to continue processing the rest of the files that match the input. This confirmation is required no matter what switches were specified.

A final switch is "/HELP" which describes the function of Rename and the various switches.

37. ReRun

ReRun is a convenient method to reexecute the default file remembered by the Shell and supply new arguments to the runfile. The command line is:

```
ReRun arguments
```

The ReRun command is most useful when the default file has an especially long name. For example, if "Programwithlongname" is the default file, the command:

```
ReRun A b 3
```

is the same as typing:

```
Programwithlongname A b 3
```

38. Run

Run is another way to invoke the default file remembered by the Shell. If you have been editing, compiling and linking the default file, simply typing:

```
RUN
```

executes that default file. The Run command sets the default file to the specified runfile. For example, the command:

```
RUN Foo arg1 arg2
```

is the same as typing:

```
Foo arg1 arg2
```

except that the first command (Run Foo arg1 arg2) sets the default file to Foo.

39. ScreenSize

The display on the PERQ screen is stored in memory and requires 48K words (192 blocks). Sometimes it is advisable to give up some of the screen and allow this memory to be used for storing data or programs. For example, the Scavenger runs with swapping off so it shrinks the screen to allow its data and code to fit into memory. Even when swapping is enabled, some programs, such as the compiler, want to trade speed for screen size.

The ScreenSize command prompts for the number of scan lines used in the display for the next program run. The screen expands to full size after that program has completed. The full screen has 1024 lines in it. The number you supply must be greater than zero, less than or equal to 1024, and a multiple of 128 (e.g. 128, 512, 768, etc.). You can specify a number in the range 1 through 8 and ScreenSize will multiply the number by 128. For example, if you specify the value 4, ScreenSize multiplies 4 by 128 and uses 512 as the number of scan lines (512 is half of the screen).

ScreenSize accepts four switches to permit you to control the bottom portion of the display. The switches are:

/ON

This switch permits you to see the data or code that is stored in the screen area. The memory manager is told that the memory is free, but the IO package still thinks it should be displayed. The screen package will not let you create a window or write into that area (unless a program calls RasterOp or Line directly). This switch is the default condition, unless an entry in the user profile has changed it (see Login above).

/OFF

This switch forces the bottom of the screen area to a solid color.

/WHITE

This switch forces the bottom of the screen area to the same color as the usable part of the screen.

/BLACK

This switch forces the bottom of the screen area to the opposite color of the general background.

The Shell shrinks the screen automatically for certain programs. If

you explicitly call ScreenSize before these programs are run, then the Shell will not override your settings.

40. Scavenger

The Scavenger checks the filesystem's structures and fixes any errors found. It can be called any time you suspect a filesystem problem or whenever you need to reconstruct a directory. It is self-prompting.

Its command line is:

SCAVENGE

For more details on this program, see the section "Scavenger Program" in the PERQ File System Utilities Manual.

41. SetBaud

SetBaud allows you to specify the baud rate to the RS232 line. Valid baud rates are: 110; 150; 300; 600; 1200; 2400; 4800; and 9600. The command syntax is:

```
SetBaud 4800
```

42. SetSearch

SetSearch allows you to add and remove paths to search lists and to change their order. Its command line is:

```
SETSEARCH [pathname][,pathname]
```

If you do not specify a pathname, SetSearch displays the current search list and then permits you to change it or simply exit.

If you specify a file name, SetSearch pushes that name onto the search list; if it is a hyphen (-), the first item on the list is popped off.

The current search list is 5 items deep with the first and last slots reserved. SetSearch gives you a warning if you attempt to push something onto the first slot or pop something off the last. In addition, when you try to exit, SetSearch checks to make sure that the Shell can be found with the new search list and path. If not, you are asked for confirmation before exiting.

43. SetTime

The SetTime command allows you to specify the date and time. Specify the time and date as follows:

```
DD MMM YY  HH:MM:SS
```

The time notation uses a 24 hour clock and the seconds are optional. An example command line which sets the date to June 3, 1955 and the time to 3:30 PM folllows:

```
SetTime 3 Jun 55  15:30
```

SetTime permits you to change only the time; if you omit the date, you correct the time.

44. Statistics

The operating system constantly collects statistics about the performance of the swapper. To display the statistics after each execution completes, specify the command:

```
STATISTICS Yes
```

To end the display, type the command:

```
STATISTICS No
```

After each program is executed, the Shell prints the Statistics for that program in the following format:

```
Load      1.6 secs.
Exec      3.6 secs.
IO        0.8 secs.
Swap      0.1 secs.
Move      0.0 secs.
Duty     97.2 percent.
```

Load is the time spent loading the program and its modules into memory. Exec is the total time spent executing including IO, Swap and Move times. IO is the time spent doing Unit-level IO not including IO time spent swapping. Swap is the amount of time spent swapping. Move is the amount of time spent moving segments from one place in memory to another to try to find room for new allocation. Duty factor is the proportion of time spent in actual work. It is computed in the following way:

```
100 * (Exec - Swap - Move) / Exec
```

45. Swap

The Swap command enables or disables the virtual memory system. The command "SWAP NO" turns virtual memory off, and swaps all active segments into memory. "SWAP YES" turns swapping on. In this case, you may specify a partition to use for the swap files. For example,

```
SWAP YES Sys:Boot>
```

enables swapping to the Sys:Boot> partition. When booting from the hard disk, swapping is initially enabled in the partition containing the boot file. If you simply specify SWAP YES with no partition name, this default partition is used. When booting from the floppy, swapping is initially disabled. We discourage swapping to a floppy disk because floppies have a small capacity, are slow, and may be dismounted at any time.

46. TypeFile

Type displays any file or files on the console. The command line:

TYPEFILE FileSpecification

displays a single file on the console. To display multiple files, sequentially, separate the file specifications with a comma (,).

When the TypeFile command finds a formfeed character (^L) in the file or when it gets to the bottom of the screen, it waits after displaying a screenful of text. This permits the user to read the page before the screen is erased and the next page is displayed. To continue, type ^Q. This waiting can be disabled by using the /NOWAIT switch. The /WAIT switch, which is the default, causes the waiting to happen. The final switch available is /HELP, which displays online documentation.

The TypeFile command displays a solid, left pointing triangle when it reaches the end of the file.

TypeFile does extension completion on the file name specified. If the file to print is FOO.PAS, it is only necessary to type FOO. The extensions that type knows about, in order tried, are: Pas, Micro, Cmd, Dfs, and Doc.

If no filename is supplied on the command line, TypeFile will display the default file name. This is the same default file used by the Editor, Linker, Run, etc. Unlike these programs, however, TypeFile does not change the default file name.

47. UserControl

UserControl is used to maintain the System.Users file which contains information about valid users and their passwords, group identifiers, and user profiles. This information is used by the Login program (see above). The System.Users file must be in the root directory (top level) of the partition where the boot file is. Its command line takes the form:

USERCONTROL [command][/switch(es)]

If you include a command, UserControl executes the command and then exits to the Shell. If you do not include a command, UserControl prompts with:

UC>

and waits for input.

The following are valid commands:

HELP

Provides online documentation.

ADDUSER [username][/switch(es)]

Adds a new user to the user file or, updates the information for an existing user. If you omit a username, the command prompts for one.

Any printing character except blanks, spaces, equal signs (=), commas (,), or slashes (/) are valid usernames. The maximum number of characters is 31. The command accepts the following switches:

/PASS - permits an existing user to change his password or enters a password for a new user. Note that when you specify the /PASS switch, UserControl prompts for the password. You can enter any printing character except blanks, spaces, equal signs (=), commas (,), or slashes (/). The maximum number of characters is 31. You can also respond to the password prompt with CR. This causes a null password to be entered for the user.

If you omit the switch for a new user, the password defaults to null. For an existing user, the password is unchanged if the switch is omitted.

/GROUP=[group id] - permits an existing user to change the group id or enters a group id for a new user. The value for group id can be any integer from 0 to 255 inclusive. If you do not specify a group id, you are prompted for a value.

If you omit the switch for a new user, the group id defaults to 1. For an existing user, group id is unchanged if the switch is omitted.

/PROF=[profile] - specifies the complete path for the user's profile file.

If you omit the switch for a new user, the default is SYS:USER>name>PROFILE. For an existing user, the profile is unchanged if you omit the switch.

CHECKUSER [username]

Validates a username and password. If you omit the username, the command prompts for one.

NEWFILE

Destroys the existing System.Users file and creates a new System.Users file.

LISTUSERS

Displays a list of all of the valid users.

REMOVEUSER [username]

Delete a user from the file. If you omit the username, the command prompts for one.

QUIT

Exits the program.

You can also specify the /HELP switch (type /HELP or press the HELP key) with any command or in response to any prompt to display specific help information.

How to Make a New System

John P. Strait

This manual describes how to change modules contained in the Three Rivers PERQ Operating System and how to create a new version of that system.

Three Rivers Computer Corporation
720 Gross Street
Pittsburgh, PA 15224
(412) 621-6250

Table of Contents

The Scope of This Manual

This is a "how to" manual. If you follow the instructions of this manual, barring errors in the manual and bugs in your modifications to the system, you should be successful in making a new system. The instructions in this manual are not guaranteed to show you the most efficient way of making your changes. It only shows you a reliable way of making the changes.

This manual doesn't attempt to explain why the system is organized the way it is nor why each step is required. If your changes are not too major, this manual will help you to create a new system. It does not explain how to change low-level interface (e.g., interface to the Stream module) or low-level data structures (e.g., the segment tables or the structures on disk).

Recommendations

We make several recommendations to help you avoid errors that will make your system non-bootable. Remember, if you can no longer boot your PERQ, you can always boot from the "PERQ System Boot Floppy". Therefore, you always have a way of bringing up your PERQ.

1. Back up important files on floppy disks before you begin changing the PERQ Operating System. While it is improbable that you will destroy any files by changing the system, the importance of backup files cannot be stressed too much.

2. Maintain at least one partition of the disk that runs the old system. This enables you to boot your PERQ in the event that your new system contains bugs. When you receive your PERQ, the hard disk is divided into several partitions. At least one partition contains a pair of boot files: System.<n>.<x>.Boot and System.<n>.<x>.MBoot. The <n> in the file name is the current system version number, and the <x> in the file name represents the character that you hold down to use the corresponding boot files and partitions. The .Boot file contains the Operating System, and the .MBoot file contains the QCode interpreter microcode.

3. Do not change System.<n>.a.Boot and System.<n>.a.MBoot until your new system is completely debugged. These are the default boot files. If you do not change these files, you can boot the old system in the case that your new system contains errors.

4. Create a new directory or select an unused partition for your new experimental files when you begin making your changes to the system. Copy the sources of the files you want to change into this area before you begin editing. Compile all new .Seg files into this area. Do not use the root directory in the default partition: the one that is entered by the default boot letter ("a" is default--the same as not holding down a key). The default boot files are System.<n>.a.Boot and System.<n>.a.MBoot.

If you do not change the default boot files or the files in the root directory of the default partition, you will still have source and binary files that you can fall back on.

Overview

Creating a new system usually consists of the following steps.

1. Evaluate the change you intend to make
2. Create a directory to work in
3. Edit and compile system modules
4. Edit and compile system programs
5. Link the system and system programs
6. Prepare the system configuration file
7. Write a boot file
8. Test the new system
9. Iterate at step 3

Evaluate the Change You Intend to Make

Before you begin, you should determine how extensive the changes are. The following criteria tell you how much you need to change.

1. Are you changing the existing exports of any system modules? If not, you need only re-compile those modules that you change. If so, you may need to re-compile those modules and programs that import the ones you are changing.

2. Are you adding exports but not changing any that already exist? If you don't change existing exports, you need to re-compile only those modules that you change. You must, however, add your new exports at the end of the export list. By adding at the end of the export list, you do not change the storage allocation of existing variables or the routine numbers of existing procedures and functions. If you change either of these, you must re-compile all modules and programs that import the ones you are changing.

3. Are you changing the definition of data structures which are known by the microcode (e.g., memory manager tables) or data structures that live across boots (e.g., structures on disk)? If so, you may need to do a complicated bootstrapping operation to bring up your new system. This is beyond the scope of this manual.

4. Are you changing the existing exports of modules that the compiler knows about (Code, Dynamic, and Stream). If so, you again need to do a complicated bootstrapping operation. This too, is beyond the scope of this manual.

5. Are you changing the format of .Seg files? If so, you need to do a complicated bootstrapping operation which is beyond the scope of this manual.

Create a Directory, Edit, and Compile

Create a new directory for your experimental files. You do this with the MakeDir utility program. This new directory should be in the partition which contains the old system (probably the Boot partition). Copy sources of the system modules and programs into this directory. You may choose instead to work in a partition which, up until now, has not been used. Using a new partition is somewhat safer than merely creating a new directory in some old partition.

Edit the modules and programs that you need to change. Re-compile those modules and programs that you have changed and any others indicated by your evaluation of your changes.

Link the New System

Once all necessary changes and compilations have been done, you should link the new system and system programs. Choose a new system version number. Three Rivers Computer Corporation intends to use the version numbers between 1 and 99 for releases of the official PERQ Operating System. You should avoid these numbers to prevent conflicts with future Three Rivers Computer's releases. For example, you should choose version number 100 for your new system.

The new run files for System, Login, Shell, and Link should be in the root directory of the partition which contains your new system. You should use the following link commands to link your new system (assuming that the partition name is Part):

```
Link SystemSystem.100/System
Link LoginLogin.100
Link ShellShell.100
Link LinkLink.100
```

Prepare the System Configuration File

Before you write the boot file, you must create a system configuration file which describes the swappability of segments in the system. You probably can just copy the configuration file for the current version of the operating system. The default configuration file is named System.<n>.Config. If you need to change the swappability of segments in the system, you can copy the old file and edit it. Each line in the file describes the swappability of a single segment in the form:

<segment name> <swappability>

The <swappability> is chosen from the following:

SW - segment is swappable.
LS - segment is swappable but the memory manager should be reluctant to swap it out (this is not implemented yet).
US - segment is not swappable but may be moved in memory.
UM - segment is neither swappable nor movable.

Names with asterisks are recognized as special segment names. They are chosen from the following list:

SAT - Segment address table (default UM)
SIT - Segment information table (default US)
Cursor - Display cursor (default UM)
Screen - Display screen (default UM)
Font - Character set (default US)
Stack - Run-time stack (default US)
Names - System segment names (default SW)
IO - Input/output tables (default UM)

The default for code segments (modules) is US. We strongly suggest that you do not change the swappability of the special segments and, unless you are sure you know what you are doing, do not change the swappability of existing system modules. System data segments that the hardware or microcode uses cannot be moved, most data used by the operating system cannot be swapped, and the code that makes up the swapping system itself cannot be swapped. Since the default for code segments is US, you should add entries to the configuration file if you add modules to the system.

The default system configuration file is:

```
*SAT* UM
*SIT* US
*Cursor* UM
*Screen* UM
*Font* US
*IO* UM
System SW
Stream SW
Writer SW
IOErrMessages SW
Loader SW
```

```
Reader SW
Perq_String SW
Screen SW
FileSystem SW
GetTimeStamp SW
FileDefs SW
Memory SW
IO_Init SW
RunRead SW
FileDir SW
Scrounge SW
```

Write a New Boot File

You are now ready to write a boot file using the MakeBoot program. Before you run MakeBoot, choose a boot-letter for this new system; use one not already in use. You can run the Details program to find out which letters are in use. After choosing a boot letter, run MakeBoot and answer the questions in the following way:

-- Underlined text is what the PERQ types
-- Commentary is given inside { }
-- <CR> means type the RETURN key without entering any text
-- In this example, assume you have chosen the boot letter "z"

:MakeBoot

Root file name: System.100

Configuration file name [System.100.Config]: <CR>

Which character to boot from? z

Do you want to write the boot area [No]: <CR>

{ The boot area of the disk contains a microprogram which runs diagnostics and reads the .Boot and .MBoot files. You need to rewrite this only if you are making modifications to Vfy.Micro or SysB.Micro. }

Write a system boot file [Yes]: <CR>
Enter name of new system boot file [System.100.z.Boot]: <CR>
Existing boot file to copy (type return to build a new one): <CR>
Enter name of character set [Fix13.Kst]: <CR>

{ This writes the boot file containing the Pascal part of the system and special system segments such as the segment tables, the cursor, and the character set. Note that you may specify a character set which is different than the standard (Fix13.Kst). If you use a non-standard character set, some programs (like the Editor) may not work well. }

```
Write an interpreter boot file [Yes]: <CR>
Enter name of new micro boot file [System.100.z.MBoot]: <CR>
Existing boot file to copy (type return to build a new one): <CR>
Use standard interpreter microcode files? [Yes]: <CR>
Interpreter microcode file: ETHER10
Interpreter microcode$file: <CR>
```

{ This writes the boot file containing the microcode which is the Q-machine interpreter. Unless you are changing the interpreter microcode, you need only write this part once for a given boot letter. Note that you may add other microcode files to the boot file (as long as they do not overlap the standard microcode). }

Test the New System

You are now ready to boot your new system and test it. Hold down the boot key you selected ("z" in the example) and press the Boot button. If all goes well, your new system will announce itself. Note that when you try to run most programs, the loader informs you that they were linked under the old system. This means you must re-link them for your new system. It is a good idea to create another new directory to hold these run files. By putting the new run files in a directory by themselves, these run files will not get in your way when you are running the old system. If you want, you can make a Login profile to add this directory to your search list when you log in under your new system.

Once you are running the new system, you need to link the system utility programs. Set your path to the new directory that you created to contain the run files. Push the directory containing the old .Seg files onto your search list, and then push the directory containing the new .Seg files. Now, type:

```
Link ProgramName
```

for each utility program you wish to link.

Re-compile any programs that import modules whose exports have changed.

If your system doesn't come up at all, you can look at the diagnostic display to determine where in system initialization the system hangs.

Rewrite Other Boot Files

Once your new system is debugged and working, you can use MakeBoot to rewrite the boot files associated with other boot letters. Before you rewrite the old boot files, you must be sure that some partition contains all files that make up the new system. This includes files that you have not changed. If you fail to make a partition containing all source, binary, and run files, you run the risk of deleting portions of your new system when you delete the old system.

Perq.Files - PERQ Files Information

Three Rivers Computer Corporation
720 Gross Street
Pittsburgh, PA 15224
(412) 621-6250

Perq.Files - PERQ Files Information

Perq.Files is a list of the files distributed with the Three Rivers Computer Corporation PERQ. Source files are included in this list, but please note that they are included with the PERQ only if a source license is purchased. The User Library programs are also listed, but must be ordered from Trust - The Three Rivers Users' Society.

Three Rivers Computer Corporation
720 Gross Street
Pittsburgh, PA 15224
(412) 621-6250

>OS.SYSTEM.SOURCE - OPERATING SYSTEM SYSTEM SOURCES

file name	version file name on floppy short description.
ACB.DFS	1.1 ACB.DFS Activation control block definitions--common to microcode and Pascal.
ALIGNMEMORY.PAS	1.1 ALIGNM.PAS Allocate buffers on multiples of 256 word boundaries.
ARITH.PAS	2.2 ARITH.PAS Double precision arithmetic for the disk system.
CODE.PAS	1.6 CODE.PAS Run file and seg file definitions.
CONTROLSTORE.PAS	1.2 CONTRO.PAS Load controlstore and jump to controlstore.
DYNAMIC.PAS	1.4 DYNAMI.PAS Dynamic allocation routines - New and Dispose.
EEB.DFS	1.0 EEB.DFS Exception Enable Block definitions--common to Pascal and microcode.
EXCEPT.PAS	2.9 EXCEPT.PAS The exceptions module.

EXCEPT.DFS	1.4	EXCEPT.DFS
	Definitions of the exceptions--common to microcode and Pascal.	
GETTIMESTAMP.PAS	1.4	GETTIM.PAS
	Get time and date as TimeStamp.	
LIGHTS.PAS	1.2	LIGHTS.PAS
	Definitions of the lights.	
LOADER.PAS	2.6	LOADER.PAS
	Perq Q-Code loader.	
MEMORY.PAS	2.13	MEMORY.PAS
	Memory manager.	
MOVEMEM.PAS	1.6a	MOVEME.PAS
	Memory manager utility to move segments.	
PASLONG.PAS	0.0	PASLON.PAS
	Handles double-precision integers.	
PASREAL.PAS	0.1	PASREA.PAS
	Real numbers.	
PERQ_STRING.PAS	2.4	PERQST.PAS
	String manipulation package.	
RD.DFS	1.1	RD.DFS
	Routine dictionary definitions--common to microcode and Pascal.	
READER.PAS	2.1	READER.PAS
	Stream package input conversion routines.	
REALFUNCTIONS.PAS	1.0	REALFU.PAS
	Standard floating-point functions.	
RUNREAD.PAS	1.1	RUNREA.PAS
	Procedures to read run files.	
RUNWRITE.PAS	1.1	RUNWRI.PAS
	Procedures to write run files.	
SCROUNGE.PAS	0.14	SCROUN.PAS
	The preliminary debugger.	
STREAM.PAS	1.20	STREAM.PAS
	Stream package base routines - Get and Put.	
SYSTEM.PAS	2.4	SYSTEM.PAS
	Operating system main program.	

SYSTEMDEFS.PAS	1.2	SYSDEF.PAS
	Common system definitions.	
VIRTUAL.PAS	2.8	VIRTUA.PAS
	The Swapper.	
VRD.DFS	1.0	VRD.DFS
	Variable Routine Descriptor Definitions common to Pascal and microcode.	
WRITER.PAS	2.2	WRITER.PAS
	Stream package output conversion routines.	

>OS.SYSTEM.BINARY - OPERATING SYSTEM SYSTEM SEGMENT FILES

file name	version	file name on floppy
	short description.	
ARITH.SEG	2.2	ARITH.SEG
ALIGNMEMORY.SEG	1.1	ALIGNM.SEG
CONTROLSTORE.SEG	1.2	CONTRO.SEG
CODE.SEG	1.6	CODE.SEG
DYNAMIC.SEG	1.4	DYNAMI.SEG
EXCEPT.SEG	2.9	EXCEPT.SEG
GETTIMESTAMP.SEG	1.4	GETTIM.SEG
LOADER.SEG	2.6	LOADER.SEG
MEMORY.SEG	2.13	MEMORY.SEG
MOVEMEM.SEG	1.6a	MOVEME.SEG
PASREAL.SEG	-	PASREA.SEG
PASLONG.SEG	-	PASLON.SEG
PERQ_STRING.SEG	2.4	PERQST.SEG
READER.SEG	2.1	READER.SEG
REALFUNCTIONS.SEG	1.0	REALFU.SEG
RUNREAD.SEG	1.1	RUNREA.SEG
RUNWRITE.SEG	1.1	RUNWRI.SEG
SCROUNGE.SEG	0.14	SCROUN.SEG
STREAM.SEG	1.20	STREAM.SEG
:LINK	-	SYSTEM.6=SYSTEM/SYSTEM
SYSTEM.SEG	2.4	SYSTEM.SEG
VIRTUAL.SEG	2.8	VIRTUA.SEG
WRITER.SEG	2.2	WRITER.SEG

>OS.PROGRAMS.SOURCE - OPERATING SYSTEM PROGRAM SOURCES

file name	version	file name on floppy
	short description.	
CLOCK.PAS	1.6	CLOCK.PAS
	Get, set, convert time and date as TimeStamp or String.	

CMDPARSE.PAS	3.6	CMDPAR.PAS Command parser.
DOSWAP.PAS	1.0	DOSWAP.PAS Module for handling the "swap" command for shell.
GPIB.PAS	1.3	GPIB.PAS Routines for dealing with the Perq IEEE-488 bus.
HELPER.PAS	1.1	HELPER.PAS Module for presenting help menu.
INITSHELL.PAS	2.2	INITSH.PAS Does initialization for Shell.
LINK.PAS	4.3	LINK.PAS Q-Code linker.
LOGIN.PAS	2.0	LOGIN.PAS Login program.
MULTIREAD.PAS	1.0	MULTIR.PAS Does a very fast multisector read of a file.
POPCMDPARSE.PAS	1.8	POPCMD.PAS Popup window command parse.
POPUP.PAS	2.4	POPUP.PAS Provides a Pop Up Menu facility for POS.
POPUPCURS.PAS	2.1	POPUPC.PAS Defines cursors for Pop Up.
PROFILE.PAS	1.1	PROFILE.PAS Module for accessing the profile.
QUICKSORT.PAS	1.2	QUICKS.PAS Sorts arrays, integers and strings.
RANDOMNUMBERS.PAS	1.2	RANDOM.PAS High-quality random number generator.
RS232BAUD.PAS	1.1	RS232B.PAS Set RS232 baud rate with optional enable input.
SHELL.PARAS	-	SHELL.PAR Information necessary to generate SHELL/HELP.
SHELL.PAS	3.4	SHELL.PAS Top-level command processor.

SHELLDEFS.PAS	1.0	SHELLD.PAS
	Definition of SHELL. Data format.	
USERPASS.PAS	1.3	USERPA.PAS
	Lookup user name/password pairs.	
UTILPROGRESS.PAS	1.16	UTILPR.PAS
	Module for showing progress of utility programs.	

>OS.PROGRAMS.BINARY - OPERATING SYSTEM PROGRAMS BINARY FILES

file name	version	file name on floppy
	short description.	
CLOCK.SEG	1.6	CLOCK.SEG
CMDPARSE.SEG	3.6	CMDPAR.SEG
DOSWAP.SEG	1.0	DOSWAP.SEG
GPIB.SEG	1.3	GPIB.SEG
HELPER.SEG	1.1	HELPER.SEG
:LINK	-	SHELL.6=SHELL
SHELL.SEG	3.4	SHELL.SEG
INITSHELL.SEG	2.2	INITSH.SEG
:LINK	-	LINK.6=LINK
LINK.SEG	4.3	LINK.SEG
:LINK	-	LOGIN.6=LOGIN
LOGIN.SEG	2.0	LOGIN.SEG
PROFILE.SEG	1.1	PROFIL.SEG
USERPASS.SEG	1.3	USERPA.SEG
MULTIREAD.SEG	1.0	MULTIR.SEG
POPCMDPARSE.SEG	1.8	POPCMD.SEG
POPUP.SEG	2.4	POPUP.SEG
POPUPCURS.SEG	2.1	POPUPC.SEG
QUICKSORT.SEG	1.2	QUICKS.SEG
RANDOMNUMBERS.SEG	1.2	RANDOM.SEG
RS232BAUD.SEG	1.1	RS232B.SEG
SHELLDEFS.SEG	1.0	SHELLD.SEG
UTILPROGRESS.SEG	1.16	UTILPR.SEG

>OS.IO.SOURCE - INPUT/OUTPUT SYSTEM SOURCE MODULES

file name	version	file name on floppy
	short description.	
ALLOCDISK.PAS	2.8	ALLOCD.PAS
	Allocation of sectors from the disk free list.	
DISKIO.PAS	3.12	DISKIO.PAS
	Medium-level disk input/output routines.	

ETHER10IO.PAS	1.8	ETHER1.PAS
	Ethernet IO interface.	
ETHERINTERRUPT.PAS	1.1	ETHERI.PAS
	Interrupt service for 10 MBaud ethernet.	
FILEACCESS.PAS	1.7	FILEAC.PAS
	File system segment routines.	
FILEDEFS.PAS	1.6	FILEDE.PAS
	Definitions used by the File System.	
FILEDIR.PAS	2.6	FILEDI.PAS
	File system directory routines.	
FILESYSTEM.PAS	7.3	FILESY.PAS
	File system--high-level disk input/output routines.	
FILETYPES.PAS	1.2	FILETY.PAS
	Definitions for the file type field.	
FILEUTILS.PAS	1.10	FILEUT.PAS
	File utilities not needed by system.	
IO.PAS	4.8	IO.PAS
	Input/output manager.	
IOERRORS.PAS	1.2	IOERRO.PAS
	Input/output error number constants.	
IOERRMESSAGES.PAS	1.1	IOERRM.PAS
	Names for the Input/output errors.	
IO_INIT.PAS	5.9	IOINIT.PAS
	Input/output manager initialization.	
IO_OTHERS.PAS	5.7	IOOTHE.PAS
	Other Input/output manager procedures and functions.	
IO_PRIVATE.PAS	5.9	IOPRIV.PAS
	Interrupt procedures and private definitions.	
IO_UNIT.PAS	6.2	IOUNIT.PAS
	The basic UnitIO procedures and functions.	
PMATCH.PAS	2.5	PMATCH.PAS
	Directory search with wild cards.	
RASTER.PAS	-	RASTER.PAS
	Raster-op definitions.	

READDISK.PAS	1.4	READDI.PAS
	Upper-level disk input/output routines.	
SCREEN.PAS	3.12	SCREEN.PAS
	Screen manager.	

>OS.IO.BINARY - OPERATING SYSTEM IO BINARY FILES

file name	version	file name on floppy
	short description.	
ALLOCDISK.SEG	2.8	ALLOCD.SEG
DISKIO.SEG	3.12	DISKIO.SEG
ETHER10IO.SEG	1.8	ETHER1.SEG
ETHERINTERRUPT.SEG	1.1	ETHERI.SEG
FILEACCESS.SEG	1.7	FILEAC.SEG
FILEDEFS.SEG	1.6	FILEDE.SEG
FILEDIR.SEG	2.6	FILEDI.SEG
FILESYSTEM.SEG	7.3	FILESY.SEG
FILEUTILS.SEG	1.10	FILEUT.SEG
IO.SEG	4.8	IO.SEG
IOERRMESSAGES.SEG	1.1	IOERRM.SEG
IO_INIT.SEG	5.9	IOINIT.SEG
IO_OTHERS.SEG	5.7	IOOTHE.SEG
IO_PRIVATE.SEG	5.9	IOPRIV.SEG
IO_UNIT.SEG	6.2	IOUNIT.SEG
PMATCH.SEG	2.5	PMATCH.SEG
READDISK.SEG	1.4	READDI.SEG
SCREEN.SEG	3.12	SCREEN.SEG

>OS.MISCELLANEOUS - OPERATING SYSTEM SPECIAL FILES

file name	version	file name on floppy
	short description.	
DEFAULT.PROFILE	-	DEFAUL.PRO
	Default login profile.	
DELETE.CURSOR	-	DELETE.CUR
	Cursor used when delete is busy.	

DIRTREE.CURSOR - DIRTRE.CUR
Cursor used by Dirtree program.

FIX13.KST - FIX13.KST
System character set file.

SCAVENGER.ANIMATE - SCAVEN.ANI
File of cursors for scavenger.

SYSTEM.USERS - SYSTEM.USE
Valid system users and passwords.

SYSTEM.6.CONFIG - SYSTEM.CON
Description of swappability for use by MakeBoot.

UTILPROGRESS.CURSOR - UTILPR.CUR
Cursor used by Utilprogress.

>OS.NOSOURCE.SOURCE - OPERATING SYSTEM NONSOURCE FILES

file name version file name on floppy
short description.

ACB.DFS 1.1 ACB.DFS
Activation control block definitions--common to microcode and Pascal.

ALLOCDISK.PAS 2.8 ALLOCD.PAS
Allocation of sectors from the disk free list.

ARITH.PAS 2.2 ARITH.PAS
Double precision arithmetic for the disk system.

CLOCK.PAS 1.6 CLOCK.PAS
Get, set, convert time and date as TimeStamp or String.

CMDPARSE.PAS 3.6 CMDPAR.PAS
Command parser.

CODE.PAS 1.6 CODE.PAS
Run file and seg file definitions.

CONTROLSTORE.PAS 1.2 CONTRO.PAS
Load controlstore and jump to controlstore.

DISKIO.PAS 3.12 DISKIO.PAS
Medium-level disk input/output routines.

DYNAMIC.PAS 1.4 DYNAMI.PAS
Dynamic allocation routines - New and Dispose.

EEB.DFS 1.0 EEB.DFS
Exception Enable Block definitions--common to Pascal and microcode.

ETHER10IO.PAS 1.8 ETHER1.PAS
Ethernet IO interface.

EXCEPT.PAS 2.9 EXCEPT.PAS
The exceptions module.

EXCEPT.DFS 1.4 EXCEPT.DFS
Definitions of the exceptions--common to microcode and Pascal.

FILEACCESS.PAS 1.7 FILEAC.PAS
File system segment routines.

FILEDEFS.PAS 1.6 FILEDE.PAS
Definitions used by the File System.

FILEDIR.PAS 2.6 FILEDI.PAS
File system directory routines.

FILESYSTEM.PAS 7.3 FILESY.PAS
File system--high-level disk input/output routines.

FILETYPES.PAS 1.2 FILETY.PAS
Definitions for the file type field.

FILEUTILS.PAS 1.10 FILEUT.PAS
File utilities not needed by system.

GETTIMESTAMP.PAS 1.4 GETTIM.PAS
Get time and date as TimeStamp.

GPIB.PAS 1.3 GPIB.PAS
Routines for dealing with the Perq IEEE-488 bus.

IO.PAS 4.8 IO.PAS
Input/output manager.

IOERRORS.PAS 1.2 IOERRO.PAS
Input/output error number constants.

IOERRMESSAGES.PAS 1.1 IOERRM.PAS
Names for the Input/output errors.

IO_INIT.PAS	5.9	IOINIT.PAS Input/output manager initialization.
IO_OTHERS.PAS	5.7	IOOTHE.PAS Other Input/output manager procedures and functions.
IO_PRIVATE.PAS	5.9	IOPRIV.PAS Interrupt procedures and private definitions.
IO_UNIT.PAS	6.2	IOUNIT.PAS The basic UnitIO procedures and functions.
LIGHTS.PAS	1.2	LIGHTS.PAS Definitions of the lights.
LOADER.PAS	2.6	LOADER.PAS Perq Q-Code loader.
MEMORY.PAS	2.13	MEMORY.PAS Memory manager.
MOVEMEM.PAS	1.6a	MOVEME.PAS Memory manager utility to move segments.
MULTIREAD.PAS	1.0	MULTIR.PAS Does a very fast multisector read of a file.
PASLONG.PAS	0.0	PASLON.PAS Handles double-precision integers.
PASREAL.PAS	0.1	PASREA.PAS Real numbers.
PERQ_STRING.PAS	2.4	PERQST.PAS String manipulation package.
PMATCH.PAS	2.5	PMATCH.PAS Directory search with wild cards.
POPCMDPARSE.PAS	1.8	POPCMD.PAS Popup window command parse.
POPUP.PAS	2.4	POPUP.PAS Provides a Pop Up Menu facility for POS.
POPUPCURS.PAS	2.1	POPUPC.PAS Defines cursors for Pop Up.
PROFILE.PAS	1.1	PROFILE.PAS Module for accessing the profile.

RANDOMNUMBERS.PAS	1.2	RANDOM.PAS
	High-quality random number generator.	
RASTER.PAS	-	RASTER.PAS
	Raster-op definitions.	
RD.DFS	1.1	RD.DFS
	Routine dictionary definitions--common to microcode and Pascal.	
READDISK.PAS	1.4	READDI.PAS
	Upper-level disk input/output routines.	
READER.PAS	2.1	READER.PAS
	Stream package input conversion routines.	
REALFUNCTIONS.PAS	1.0	REALFU.PAS
	Standard floating-point functions.	
RS232BAUD.PAS	1.1	RS232B.PAS
	Set RS232 baud rate with optional enable input.	
RUNREAD.PAS	1.1	RUNREA.PAS
	Procedures to read run files.	
RUNWRITE.PAS	1.1	RUNWRI.PAS
	Procedures to write run files.	
SCREEN.PAS	3.12	SCREEN.PAS
	Screen manager.	
SCROUNGE.PAS	0.14	SCROUN.PAS
	The preliminary debugger.	
STREAM.PAS	1.20	STREAM.PAS
	Stream package base routines - Get and Put.	
SYSTEM.PAS	2.4	SYSTEM.PAS
	Operating system main program.	
SYSTEMDEFS.PAS	1.2	SYSDEF.PAS
	Common system definitions.	
USERPASS.PAS	1.3	USERPA.PAS
	Lookup user name/password pairs.	
UTILPROGRESS.PAS	1.16	UTILPR.PAS
	Module for showing progress of utility programs.	
VIRTUAL.PAS	2.8	VIRTUA.PAS
	The Swapper.	

VRD.DFS	1.0 VRD.DFS Variable Routine Descriptor Definitions common to Pascal and microcode.
WRITER.PAS	2.2 WRITER.PAS Stream package output conversion routines.
>CANON.SOURCE	- CANON PRINTER SOURCE FILES
file name	version file name on floppy short description.
CANON.PAS	1.0 CANON.PAS Main PASCAL interface to CANON LBP-10 laser printer.
CAN.PAS	1.0 CAN.PAS CANON laser printer, fixed width font document print program.
CANP.PAS	1.0 CANP.PAS CANON laser printer, proportional spaced font document print program.
CANON.MICRO	1.0 CANON.MIC Microcode support required for, and loaded by CANON.PAS.
CANONBLOCK.PAS	1.0 CANONB.PAS PASCAL module to support memory bit image dump to CANON LBP-10.
CANONUTILS.PAS	1.0 CANONU.PAS General utility routines for CANON.
INFILE.PAS	1.0 INFILE.PAS High speed disk buffer reading support for CANON modules.
CBT.PAS	1.0 CBT.PAS Test program for CANONBLOCK.

>CANON.BINARY - CANON PRINTER BINARY FILES

file name	version	file name on floppy short description.
CAN40.KST	-	CAN40.KST
CAN40P.KST	-	CAN40P.KST
CANON.SEG	1.0	CANON.SEG
CAN.SEG	1.0	CAN.SEG
CANP.SEG	1.0	CANP.SEG
CANON.BIN	1.0	CANON.BIN
CANONBLOCK.SEG	1.0	CANONB.SEG
CANONUTILS.SEG	1.0	CANONU.SEG
INFILE.SEG	1.0	INFILE.SEG
CBT.SEG	1.0	CBT.SEG

>UTILITY.OTHERS.SOURCE - UTILITIES SOURCES

file name	version	file name on floppy short description.
BYE.PAS	2.2	BYE.PAS Logoff.
DETAILS.PAS	1.11	DETAIL.PAS Print system status information.
EDITOR.PAS	2.0	EDIT.PAS Editor main program.
EDITORI.PAS	2.0	EDITI.PAS Editor initialization module.
EDITORU.PAS	2.0	EDITU.PAS Editor utilities module.
EDITORK.PAS	2.0	EDITK.PAS Editor key-selection module.
EDITORT.PAS	2.0	EDITT.PAS Editor termination module.
EDITORK.PARAS	2.0	EDITK.PAR Souce to generate .HELP files for EDITOR.
EXPANDTABS.PAS	1.0	EXPAND.PAS Copy a text file and expand tabs to 8 character columns.
FINDSTRING.PAS	2.1	FINDST.PAS Searches files in directory for specified string.

GOODBY.MICRO	1.0	GOODBY.MIC
	Power down microcode.	
HELPGEN.PAS	0.0	HELPGE.PAS
	Process .PARAS files to make help files for use by Helper.	
MAKEBOOT.PAS	4.4	MAKEBO.PAS
	Make SYSTEM.nn.BOOT files.	
PATCH.PAS	1.7	PATCH.PAS
	Program to peek and poke into files.	
PERQ.FILES.PAS	1.3	PERQFI.PAS
	Program to gobble this file.	
SETTIME.PAS	2.0	SETTIM.PAS
	Sets the system date and time.	
USERCONTROL.PAS	1.3	USERCO.PAS
	Add/delete users from the password file.	

>UTILITY.OTHERS.BINARY - UTILITY OTHER BINARY FILES

file name	version	file name on floppy short description.
:LINK	2.2	BYE
BYE.SEG	-	BYE.SEG
GOODBY.BIN	-	GOODBY.BIN
.break :LINK		1.11 DETAILS
DETAILS.SEG	-	DETAIL.SEG
:LINK	2.0	EDITOR
EDITOR.SEG	-	EDIT.SEG
EDITORI.SEG	-	EDITI.SEG
EDITORK.SEG	-	EDITK.SEG
EDITORU.SEG	-	EDITU.SEG
EDITORT.SEG	-	EDITT.SEG
:LINK	1.0	EXPANDTABS
EXPANDTABS.SEG	-	EXPAND.SEG
:LINK		
FINDSTRING.SEG	2.1	FINDST.SEG
HELPGEN.SEG	0.0	HELPGE.SEG
:LINK	4.4	MAKEBOOT
MAKEBOOT.SEG	-	MAKEBO.SEG
:LINK	1.4	PATCH
PATCH.SEG	-	PATCH.SEG
:LINK	1.3	PERQ.FILES
PERQ.FILES.SEG	-	PERQFI.SEG
:LINK		
SETTIME.SEG	2.0	SETTIM.SEG
:LINK	1.3	USERCONTROL
USERCONTROL.SEG	-	USERCO.SEG

>UTILITY.DEVICE.SOURCE - UTILITIES DEVICES

file name	version file name on floppy short description.
CHATTER.PAS	0.6 CHATTE.PAS RS232 dumb terminal program.
FLOPPYCOPY.PAS	2.2 FLPCOP.PAS Floppydup part of floppy (formerly COPYFLOPPY).
FLOPPY.PAS	3.2 FLOPPY.PAS Main program for general floppy utility.
FLOPPYDEFS.PAS	0.0 FLPDEF.PAS Global defs for floppy.
FLOPPYUTILS.PAS	0.1 FLPUTI.PAS Utility routines for floppy.
FLOPPYFORMAT.PAS	0.1 FLPFOR.PAS Diskette format routines for floppy (formerly module FORMAT).
FTPUTILS.PAS	4.2 FTPUTILS.PAS Utilities for file transfer module.
FTPUSER.PAS	4.3 FTPUSE.PAS Implements all the commands.
FTP.PAS	4.3 FTP.PAS File transfer program.
PRINT.PAS	2.13 PRINT.PAS Print a text file on an HP 7310A printer or through the TNW GPIB to RS232 converter.
FLOPPYTRANSFERS.PAS	0.1 FLPTRA.PAS Utility routines for floppy (formerly RT11 and RT11Utils).
SETBAUD.PAS	0.0 SETBAU.PAS Program to set RS232 Baud rate.

>UTILITY.DEVICE.BINARY - UTILITY DEVICE BINARY FILES

file name	version	file name on floppy short description.
:LINK	0.6	CHATTER
CHATTER.SEG	-	CHATTE.SEG
:LINK	3.2	FLOPPY
FLOPPY.SEG	-	FLOPPY.SEG
FLOPPYDEFS.SEG	-	FLPDFS.SEG
FLOPPYCOPY.SEG	2.1	FLPCOP.SEG
FLOPPYUTILS.SEG	0.1	FLPUTI.SEG
FLOPPYFORMAT.SEG	0.1	FLPFOR.SEG
FLOPPYTRANSFERS.SEG	0.1	FLPTRA.SEG
:LINK	4.3	FTP.SEG
FTP.SEG	-	FTP.SEG
FTPUTILS.SEG	4.2	FTPUTI.SEG
FTPUSER.SEG	4.3	FTPUSE.SEG
:LINK	2.13	PRINT
PRINT.SEG	-	PRINT.SEG
:LINK	0.0	SETBAUD
SETBAUD.SEG	-	SETBAU.SEG

>UTILITY.FILE.SOURCE - FILE SYSTEM UTILITIES SOURCES

file name	version file name on floppy short description.
APPEND.PAS	3.1 APPEND.PAS Append a file to the end of another.
COPY.PAS	5.3 COPY.PAS Copy a file to another.
DELETE.PAS	2.4 DELETE.PAS Delete a file or files.
DIRECT.PAS	4.4 DIRECT.PAS Print a directory listing.
DIRTREE.PAS	3.2 DIRTRE.PAS Display the directory structure of a partition as a tree.
FIXPART.PAS	0.5 FIXPAR.PAS Fix smashed partition or disk information blocks.
MAKEDIR.PAS	2.2 MAKEDI.PAS Make directories.
PARTITION.PAS	3.2 PARTIT.PAS Initialize partitions on disks.
RENAME.PAS	5.3 RENAME.PAS Change the name of a file.
SCAVENGER.PAS	3.2 SCAVEN.PAS Analyze and reconstruct disks and directories.
DIRSCAVENGE.PAS	- DIRSCA.PAS Reconstruct directories.
SETSEARCH.PAS	1.3 SETSEA.PAS Change search lists.
TYPEFILE.PAS	4.2 TYPEFI.PAS Type file to the console.

>UTILITY.FILE.BINARY - UTILITY FILE SEG FILES

file name	version	file name on floppy short description.
:LINK	2.1	APPEND
APPEND.SEG	-	APPEND.SEG
:LINK	5.3	COPY
COPY.SEG	-	COPY.SEG
:LINK	2.4	DELETE
DELETE.SEG	-	DELETE.SEG
:LINK	4.4	DIRECT
DIRECT.SEG	-	DIRECT.SEG
:LINK	2.1	DIRTREE
DIRTREE.SEG	-	DIRTRE.SEG
:LINK	0.4	FIXPART
FIXPART.SEG	-	FIXPAR.SEG
:LINK	1.3	MAKEDIR
MAKEDIR.SEG	-	MAKEDI.SEG
:LINK	3.1	PARTITION
PARTITION.SEG	-	PARTIT.SEG
:LINK	5.3	RENAME
RENAME.SEG	-	RENAME.SEG
:LINK	3.2	SCAVENGER
SCAVENGER.SEG	-	SCAVEN.SEG
:LINK	1.2	SETSEARCH
SETSEARCH.SEG	-	SETSEA.SEG
:LINK	4.2	TYPEFILE
TYPEFILE.SEG	-	TYPEFI.SEG

>MICROCODE.SOURCE - MICROCODE SOURCES

file name	version file name on floppy short description.
PERQ.MICRO	2.4 PERQ.MIC Perq Q-code interpreter microcode.
PERQ.DFS	1.4 PERQ.DFS Definitions of registers, constants, and entrypoints used by Perq.Micro and other microprograms.
PERQ.QCODES.DFS	- QCODES.DFS Definitions of QCode instruction names and numbers.
PERQ.QCODE.1	- QCODE.1 Opcode interpreter routines for Perq.Micro (part 1).
PERQ.QCODE.2	- QCODE.2 Opcode interpreter routines for Perq.Micro (part 2).
PERQ.QCODE.3	- QCODE.3 Opcode interpreter routines for Perq.Micro (part 3).
PERQ.QCODE.4	- QCODE.4 Opcode interpreter routines for Perq.Micro (part 4).
PERQ.QCODE.5	- QCODE.5 Opcode interpreter routines for Perq.Micro (part 5).
PERQ.QCODE.6	- QCODE.6 Opcode interpreter routines for Perq.Micro (part 6).
PERQ.QCODE.7	- QCODE.7 Opcode interpreter routines for Perq.Micro (part 7).
PERQ.FLOAT.MUL	FLOAT.MUL Special multiply for floating point.
PERQ.ROUTINE.1	- ROUTIN.1 Subroutines for Perq.Micro (part 1).
PERQ.ROUTINE.2	- ROUTIN.2 Subroutines for Perq.Micro (part 2).

PERQ.INIT	- PERQ.INI Initialization for Perq.Micro.
RO.MICRO	0.6 RO.MIC Raster-op microcode.
LINE.MICRO	1.1 LINE.MIC Line drawing microcode.
IO.MICRO	1.10 IO.MIC Input/output microcode.
IO.DFS	1.5 IO.DFS Definitions of registers, constants, and entrypoints used by IO.Micro and other microprograms.
IOE3.MICRO	1.2 IOE3.MIC Microcode to drive the 3MBaud EtherNet.
VFY.MICRO	1.8 VFY.MIC Verify that the hardware seems to work.
SYSB.MICRO	2.5 SYSB.MIC System boot microcode.
BOOT.MICRO	4.0 BOOT.MIC Boot-prom microcode.
KRNL.MICRO	1.2 KRNL.MIC Perq microcode kernel.
LINK.MICRO	1.2 LINK.MIC 16-bit parallel interface microcode.

>MICROCODE.BINARY - MICROCODE BIN FILES

file name	version	file name on floppy short description.
PERQ.BIN	2.4	PERQ.BIN
IO.BIN	1.10	IO.BIN
VFY.BIN	1.8	VFY.BIN
SYSB.BIN	2.5	SYSB.BIN
BOOT.BIN	4.0	BOOT.BIN
KRNL.BIN	1.2	KRNL.BIN
LINK.BIN	1.2	LINK.BIN

>MICROCODE.MORE.SOURCE - MICROCODE SOURCES

file name	version	file name on floppy short description.
ETHER10.MICRO	4.0	ETHER1.MIC 10 MBaud ethernet microcode.

>MICROCODE.MORE.BINARY - MICROCODE MORE BINARY FILES

file name	version	file name on floppy short description.
ETHER10.BINARY	4.0	ETHER1.BIN

>MICROCODE.SUPPORT.SOURCE - MICROCODE SUPPORT SOURCES

file name	version	file name on floppy short description.
MICROOPTION.PAS	1.0	MICROO.PAS Option processor for PrqMic and PrqPlace.
ODTPRQ.PAS	8.0	ODTPRQ.PAS Simple Perq to Perq microcode debugger.
ODTUTILS.PAS	-	ODTUTI.PAS Utility routines for ODTPRQ.
ODTDUMP.PAS	-	ODTDUM.PAS The ODTPRQ dump subsystem.
PRQDIS.PAS	1.2	PRQDIS.PAS Perq microcode disassembler.
PRQPL_SORT.PAS	2.0	PRQPLS.PAS Sorting routines for PrqPlace.
PRQPLACE.PAS	2.5	PRQPLA.PAS Perq microcode placer.
PRQMIC.PAS	2.8	PRQMIC.PAS Perq microcode assembler.
PMEGEN.PAS	-	PMEGEN.PAS Program to create PRQMIC.ERROR from PRQMIC.ERR.TEXT.
PRQMIC.ERR.TEXT	-	PRQERR.TXT Source for PRQMIC.ERROR (error message text).

>MICROCODE.SUPPORT.BINARY - MICROCODE SUPPORT SEG FILES

file name	version	file name on floppy short description.
MICROOPTION.SEG	1.0	MICROO.SEG
PRQPL_SORT.SEG	2.0	PRQPLS.SEG
:LINK	2.8	PRQMIC
PRQMIC.SEG	-	PRQMIC.SEG
PRQMIC.ERROR	-	PRQMIC.ERR
:LINK	2.5	PRQPLACE
PRQPLACE.SEG	-	PRQPLA.SEG
:LINK	1.2	PRQDIS
PRQDIS.SEG	-	PRQDIS.SEG
:LINK	8.0	ODTPRQ
ODTPRQ.SEG	-	ODTPRQ.SEG
ODTUTILS.SEG	-	ODTUTI.SEG
ODTDUMP.SEG	-	ODTDUM.SEG
ODT13.KST	-	ODT13.KST

>DOCUMENTATION - DOCUMENTATION

file name	version file name on floppy short description.
EDITOR.DOC	- EDITOR.DOC Editor quick guide.
EDITORK.DOC	2.0 EDITK.DOC Editor User's Guide.
EXAMPLES.DOC	D.6 EXAMPL.DOC Programming examples.
FAULT.DOC	D.6 FAULT.DOC Fault dictionary for the diagnostic display.
FILE.FORMAT	- FILE.FOR Describe source file format for Perq software.
FILES.DOC	- FILES.DOC File system user's manual.
MICRO.DOC	D.6 MICRO.DOC Microprogrammer's guide.
MAKESYSTEM.DOC	- MAKESY.DOC How to make a new version of the operating system.
PASCAL.DOC	D.6 PASCAL.DOC Pascal extensions.
PERQ_Z80.DOC	- PERQZ8.DOC Description of PERQ/Z80 protocol.
QCODE.DOC	D.6 QCODE.DOC QCode reference manual.
SEGMENT.DOC	- SEGMEN.DOC Segment file format.
SETBAUD.DOC	- SETBAU.DOC User Information for program to set RS232 Baud rate.

>DOCUMENTATION.MORE - MORE DOCUMENTATION

file name	version file name on floppy short description.
UTILITIES.DOC	D.6 UTILIT.DOC Utility programs manual.
INTRO.DOC	D.6 INTRO.DOC Intorduction to the PERQ operating system.
PERQ.FILES	- PERQ.FIL This list.
PERQ.FILES.LABELS	- PERQFI.LAB Masters for labels on Perq.Files floppies.
POS.DOC	- POS.DOC Operating system interface guide.

>TEST.SOURCE - TEST PROGRAM SOURCES

file name	version file name on floppy short description.
CHARS.PAS	1.1 CHARS.PAS Screen magnifier.
CROSSHATCH.PAS	0.2 CROSSH.PAS Put crosshatch or checkerboard on display screen.
DISPATCH.MICRO	1.1 DISPAT.MIC Dispatch diagnostic.
DTST.MICRO	1.2 DTST.MIC Disk test microcode.
DUAL.MICRO	1.0 DUAL.MIC Microstore dual address test.
HIGH.MICRO	1.0 HIGH.MIC Test for stuck bits in the high bank of the microstore.
JUMP.MICRO	1.0 JUMP.MIC Test microcode jumps.
KEYTEST.PAS	1.5 KEYTST.PAS Keyboard test program.
LOOP.MICRO	0.0 LOOP.MIC Simple tests that repeat--allowing probing of boards.
LOW.MICRO	1.0 LOW.MIC Test for stuck bits in the low bank of the microstore.
MEM.MICRO	2.0 MEM.MIC Dual addressing test of the memory.
NEXTOP.MICRO	1.1 NEXTOP.MIC NextOp diagnostic.
	Register diagnostic.
NXTI.MICRO	1.1 NXTI.MIC NextInst diagnostic.

PART.MICRO	1.1	PART.MIC
	Memory Parity diagnostic.	
PBT.MICRO	1.1	PBT.MIC
	Pre-boot diagnostic.	
PDM.PAS	0.4	PDM.PAS
	Perq diagnostic monitor.	
PDM2.PAS	-	PDM2.PAS
	PDM 2nd module.	
PDCOMMON.PAS	-	PDCOMM.PAS
	PDM/PDS common definitions.	
PDMUTILS.PAS	-	PDMUTI.PAS
	PDM utility routines.	
PDMLOAD.PAS	-	PDMLOD.PAS
	PDM Pascal program loader.	
PDM.MAS	-	PDM.MAS
	PDM master file for current diagnostics.	
PDS.PAS	-	PDS.PAS
	Perq diagnostic slave.	
PDM.HELP	-	PDM.HELP
	PDM help file.	
PDM.MICRO	-	PDM.MIC
	Pseudo PDS microcode example.	
PDMVFY.MICRO	1.1	PDMVFY.MIC
	PDM version of VFY.	
RAT.MICRO	1.3	RAT.MIC
	Source data suspicious raster-op test.	
REGT.MICRO	1.1	REGT.MICRO
	Test of XY register file.	
SHIFT.MICRO	1.0	SHIFT.MIC
	Test of the shift hardware.	
STACK.MICRO	1.0	STACK.MIC
	20-bit, 16-level stack test.	
TESTFLOPPY.PAS	2.1	TSTFPY.PAS
	Test and format floppies.(formerly FLOPPY)	
TST.MICRO	1.1	TST.MIC
	Pre-Boot diagnostic.	

>TEST.BINARY - TEST PROGRAM SEG, RUN, AND BIN FILES

file name	version	file name on floppy short description.
:LINK	1.1	CHARS
CHARS.SEG	-	CHARS.BIN
:LINK	0.2	CROSSHATCH
CROSSHATCH.SEG	-	CROSSH.SEG
DISPATCH.BIN	1.1	DISPAT.BIN
DTST.BIN	1.2	DTST.BIN
DUAL.BIN	1.0	DUAL.BIN
HIGH.BIN	1.0	HIGH.BIN
JUMP.BIN	1.0	JUMP.BIN
:LINK	1.5	KEYTEST
KEYTEST.SEG	-	KEYTST.SEG
LOOP.BIN	0.0	LOOP.BIN
LOW.BIN	1.0	LOW.BIN
MEM.BIN	2.0	MEM.BIN
NEXTOP.BIN	1.1	NEXTOP.BIN
NXTI.BIN	1.1	NXTI.BIN
PART.BIN	1.1	PART.BIN
PBT.BIN	1.1	PBT.BIN
RAT.BIN	1.3	RAT.BIN
REGT.BIN	1.1	REGT.BIN
SHIFT.BIN	1.1	SHIFT.BIN
STACK.BIN	1.0	STACK.BIN
TST.BIN	1.1	TST.BIN
:LINK	2.1	TESTFLOPPY
TESTFLOPPY.SEG	-	TSTFPY.SEG

>PASCAL.SOURCE - PASCAL COMPILER SOURCES

file name	version	file name on floppy
	short description.	
PASCAL.PAS	6.0	PASCAL.PAS
	Pascal compiler global definitions.	
PAS0.PAS	-	PAS0.PAS
PAS1.PAS	-	PAS1.PAS
PAS2.PAS	-	PAS2.PAS
QCODES.DFS	-	QCODES.DFS
	Q-Code const definitions.	
COMPINIT.PAS	-	COMPIN.PAS
	Initialization.	
CODEGEN.PAS	-	CODEGE.PAS
	Code generator.	
DECPART.PAS	-	DECPAR.PAS
	Declaration processor.	
DEC0.PAS	-	DEC0.PAS
DEC1.PAS	-	DEC1.PAS
DEC2.PAS	-	DEC2.PAS
BODYPART.PAS	-	BODYPA.PAS
	Procedure/function/program body processor.	
BODY0.PAS	-	BODY0.PAS
BODY1.PAS	-	BODY1.PAS
BODY2.PAS	-	BODY2.PAS
BODY3.PAS	-	BODY3.PAS

>PASCAL.BINARY - PASCAL COMPILER SEG FILES

file name	version	file name on floppy short description.
:LINK	6.0	PASCAL
PASCAL.SEG	-	PASCAL.SEG
COMPINIT.SEG	-	COMPIN.SEG
CODEGEN.SEG	-	CODEGE.SEG
DECPART.SEG	-	DECPAR.SEG
BODYPART.SEG	-	BODYPA.SEG
EXPEXPR.SEG	-	EXPEXP.SEG
PASCAL.SYNTAX	-	PASCAL.SYN
PASCAL.RESWORDS	-	PASCAL.RES
LEX.SEG	0.0	LEX.SEG
FQCODES.SEG	1.0	FQCODE.SEG
FRESWORDS.SEG	1.0	FRESWO.SEG
FSYNTAX.SEG	1.0	FSYNTA.SEG
:LINK	2.0	QDIS
QDIS.SEG	-	QDIS.SEG
QCODES	-	QCODES

>PASCAL.MORE.SOURCE - MORE PASCAL COMPILER SOURCES

file name	version	file name on floppy short description.
EXPEXPR.PAS	-	EXPEXP.PAS Expression expansion.
EXPR0.PAS	-	EXPR0.PAS
EXPR1.PAS	-	EXPR1.PAS
EXPR2.PAS	-	EXPR2.PAS
EXPR3.PAS	-	EXPR3.PAS
FSYNTAX.PAS	1.0	FSYNTA.PAS Program to generate PASCAL.SYNTAX from SYNTAX.DAT.
SYNTAX.DAT	-	SYNTAX.DAT Error message data file.
QDIS.PAS	2.0	QDIS.PAS Q-Code disassembler. Also needs QCODES.PAS.
FQCODES.PAS	1.0	FQCODE.PAS Program to generate QCODES from QCODES.DAT.
QCODES.DAT	-	QCODES.DAT Q-Code name data file.
FRESWORDS.PAS	1.0	FRESWO.PAS Program to generate QCODES from QCODES.DAT.
RESWORDS.DAT	-	RESWOR.DAT Q-Code name data file.
LEX.PAS	0.0	LEX.PAS Lexical scanner for compiler.

>DEMO.SOURCE - DEMONSTRATION PROGRAM SOURCES

file name	version file name on floppy short description.
KAL.PAS	- KAL.PAS Kaleidoscope display.
KINETIC.PAS	- KINETI.PAS Demonstrate random raster-ops.
LIFE.PAS	- LIFE.PAS The game of life.
LINE.PAS	- LINE.PAS Line drawing display.
PETAL.PAS	- PETAL.PAS Cycloid drawing display.
MULDIV.PAS	- MULDIV.PAS Double precision multiply and divide for Petal.
SLEEP.PAS	- SLEEP.PAS Sleep for a specified period of time.
SCREENDUMP.PAS	0.0 SCRDMP.PAS Print an image of the screen to an HP 7310A printer.
SEISMO.PAS	- SEISMO.PAS Multi-pen chart recorder display.
SKETCH.PAS	- SKETCH.PAS Sketch on the screen using the tablet.

>DEMO.BINARY - DEMONSTRATION PROGRAM SEG FILES

file name	version	file name on floppy short description.
:LINK	-	KAL
KAL.SEG	-	KAL.SEG
:LINK	-	KINETIC
KINETIC.SEG	-	KINETI.SEG
:LINK	-	LIFE
LIFE.SEG	-	LIFE.SEG
:LINK	-	LINE
LINE.SEG	-	LINE.SEG
:LINK	-	PETAL
PETAL.SEG	-	PETAL.SEG
MULDIV.SEG	-	MULDIV.SEG
SLEEP.SEG	-	SLEEP.SEG
:LINK	0.0	SCREENDUMP
SCREENDUMP.SEG	-	SCRDMP.SEG
:LINK	-	SEISMO
SEISMO.SEG	-	SEISMO.SEG
:LINK	-	SKETCH
SKETCH.SEG	-	SKETCH.SEG

>DEMO.SIGGRAPH.SOURCE - SIGGRAPH 80 DEMONSTRATION SOURCES

file name	version file name on floppy short description.
INITDEMO.PAS	- INITD.PAS Initialize the demo.
PETALDEMO.PAS	- PETALD.PAS Cycloid drawing display.
LINEDEMO.PAS	- LINED.PAS Line drawing display.
LIFEDEMO.PAS	- LIFED.PAS The game of life.
SEISDEMO.PAS	- SEISD.PAS Multi-pen chart recorder display.
GETSAVE.PAS	- GETSAV.PAS Get display from file to screen or save from screen to file.
CREATEWIN.PAS	- CREWIN.PAS Create entry in Screen package's window table.
SLIDER.PAS	- SLIDER.PAS Slide a window from one position to another on the screen.
JUST.PAS	- JUST.PAS Justify text with various fonts in a window.
WIPEWIN.PAS	- WIPWIN.PAS Wipe a picture into a window.
SNOOZE.PAS	- SNOOZE.PAS Pause for a specified period of time.
	Modules for SigGraph demo: MulDiv, Sleep, WindowLib, SigUtils, FontStuff.
WINDOWLIB.PAS	- WINLIB.PAS Window routine library.
SIGUTILS.PAS	- SIGUTI.PAS SigGraph 80 demo utilities.
FONTSTUFF.PAS	- FONTST.PAS Load and unload fonts for Just.

>DEMO.SIGGRAPH.BINARY - SIGGRAPH 80 DEMONSTRATION BINARY FILES

file name	version	file name on floppy short description.
:LINK	-	INITDEMO
INITDEMO.SEG	-	INITD.SEG
:LINK	-	PETALDEMO
PETALDEMO.SEG	-	PETALD.SEG
:LINK	-	LINEDEMO
LINEDEMO.SEG	-	LINED.SEG
:LINK	-	LIFEDEMO
LIFEDEMO.SEG	-	LIFED.SEG
:LINK	-	SEISDEMO
SEISDEMO.SEG	-	SEISD.SEG
:LINK	-	GETSAVE
GETSAVE.SEG	-	GETSAV.SEG
:LINK	-	CREATEWIN
CREATEWIN.SEG	-	CREWIN.SEG
:LINK	-	SLIDER
SLIDER.SEG	-	SLIDER.SEG
:LINK	-	JUST
JUST.SEG	-	JUST.SEG
:LINK	-	WIPEWIN
WIPEWIN.SEG	-	WIPWIN.SEG
:LINK	-	SNOOZE
SNOOZE.SEG	-	SNOOZE.SEG
WINDOWLIB.SEG	-	WINLIB.SEG
SIGUTILS.SEG	-	SIGUTI.SEG
FONTSTUFF.SEG	-	FONTST.SEG
FEATURES.SLIDE	-	FEATUR.SLI
SOFTWARE.SLIDE	-	SOFTWA.SLI
NETWORK.SLIDE	-	NETWOR.SLI
UCODE.SLIDE	-	UCODE.SLI
IO.SLIDE	-	IO.SLI
3RCC.SLIDE	-	3RCC.SLI
JUST.DEMO	-	JUST.DEM
GRAPH.PIC	-	GRAPH.PIC
3RCC.PIC	-	3RCC.PIC
BLANK.PIC	-	BLANK.PIC
WASHDC.PIC	-	WASHDC.PIC
NGR13.KST	-	NGR13.KST
MET22.KST	-	MET22.KST
DEMO	-	DEMO
DEMO1.CMD	-	DEMO1.CMD

>PERQFILE.USERLIBRARY - USER LIBRARY SOURCES

file name	version	file name on floppy short description.
CURSDESIGN.PAS	1.0	CURSDE.PAS Program used to design new cursors.
FONTED.PAS	-	FONTED.PAS Program used to create new fonts.
FONT2ED.PAS	-	FONT2E.PAS Part of Fonted.
FONTED.CURSOR	-	FONTED.CUR File of pictures needed by the font editor.
DR.MEMORY.PAS	-	DRMEMO.PAS Peek and poke into the memory manager tables.
TD.PAS	-	TD.PAS Demo of Grapics.
TD1.PAS	-	TD1.PAS Part of TD.
TD2.PAS	-	TD2.PAS Another part of TD.
MAZE.PAS	-	MAZE.PAS Draws a maze and runs a mouse through it.
GENMAZE.PAS	-	GENMAZ.PAS Creates a well-formed random maze.
MAZEPLAYER.PAS	-	MAZEPL.PAS Used by Maze to allow user to try to get through the maze.

PERQ Fault Dictionary

The Key to the PERQ Diagnostic Display

Three Rivers Computer Corporation
720 Gross Street
Pittsburgh, PA 15224
(412) 621-6250

Display	Description
000	Boot never got going, StackReset doesn't work or other major problem in the processor board (or clock).
001	Simple Branches fail.
002	Main Data Path Failure.
003	Dual Address failure on Registers.
004	Y Ram Failure.
005	Const/Carry Propogate failure.
006	ALU failure.
007	Conditional Branch failure.
008	Looping failure.
009	Control Store (or Write Control Store) failure.
010	Hung in Disk Boot.
011	Memory Data Error.
012	Memory Address Error.
013	Disk never became ready.
014	Couldn't boot from either disks.
015 - 020	Bad Interrupts Reading Floppy Disk Data.
030	VFY Hung.
050	Bad Error Message from VFY.
051	Empty stack bit not working.
052	Could not load TOS.
053	Push did not work.
054	Stack Empty did not go off.
055	Data error in push.
056	Empty or Full set when that is not the case.
057	Data error in bit 15 of the stack.
058	Stack empty set when the stack is full.
059	Data error on stack.
060	Data error after POP. Bit 14.
061	Data error after POP. Bit 13.
062	Data error after POP. Bit 12.
063	Data error after POP. Bit 11.
064	Data error after POP. Bit 10.
065	Data error after POP. Bit 9.
066	Data error after POP. Bit 8.
067	Data error after POP. Bit 7.
068	Data error after POP. Bit 6.
069	Data error after POP. Bit 5.
070	Data error after POP. Bit 4.
071	Data error after POP. Bit 3.
072	Data error after POP. Bit 2.
073	Empty wrong.
074	Data error after POP. Bit 1.
075	Data error after POP. Bit 0.
076	Empty not set after all pops.
077	Call test falied.
078	Odd didn't jump on a 1.
079	Odd jumped on a 0.

Display	Description
080	Byte sign didn't jump on 200.
081	Byte sign jumped on 0.
082	C19 didn't jump when it should have.
083	BCP[3] didn't jump when it should have.
084	C19 jumped when it shouldn't have.
085	BCP[3] jumped when it shouldn't have.
086	GTR didn't jump.
087	GTR jumped when it shouldn't have.
088	GEQ didn't jump.
089	GEQ jumped when it shouldn't have.
090	LSS didn't jump when it should have.
091	LSS jumped when it shouldn't have.
092	LEQ didn't jump.
093	LEQ jumped when it shouldn't have.
094	GEQ didn't jump on equal.
095	LEQ didn't jump on equal.
096	Carry didn't jump when it should have.
097	Carry jumped when it shouldn't have.
098	Overflow didn't jump when it should have.
099	Overflow jumped when it shouldn't have.
100	And-Not ALU function failed.
101	Or ALU function failed.
102	Or-Not ALU function failed.
103	And ALU function failed.
104	Or-Not ALU function failed.
105	Not-A ALU function failed.
106	Not-B ALU function failed.
107	Xor ALU function failed.
108	Xnor ALU function failed.
109	OldCarry-Add ALU function failed.
110	OldCarry-Sub ALU function failed.
111	OldCarry-Add /w No OldCarry failed.
112	Fetch error on Force Bad Parity.
113	Unexpected Parity error.
114	No parity errors on force bad parity.
115	Wrong address on force bad parity.
116	Upper 4 bit test failed.
117	MDX test failed.
118	Stack upper bits test failed.
119	Store/Fetch test failed.
120	Unexpected refill.
121	BPC test failed.
122	Fetch4 test failed.
123	Fetch4R test failed.
124	Store4 test failed.
125	Fetch2 test failed.
126	Store2 test failed.
127	NextOp test failed.
128	Fetch/Store overlap failed.

Display	Description
129	Bad interrupt Loc 4.
130	Bad interrupt Loc 14.
131	Bad interrupt Loc 20.
132	Bad interrupt Loc 30.
133	Data error on memory sweep.
134	Address error on memory sweep.
135	Field didn't work.
136	Dispatch did not jump.
137	Wrong Dispatch target.
138	Data error on inverted memory sweep.
139	Address error on inverted memory sweep.
150	Sysb not loaded correctly or hung.
151	Sysb did not complete.
152	Illegal Boot Key.
153	Hard Disk Restore Failure.
154	No such boot.
155	No interpreter for that key.
156	Interpreter file is empty.
157	Disk Error.
158	Floppy error.
159	Malformed Boot File.
160	CheckSum error in microcode.
161	CheckSum error in QCode.
162 - 168	Bad interrupts.
198	QCode interpreter microcode not entered correctly.
199	System not entered - calls or assignments don't work.
200	System entered, InitMemory to be called.
201	InitMemory entered.
203	SAT and SIT pointers set.
204	StackSegment number set.
205	Reading the BootBlock.
206	System version number set.
207	Head of free-segment-number list set.
208	First system segment number set.
209	System boot disk set.
210	System boot character set.
211	Boot block read.
212	Default heap segment number set.
213	First used segment number set.
214	Before setting freelists of data segments.
215	Before trying to allocate a segment number.
216	Temporary segment number allocated.
217	Ready to enter loop to find memory size.
218	Exited from memory size loop.
219	Restored mangled word.
220	Released temporary segment number.

Display	Description
221	Located segment adjacent to the I/O segment.
222	Modified the location of I/O segment.
223	Adjusted free memory.
224	Freelists of data segments set.
225	Set screen segment.
226	Header buffer allocated for swapping.
227	Status buffer allocated for swapping.
228	SwappingAllowed set false.
229	All boot-loaded segments set UnSwappable (if booted from floppy), InitMemory complete, ready to return to System.
230	Starting to increase number of segments allowed (because memory is larger than 1/4 megabyte).
231	Changed maximum of SITSeg.
232	Changed size of SITSeg.
233	Changed maximum of SATSeg.
234	Changed size of SATSeg.
235	Created new unallocated segment numbers.
236	Finished InitMemory.
300	InitIO to be called.
301	InitIO entered.
302	KeyEnable set false.
303	Buffers allocated.
310	InitInterruptVectors to be called.
320	InitInterruptVectors complete, InitDeviceTable to be called.
322	Starting to initialize ETHERNET.
325	ETHERNET initialized.
330	InitDeviceTable complete, InitScreen to be called.
340	InitScreen complete, InitTablet to be called.
350	InitTablet complete, InitCursor to be called.
360	InitCursor complete.
361	Interrupts are now off; about to send device table to microcode.
363	Got control block for Z80 speech.
364	Set up video registers.
365	Screen is now started.
366	Got control blocks for keyboard, RS232, and GPIB.
368	Microcode returned.
369	Interrupts are now turned on.
370	Microcode informed that the device table has been initialized, IO microcode initialization complete, LocateDskHeads to be called.
371	LocateDskHeads entered, buffers allocated.
373	Disk heads at cylinder 0 or disk broken.

Display	Description
374	Disk heads at cylinder 0 (not broken).
375	Microcode instructed to consider current position as cylinder 0.
376	Dummy read of cylinder 0, sector 0 complete, about to dispose buffers and exit LocateDskHeads.
380	LocateDskHeads complete, FindSize to be called.
381	FindSize entered and buffers allocated.
382	Disk access attempt returned.
383	Size of disk determined, about to dispose buffers and exit FindSize.
390	FindSize complete.
400	Keyboard enabled.
410	InitGPIB to be called.
411	InitGPIB entered, buffers allocated.
412	First GPIB command built.
413	First GPIB command sent to Z80.
414	Second GPIB command built.
415	Second GPIB command sent to Z80, about to dispose buffers and exit InitGPIB.
420	InitGPIB complete.
499	Clock enabled, about to exit InitIO.
500	InitIO complete, InitStream to be called.
600	InitStream complete, FSInit to be called.
700	FSInit complete.
800	Command file and Console opened, InitExceptions to be called.
810	InitExceptions complete.
820	System version number set.
822	Current 60 Hz. clock value read.
824	60 Hz time reference set, TimeStamp time reference to be set.
900	FSSetUpSystem to be called.
950	FSSetUpSystem complete.
951	About to enable swapping (if booted from hard disk).
952	FSLocalLookup and EnableSwapping complete.
999	System fully initialized, system title line to be printed.

Three Rivers Computer

SOFTWARE REPORT FORM

SOF-1 Orig. 5-81

Reference # (Supplied by Submitter)

For 3RCC Use Only:	REQ.#:	Date Recd:	Date Processed:

FROM

Person to Contact

Company Name

Customer Address

City/State/Country ZIP

Phone Date

HARDWARE

PERQ Model # _____

PERQ Serial # _____

Memory Size _____

Options:

Floppy ☐ Network ☐ WCS ☐

DISK SIZE

☐ 12

☐ 24

☐ 48

O.S. Version # _____

Subsystem, Program, Module or Manual Name: _____

Version # _____

REPORT TYPE

- ☐ Logic/Coding Error
- ☐ Documentation Error (Page #_____)
- ☐ Suggestion for Improved Efficiency
- ☐ Request for Software Modification
- ☐ Inquiry
- ☐ For Your Information

PRIORITY

- ☐ CRITICAL
- ☐ HIGH
- ☐ STANDARD
- ☐ LOW

Program demonstrating problem attached:

☐ on floppy (preferred) ☐ hard copy ☐ Not Avail. Not Applic

Can this problem be reproduced at will? ☐ Yes ☐ No

DESCRIPTION OF PROBLEM:

CUSTOMER _____ PERQ SERIAL NUMBER _____

PERQ Customer's Report & Reader's Comments

Packing

What was the condition of the packing when you received your PERQ? Please be specific?

_____

_____

_____

_____

Missing Materials

What (if anything) was listed on the packing slip but not received?

_____

_____

_____

PERQ's Overall Condition

Please describe any damage(s) to the PERQ system when received. Be specific.

BASE _____

_____

_____

DISPLAY _____

_____

_____

KEYBOARD _____

_____

_____

TABLET (BIT PAD) _____

_____

_____

OTHER _____

_____

_____

Clarity of Installation Instructions

Were you able to easily unpack and install the system?

_____

_____

_____

_____

Please note below any suggestions or comments which you feel may be beneficial to us in better serving our customers.

Please designate below a software and hardware person in your organization whom we can contact for information or problems.

Name _____ Title _____

SOFTWARE CONTACT

Company/Organization _____ Telephone Number _____

Address _____

Address _____

Name _____ Title _____

HARDWARE CONTACT

Company/Organization _____ Telephone Number _____

Address _____

Address _____

Please comment on the documentation manual that you received with your PERQ. Did you find it understandable, usable, and well organized? Did you find any errors? If so, identify the page number and text.

Prepared/Completed by: _____ / /
Name Title Date

THANK YOU - THREE RIVERS COMPUTER CORPORATION
720 Gross Street
Pittsburgh, PA 15224
(412) 621-6250