PERQ
Systems
Corporation

ON-LINE DEBUGGING TECHNIQUE

FOR THE PERQ (ODTPRQ PROGRAM)

March 1984

This manual describes ODTPQR Version 8.4 and is for use with
POS Version 6.5 and subsequent releases until further notice.

PERQ Systems Corporation
2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

ON-LINE DEBUGGING TECHNIQUE

FOR PERQ (ODTPRQ PROGRAM)

ODTPRQ, On-line Debugging Technique for PERQ, is the program used to debug code or hardware on a PERQ testbed.

1. SETUP

Two PERQs are connected by link cables between their Link or OIO boards. The master must be a PERQ capable of running stand-alone. The testbed must be at least a PERQ cardfile, power supply, CPU, memory and link (or OIO). Care should be taken that the master and testbed are both plugged into grounded outlets with grounds at the same potential.

If no IO board is used in the testbed, the IO MEM RQST line on the backplane must be jumpered low. JA of the master is connected to JB of the testbed and JB of the master to JA of the testbed by 50-pin ribbon cables. The master is then powered up. The testbed must not be powered up until the master has booted. When the testbed is powered up, its boot code will detect that a link is active and will not try to boot from the hard disk or floppy. At this point ODTPRQ is run on the master. ODTPRQ takes one optional parameter when run, which is a state file name (see GetState).

2. COMMANDS

Command names may be abbreviated. Parameters may be entered in order on the same line separated by spaces or commas. Defaulted parameters are entered as two commas. When numbers are entered, they are interpreted as octal unless followed by a period, in which case they are interpreted as decimal. Commands may be entered from three sources: the keyboard, the tablet or a command file. If a command is issued from the keyboard with no parameters, the user will be prompted for them. A menu is displayed between the top and bottom screen windows for entering commands from the tablet.

When a location is opened, its current value is displayed, and its contents may be modified. When a numeric form is allowed as a parameter, it consists of up to 3 characters which specify how numbers are to be displayed. These characters specify base, size and mode where base is o, d or c (octal, decimal or character), size is w or b (word or byte) and mode is s or u (signed or unsigned). The defaults are owu.

Help
or HELP key — Print help listing.

Quit — Return to shell.

Boot — Boot the testbed by loading the file KRNL.BIN into the testbed microstore and running it.

Memory <addr> <form>
Open a memory word. After opening the location the user may type a new value or leave the value unchanged. A trace can be put on the value by typing w (see the Watch command). The user may end the command with return, LF to open the next location, ^ to open the previous location or = to prompt for the fields of the value in order to change them. The memory command only has a memory value field.

Register <addr> <form>
Open a register.

Variable <routine> <offset> <form>
Open a Pascal variable in memory.

Global <offset> <form>
Open a Pascal global variable in memory.

Virtual <offset> <form>
Open a virtual memory variable.

UCode <addr> — Open a microinstruction. When = is used to change the value, the user is prompted for each of the 12 fields of the microinstruction: X, Y, A, B, W, H, ALU, F, SF, Z, CND, JMP. The numeric form is always octal. Note that the microstore cannot actually be read, so a copy of its contents is kept in the master PERQ. This means that self-modifying code will not be displayed correctly.

Watch <name> <x> <y>
This causes a trace to be placed on the current open location. Whenever the KRNL code is entered after the user program exits, the value of the location is displayed, along with the user-specified name in the top window of the screen. The range of x is 0..3 and y is 0..24; however y=24 should not be used since it is the

bottom of the screen window (and hence causes scrolling).

Clear — clear the low 128K words of memory.

Load <file> — initial load of a micro binary. Assumes the .BIN extension.

Overlay <file> — load a micro binary over previously loaded code.

QLoad <RootFile> <char> <RunFile> <boot> MBoot>
Load Q-code (Pascal) program. This is used to load a system onto the testbed. The RootFile name provides default names for the RunFile, Boot file and MBoot file although the defaults can be overridden. The char is the boot character (a..z or A..Z).

BLoad <file> <addr>
Load binary file into memory. This loads a file into memory without any checking or conversion.

ListSegments — List Q-code segments currently loaded.

Go <addr> — Start testbed executing at the specified micro address.

Break <addr> — Set microcode breakpoint. This replaces the microinstruction with a jump to the KRNL. When proceeding from a breakpoint, ODTPRQ replaces the original instruction and puts a breakpoint on the next instruction, executes it, then puts back the breakpoint and the next instruction. This means that the breakpoint instruction is not executed either immediately after its predecessor or immediately before its successor (unless the breakpoint is killed before proceeding). Therefore, the breakpoint instruction cannot use or set the condition codes, shifter, hold bit, jumps whose target is not specified in the instruction, interrupts, etc.

KillBreak <addr> — Clear microcode breakpoint.

QBreak <routine> <instruction>
Set Q-code breakpoint.

QKillBreak <routine> <instruction>
Clear Q-code breakpoint.

ListBreaks List active breakpoints.

Proceed <addr> Proceed from recent breakpoint.

Dump <DumpCommand>
Execute a dump command. If Dump is given with no parameters, the Dump subsystem is entered (see below).

@<file> Read alternate command file. Command files cannot be nested but may be chained. An @ in a command file will start a second command file and close the first one.

ListFile <file> Write alternate list file. All ODTPRQ output will go to the file until another ListFile command. Typing return when ListFile prompts for a file name causes the output to go to the screen.

SaveState <file>
Save OdtPrq state on a file. Note that this does not save the testbed state. It saves the state that ODTPRQ thinks the testbed is in. It also saves any watches, breakpoints and Qcode segment names that it knows about.

GetState <file>
Get OdtPrq state from a file. This does not load anything into the testbed. It loads ODTPRQ with a previously saved state. SaveState and GetState are useful when a user wants to exit from ODTPRQ and return to it without disturbing the testbed. If ODTPRQ is run with a state file name as a parameter on the command line, then a breakpoint which occurred while ODTPRQ was not running will be correctly displayed.

Debug <on|off> Controls ODTPRQ's internal debug mode. Leave it off.

/ Open current location.

(LF) Open next location.

^ Open previous location.

= Change currently open location.

3. DUMP COMMANDS

When a Dump command is issued, ODTPRQ enters the Dump subsystem. This is signaled by the prompt changing from '>' to 'DUMP>'. The commands which may be entered in this mode are described below. A ListFile command stays in effect during Dumps.

Help or HELP key	Print Dump subsystem help.
Quit	Leave Dump subsystem.
All	Print all dumps.
Memory <FirstAddr> <LastAddr> <form>	Dump an area of memory.
Registers <FirstAddr> <LastAddr> <form>	Dump an area of the XY register file.
Stack	Do a memory stack trace back. At each level, the user is prompted with 'Print Info?:'. Answering 'y' gives information about that level of the stack.
Trace	Do a brief memory stack trace back. No 'Print info?' prompts.
MTables	Dump the memory manager tables.
IOTables	Dump the I/O tables. This only works on POS F.1 or earlier versions of the operating system.

4. MISCELLANEOUS

ODTPRQ accepts 20-bit memory addresses and displays 20-bit register values, but will not write 20-bit values to a register.

When debugging microcode, it is sometimes useful to use a logic analyzer for tracing execution. The JC connector of the CPU brings out all of the micro address lines and the clock. JA and JB of the CPU carry the micro instruction.

Since the Clear command does not write to all of the address range of large memory boards, it is sometimes convenient to load and run the VFY microcode to eliminate the possibility of parity interrupts if an uninitialized upper memory location is read. To do this, issue the ODTPRQ commands:

```
>load VFY
>r0/2                    {set R0 to 2}
>go 4001
```

When running a POS operating system, a Qbreakpoint set at SCROUNGE 0 0 will get any uncaught exceptions.

In POS, Mem402 is the hardware configuration. Its value is normally initialized by the SYSB microcode at boot time. Bit 0 is set for a 16K CPU, bit 6 is set for a portrait screen and bit 11 is set for an EIO board. A PERQ is typically 100 (portrait screen, 4K CPU, IOB) and a PERQ2 is typically 4101 (portrait, 16K CPU, EIO).

The KRNL uses R370 to hold the breakpoint number when it is entered at location 7401. This is one way for microcode to report errors. Some values of breakpoints are pre-defined:

```
1..19: Microcode breakpoint
20:    Seg Fault
21:    Stk Ovl
22:    Run Err
23:    IO Seg Fault
24:    Memory parity error
30:    Krnl detected a bad command
31:    Krnl detected a bad interrupt return
32:    Krnl detected a bad interrupt
>1023: QBreakpoint
```

Setting bit 0 of R360 prevents the KRNL from serving display interrupts.

Changing R371 moves the screen base in the KRNL display code.