

# Chapter 47 - Driving The Hardware From IE Script

---

IE Script is not a different machine. It is a lab bench beside the same bus. A script can stage bytes, freeze a CPU while copying a large block, write MMIO registers, wait for frames, and collect observations.

The supplied `rotozoomer_ies.ies` script drives the same VideoChip and MIDI player used by the BASIC demo. The difference is clarity: names, functions, and loops make it easier to run the same experiment many times.

## 47.1 Register Names In A Script

---

The script begins by naming the registers it will write:

```
local VIDEO_CTRL = 0xF0000
local VIDEO_MODE = 0xF0004
local BLT_CTRL = 0xF001C
local BLT_OP = 0xF0020
local BLT_SRC = 0xF0024
local BLT_DST = 0xF0028
local BLT_WIDTH = 0xF002C
local BLT_HEIGHT = 0xF0030
local BLT_MODE7_U0 = 0xF0058
local BLT_MODE7_DU_COL = 0xF0060
```

These are not script-only names. They are the same MMIO addresses from Chapter 4 and Appendix D. The script is only giving them labels.

## 47.2 Staging Assets

---

Large textures and MIDI files are copied into guest memory before the effect starts:

```
local texture = sys.read_file("ROTO.RAW")
cpu.freeze()
mem.write_block(0x600000, texture)
cpu.resume()

local midi = sys.read_file("SONG.MID")
cpu.freeze()
mem.write_block(0xA40000, midi)
cpu.resume()
```

Freezing the CPU is a courtesy to the running machine. It prevents a CPU from reading a half-written asset while the script is staging it.

## 47.3 Starting MIDI

---

The script starts MIDI by writing the player registers:

```
mem.write32(0xF0BA0, 0xA40000)
mem.write32(0xF0BA4, string.len(midi))
mem.write32(0xF0BB4, 180)
mem.write32(0xF0BA8, 1 + 4)
```

1 starts playback. 4 requests looping. The volume register is the same MIDI\_VOLUME register used in Chapter 21.

## 47.4 Rendering A Frame

The script version writes the Mode 7 registers directly:

```
mem.write32(0xF0020, 5)
mem.write32(0xF0024, 0x600000)
mem.write32(0xF0028, 0x900000)
mem.write32(0xF002C, 640)
mem.write32(0xF0030, 480)
mem.write32(0xF0058, u0)
mem.write32(0xF005C, v0)
mem.write32(0xF0060, du_col)
mem.write32(0xF0064, dv_col)
mem.write32(0xF0068, du_row)
mem.write32(0xF006C, dv_row)
mem.write32(0xF0070, 255)
mem.write32(0xF0074, 255)
mem.write32(0xF001C, 1)
```

The command is less friendly than `BLIT MODE7`, but it exposes exactly what BASIC writes for you.

## 47.5 Waiting And Measuring

The supplied script waits for the blitter by polling the busy bit, then waits for the next display frame:

```
while bit32.band(mem.read32(0xF001C), 2) ~= 0 do
  sys.wait_ms(1)
end

sys.wait_frames(1)
```

For automated experiments, add a frame hash:

```
sys.wait_frames(2)
sys.print(video.frame_hash())
```

The hash is useful when you are changing parameters and want to know whether the final picture changed.

## 47.6 When To Use IE Script

Use IE Script when:

- The setup is repetitive.
- Assets have to be staged before every run.
- You want a frame hash or a screenshot after a known number of frames.

- You need to freeze, inspect, and resume the machine.
- You are comparing two versions of the same effect.

Use BASIC when you are still learning the effect. Use IE Mon when you are entering or debugging small machine-code sections.

Chapter 48 converts the BASIC floating-point motion into the tables used by the native rotozoomers.