Getting Started

First, turn your Perq on. There is a switch on the front of the "base unit" push it to the right to power up the system. You should hear the fans come on and the Disk start to spin up. The disk takes TWO MINUTES to spin up to speed, during which time the Perq runs a memory diagnostic. The screen will not be stable while this is happening. After the disk spins up, the Boot program will read the system into memory, with additional diagnostics. The diagnostics show up as a few flashes on the screen. Boot takes about 6 seconds with some variation depending on where the disk heads are when you boot. If you wait for more than two minutes and the Perq does not come up, try pressing the boot button, located on the back of the keyboard. If after several trys it wont come up, something is probably wrong and you should call your maintenance person (see Diagnostics section).

As of 10/24/80 some systems did not power up boot completely the first time. This problem shows up as a stable screen, but no greet message. Booting a second time will clear the problem.

When typing commands to most programs the following editing keys can be used:

```
BACK SPACE (or Control-H) erases the last character typed
OOPS (or Control-U) erases the last line typed
Control-C once aborts the current program the next time
             it reads input from the keyboard
Control-C twice aborts the current program immediately
Control-S stops type out
Control-Q resumes type out after a Control-S
Control-Shift-C aborts an indirect command file
Control-Shift-D types out a stack trace back
```

Errors while executing cause an informative message to print. Fatal errors print a stack trace back and exit to the command interpreter. Currently all IO errors, recoverable or not, print messages.

Command Interpreter

The user interface to the operating system is the command interpreter. The prompt for the command interpreter is a colon. Most commands are a keyword or the name of any runnable program.

Keywords can be abbreviated to uniqueness. The current commands are:

HELP	Type a helpful message
?	Print the List of known keywords
EDIT	Start the Editor
COMPILE	Start the Pascal compiler
LINK	Start the Linker
FILEUTILITY	Start the FileUtility
DIRECTORY	Start the FileUtility and issue a Directory command
TYPE	Start the FileUtility and issue a Type command
RENAME	Start the FileUtility and issue a Rename command
DELETE	Start the FileUtility and issue a Delete command
PREFIX	Set the prefix string for all file names
PAUSE	Print out the Command line and wait for a RETURN

When the command interpreter decides to run a program it calls the loader which types out some interesting information about the segments it loads as it puts them into memory.

The command interpreter can also work from an indirect command file. Type @ <space> FileName. The command file may have another command file specified as the last command in the file. You can even set up a loop.

The PAUSE command is especially useful in command files. It will print the command line on the terminal and wait for a confirming return. This can be used for prompts to change floppy disks, cables, etc.

The Pascal Compiler

The Keyword COMPILE and the run file PASCAL call the Perq Pascal Compiler. The command line can be typed as a response to the compiler's prompt (an asterisk), or can be given on the original command line. The general form of the command is OutputFile=InputFileList. OutputFile is the binary output, default extension is .SEG. InputFileList is one or more .PAS files separated by commas. If only the OutputFile is given, the input file name is assumed to be the same as the output file.

A switch may be used on the command line. The only important switch is /SYMBOL:nn (may be abbreviated to /S:nn) which sets the number of in-core symbol table blocks. Use /S:14 for any large Pascal program.

Examples:

```
:COM MyFile                   {Abbreviate COMPILE to COM, behaves as
                              MyFile.SEG=MyFile.PAS}
:PASCAL
Loading segment 35 (PASCOMP.SEG)   <--- Main Program
Loading segment 26 (BODYPART.SEG)
Loading segment 25 (DECPART.SEG)
Loading segment 24 (COMPINIT.SEG)
Loading segment 27 (EXPEXPR.SEG)
Loading segment 36 (CODEGEN.SEG)
Perq PASCAL Compiler QB.1
*MyOutFile=MyInFile1,MyInFile2
```

The compiler input syntax is defined by the Pascal Report modified by the Perq Pascal Extensions Manual.

The Linker

The output of the compiler is a .SEG file. Each module or program is a separate seg file. The linker creates a .RUN file by looking at the declarations and Imports of each segment and computing appropriate linkage information and a prototype stack. The linker is smart enough to follow nested Imports declarations to determine which segments need to be loaded for a program to run. Thus, one usually only needs to tell the linker the name of the output file and the name of the main program input file. The command syntax is:
OutputRunFile,MapFile=InputFileList.
The command can be given to the command interpreter (i.e. LINK FOO=FOO) or in response to a Linker prompt (asterisk). The MapFile is optional. The InputFileList may have one or more .Seg files separated by commas. FileExtensions (.RUN,.SEG,.MAP) are defaultable. As in the compiler, the command can be abbreviated to a single file (LINK FOO), which behaves as if LINK FOO=FOO was typed.

The Linker will process all files on its InputFileList to satisfy Import Module declarations before it uses the "from" clause of the import declaration to find a module. You can exploit this feature to create test modules which have the same module name but different .SEG file name, without changing the import declarations.

The command line may include one or more switches. The current switches are

```
/VERBOSE      - List the imports of each module as it processes it
/SYSTEM       - Link this program as a Stand alone (SYSTEM) program
/USER         - Link this program as a normal (user) program
                (this is the default)
/VERSION:nn.  - Link this program for use with version nn.
                  of the system
                  (note- use a period after the number
                         if it is decimal)
```

A word about versions. Any module needed by the system is loaded at boot time and stays around forever. If a user program needs a module that is part of the system, the linker generates linkage information from the System.nn.Run file. The default version is the version of the operating system currently executing. Any time a new system is linked, all programs must be relinked before they can be run under the new system. Since the linker is the only program which MUST be relinked BEFORE a new operating system is installed, the LINK run file has a version in it (i.e. LINK.nn.RUN). The procedure for installing a new system is to relink the system using /SYSTEM and /VERSION switches, relink the linker using /VERSION, creating a SYSTEM.nn.BOOT file and rewriting the boot file on the disk (See WRITEBOOT documentation for more info). After the new system is booted, all other programs can be relinked using the new linker.

FileUtility

The FileUtility (read PIP) messes around with the file system in all the usual ways. It is currently not plush. Commands can be given on the Command Line, or in response to an asterisk prompt. Commands are keywords followed by parameters. Keywords can be abbreviated to uniqueness. Any parameter omitted will be prompted for. When in doubt, type keyword<RETURN>, it will prompt you for all its parameters.

Available commands are:

HELP	Print something like this table
DIRECTORY	List File Names. One wild card (*) is allowed.
TYPE	Type out a file on the console
COPY	Copy Destination to Source
DELETE	Remove (It's gone) a file
RENAME	Old to New Name
INITDISK	Initialize the Disk *****WARNING**** this command will WIPE the disk clean.
CREATE	A file in the directory
BLKREAD	Read any block of any file
BLKWRITE	Write any block of any file
PREFIX	Set the default prefix for all file names
SCAVENGE	Recreate the directory
FTP	Start the File Transfer Subsystem
APPEND	Concatenate two files together
QUIT	Get out of FileUtility

The current file system (POS file system) is very temporary and clearly a hack. There are a fixed number (18Ø) of files with a fixed maximum length (128 blocks of 512 bytes). The disk space is preallocated for all files at the maximum size. The blocks for a file are numbered 1 to 128. Block number -1 is the Header block which has the name of the file, it's current size, etc. The directory is file number 1 and contains a copy of the information in each Header Block. The directory is recreatable by reading all headers (which are in known locations on the disk). File Ø is the boot file, it contains Boot microcode, and a memory image for the initial system as well as the running QCode microcode.

The FTP subsystem can copy files across the RS232 link to another Perq or any other computer that can run its own version of the FTP protocol. The FTP subsystem has its own commands which are:

```
GET ThereFileName
PUT HereFileName ThereFileName
MODE ModeSetting (Currently Perq or PDP11, default PERQ)
BAUD BaudRate (initial setting is 9600)
POLL (Goes into poll mode, waiting for Put/Get from the other machine)
```

One machine must be in poll mode while the other does Get/Put.

Patch

Patch will peek and poke on a disk file. Run Patch and play around. Someone will write more info on it soon.

Dr.Memory

Run it. It will print some interesting memory state information, then let you play around. There is a password to let you execute some dangerous procedures.

FileList

FileList will make a text file of a directory search with specified strings between file names. Used to help make indirect command files. Self prompting.

PLX

PLX is a program which reads and writes RT-11 format Floppy Disks. In Response to its PLX> prompt you can type a number of command (including HELP). The most usefull are
DIRECTORY - Print a directory of the Floppy
PUT - Copy a file from the Perq Hard Disk to the floppy
GET - Copy a file from the floppy to the hard disk
PLX will work with Single and Double sided disks (Do NOT use Double Density mode yet). The command SIDES, which takes 1 or 2 as its argument, sets the number of sides to use.

PLX has a verify mode which will do read after write verification of each transfer, as well as a VERIFY command which will compare the floppy file tothe disk file and report differences.

An indirect command file facility is also part of PLX. Type @ FileName (no separating space). There is a confirmation switch in PLX which asks for confirmation before overwritting files. It can be turned off for command files if you wish (SAFE/FAST commands).

See Perq.Files for a usefull way to maintain floppy back-ups of files.

WriteBoot

This program let you write a boot file. The boot file has 3 parts, the boot microcode, the System QCode, and the System microcode. WriteBoot first asks whether you want to write a boot file, if you answer no, WriteBoot will verify the boot file. Then it asks if you want to write a boot file on the hard disk. A No answer will write on the Floppy Disk. WriteBoot then will prompt for the file names for each of the three sections. For microcode, more than one file is usu-ally required. WriteBoot will continue prompting for microcode files until a blank line is typed, after which it will go on. The default extension for microcode files is .BIN The QCode file is a .BOOT (SYS-TEM.xx.BOOT) where xx is the version number. WriteBoot asks if you are making a system and if so prompts for the version number and makes the filename out of it. If you answer no to the question, WriteBoot prompts for the file name of a boot file, which may be any bootable program.

To make a BOOT file, link the program with the /SYSTEM switch. Then "run" the program; the loader will create the boot file. Note that a program must follow a number of conventions to be made boot-able.

DiskUp

DiskUp is a disk fixit program which will scan the disk looking for unreadable blocks. It has a command to FIX a block by rewriting the Logical Header. It can also reformat a track of the disk if it gets clobbered. DiskFix will also do a Scavenge, etc. There is a Help Command, try it for a list of available keywords. BE CAREFULL, DiskUp can ruin your data.

Chatter

A program which makes a Perq into a terminal on another system using the RS232 port. Chatter will send anything you type on the keyboard out the RS-232 port and will display on the screen any characters it receives.

FloppyDup

Duplicates floppy disks by coping the data onto the hard disk, and then onto a fresh floppy. You need nine POS files free to run FloppyDup

QDis

A Dissassebler for Q-Code

PrqDis

A Disassembler for microcode

HpPrint

Prints a text file on the HP7310A printer (PRQ-GRP-ØØ1-A)

ScreenDump

Makes a copy of the screen on the HP7310A printer (PRQ-GRP-001-A) Note: the printer image is 720 dots by 1003 lines, the screen image is 768 by 1024.

Demo Programs

Several nifty demo programs are included on your disk. Most have Help information displayed on the screen.

LIFE	A Version of John Conway's population game
LINE	A neat line drawing pretty picture program
PETAL	Another line drawing pretty picture program
KAL	More pretty pictures
EF1	Ditto
KINETIC	Ditto

Diagnostic Programs

TestFloppy will run a test of your floppy drive (and wipe out the floppy data on the floppy in the drive). It is a convenient way to format a floppy disk.

RS232Test will run an echo test of the rs232 channel.

Microcode Programs

PrqMic and PrqPlace are the microcode assembler/placer programs respectively. PrqMic takes a .MICRO source file and produces .REL and .RSYM output files. These files are used by PrqPlace to figure out where to put each instruction in the Writable Control Store. Prqplace writes a .BIN file. BIN files are used by OdtPrq, the microcode debugger (and WriteBoot). OdtPrq is a simple minded debugger for microcode. It needs two Perqs and a Link between them (currently a special piece of hardware, but soon just the RS232 channel). Also included is DmpPrq, a heavy duty Crash Dump Analyzer for Pascal Programs (which needs the same environment as OdtPrq). More stuff on these programs is forthcoming.

On your Disk is the following Microcode source files (and .BIN)
PERQ is the QCode interpreter
IO   is the IO microcode
RO   is RasterOp
LINE is the Line Drawer
SYSB is the System Microcode Loader for the Rigid Disk
FLOPSYSB is a version of SYSB for a floppy
KRNL is the OdtPrq Kernal microcode
VFY is the boot diagnostic
H is a null diagnostic needed on boot floppys
LINK is microcode for the machine running ODTPRQ to talk to KRNL.
DTST is a microcode disk test program runnable by OdtPrq
It wipes your disk (Also an easy way to format a disk).