

# Chapter 43 - Debugging and Profiling Cookbook

IE Mon is the workbench. IE Script is the repeatable notebook. This chapter shows common tasks rather than another full command reference.

Use Chapter 33 when you need the command table. Use Chapter 34 when you need the full script module list.

## 43.1 Stop At A Known Address

The smallest debugging loop is still:

```
(ie64)> d 1000 #4
(ie64)> b 1018
(ie64)> r pc 1000
(ie64)> g
(ie64)> r
(ie64)> bc 1018
```

`d` proves the bytes. `b` sets a breakpoint. `r pc 1000` sets the entry point. `g` runs until the breakpoint. The final `r` shows what changed.

If the programme runs past the place you expected, set the breakpoint on the self-loop or on the first MMIO write after the setup sequence.

## 43.2 Catch A Bad MMIO Write

Use a write watchpoint when you know the exact address:

```
(ie64)> bpwdw F0044
(ie64)> g
(ie64)> wl
(ie64)> wc F0044
```

`bpwdw` watches a 32-bit write. The monitor also has byte, word, and quad forms:

Command	Width	Access
<code>bpmbw addr</code>	byte	write
<code>bpmww addr</code>	word	write
<code>bpwdw addr</code>	long	write
<code>bpmqw addr</code>	quad	write
<code>bpmdr addr</code>	long	read
<code>bpmd addr</code>	long	read or write

Use `wl` to list watchpoints and `wc addr` to clear one.

## 43.3 Find Who Touched A Region

---

When you do not know the exact access, turn on the access log:

```
(ie64)> accesslog on #64
(ie64)> g
(ie64)> accesslog show #10
(ie64)> who wrote F0044
(ie64)> accesslog off
```

`accesslog` records recent reads, writes, and instruction fetches. `who wrote addr` asks for the last recorded writer of one address.

Some access tools require CPU and bus access instrumentation. If the monitor says the service is unavailable, fall back to a normal watchpoint or a smaller breakpoint-driven session.

## 43.4 Use I/O Views

---

The `io` command reads a named device view. It is often clearer than a raw memory dump:

```
(ie64)> io video
(ie64)> io blitter
(ie64)> io voodoo
(ie64)> io audio
(ie64)> io coproc
(ie64)> io midilive
```

Use I/O views after a setup routine and before running the next part of the programme. They answer the basic question: did the hardware receive the state I meant to write?

## 43.5 Symbols

---

Symbols let the monitor use names in address expressions:

```
(ie64)> sym add loop 1018
(ie64)> sym lookup loop
(ie64)> b loop
(ie64)> d loop #2
```

You can also resolve an address:

```
(ie64)> sym resolve 1018
```

For PRG-style work, `sym add` is usually enough. The monitor also knows how to load label and ELF symbol files, but the reader path in this guide does not require those files.

## 43.6 Reverse Step

---

`bs` or `rs` steps backwards through the CPU-local timeline. Use `history horizon` when you want to inspect the whole-machine reverse snapshot horizon.

```
(ie64)> g
(ie64)> bs
(ie64)> r
(ie64)> history horizon
```

Reverse history is a debugging aid. Do not use it as a timing device. If the bug depends on an interrupt or a VBlank edge, record the device status registers as well as the CPU registers.

## 43.7 Automate A Repro With IE Script

This script types a short BASIC programme, waits for output, then asks the video module for a frame hash.

```
term.type_line('10 SCREEN 1')
term.type_line('20 PALETTE 1,255,0,0')
term.type_line('30 PLOT 10,10,1')
term.type_line('RUN')
term.wait_output('Ready', 2000)
sys.wait_frames(2)
sys.print(video.frame_hash())
```

Use this pattern when a bug needs the same setup every time. The script does not replace the programme. It presses the keys, waits for the machine, and records the observation.

## 43.8 Capture Output

For terminal repros:

```
sys.capture_output('run.log')
term.type_line('RUN')
term.wait_output('Ready', 2000)
sys.capture_output_off()
```

For video repros, prefer a stable frame hash before saving a screenshot:

```
sys.wait_frames(3)
h = video.frame_hash()
sys.print(h)
rec.screenshot('frame.png')
```

The hash is the quick comparison. The screenshot is the human check.

## 43.9 Use Performance Reports Carefully

`sys.perf_reset()` and `sys.perf_report()` measure instrumented subsystems when performance accounting is active:

```
sys.perf_reset()
sys.wait_frames(60)
report = sys.perf_report()
sys.print(report)
```

An empty report means either performance accounting is off or no instrumented path ran during the measured span. A non-empty report is a guide to where time went. It is not a promise that the same programme will take the same time on every machine.

## 43.10 A Practical Debug Order

---

When a programme misbehaves, use this order:

1. d the code or LIST the BASIC programme.
2. Set one breakpoint at the first wrong-looking step.
3. Dump the memory or I/O view before and after the step.
4. Add one watchpoint only if the writer is unknown.
5. Use `accesslog` or `who` only when watchpoints are not enough.
6. Check `history horizon` if reverse snapshots matter to the fault.
7. Turn the session into an IE Script only after the manual steps are understood.

The fewer moving parts in the debug session, the more likely the answer is the real machine state.