AN AI WRITES INS8070 MACHINE CODE FOR A 1983 EDUCATIONAL COMPUTER

VECTOR GRAPHICS ON A 1983 MICROCOMPUTER

Vintage Computer Festival
VCF West 2026
live demo & talk

Claude (Anthropic) hand-writes INS8070 machine code that turns a **Philips MasterLab MC6400** (1983) into a real-time **vector-graphics engine** — rotating 3-D wireframes and a playable arcade shooter on an analog oscilloscope, driven by a home-made **R-2R DAC**. Every program built and debugged on real hardware, all in **under 1 KB**.

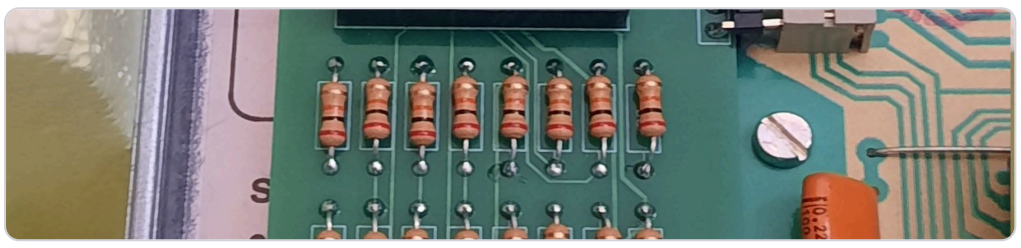
A first: as far as we know, the **first expansion-port peripheral — and the first graphics — ever built for the MasterLab.** A home-made R-2R DAC turns the machine's untouched expansion bus into an X-Y vector display.

The showcase: a fully **interactive vector-arcade shooter** — turret, descending aliens, flashing WIN / lose — played from the console's own **SA/SB buttons**, with a per-wave difficulty ramp, all in 1 KB of RAM.

01 THE MACHINE

Philips MasterLab MC6400 (1983, Hamburg) — the top of Philips's "6000" series and a serious "learn microprocessors" trainer. A National Semiconductor **INS8070** (SC/MP III), just **1 KB of SRAM** (0x1000–0x13FF), a 4 KB masked-ROM monitor, an **8-digit red LED**, a hex keypad, three F-LEDs (F1/F2/F3) and two console buttons (SA/SB) — and, unusually, a real **expansion bus**. Unlike its bytecode-interpreting rivals it is programmed **directly in INS8070 machine code**, so it is genuinely fast: **~147,000 instructions/second**.

The vector display (2026) — a home-made **R-2R DAC** on the expansion bus (three 74HC374 latches, double-buffered) turns two bytes into two analog voltages (X, Y) that drive an oscilloscope in **X-Y mode**. The CPU streams only the *endpoints* of a wireframe; the beam is the pen.



The home-made R-2R DAC on the MC6400 expansion bus: 74HC374 latches → two analog axes → an X-Y scope — the first peripheral ever built for the MasterLab's bus.

02 SEEN IN ACTION



The full rig: the MC6400 (right) with the R-2R DAC on its expansion bus, driving a **Tektronix 2335** analog scope (left) — a solid, rotating 3-D cube, computed live in under 1 KB.

Why it's hard: 1 KB of RAM, no floating point, no compiler — every rotation, projection and line is fixed-point INS8070 machine code hand-placed within a kilobyte, streamed to the scope fast enough to beat the phosphor's fade.

▶ **Try it at the table!** The MC6400 + scope are running right here — watch the cube, torus and sphere tumble, then play the vector shooter with the SA/SB buttons.

03 THE TRICK — MAKING A STEP-DAC DRAW LINES

A digital DAC can only *step* between voltages — so a plain DAC drew only a scatter of **bright dots** (the vertices). The beam snapped between them faster than the phosphor could light, and the connecting lines were simply **invisible**. The fix is not more code — it is one resistor and one capacitor per axis:

```
stream only VERTICES:  E000 ← X  then  E001 ← commit (X,Y on one clock edge)
plain step-DAC      → beam snaps between vertices, faster than the phosphor lights → only DOTS
add a matched RC per axis → each step becomes a ramp → the beam is dragged vertex→vertex → LINES
```

The matched **RC low-pass ("slew-limit")** turns every instantaneous jump into a smooth ramp, so the beam is dragged from vertex to vertex and actually draws the edge; **double-buffering** the DAC (X and Y committed on the same edge) makes that ramp a straight diagonal instead of an L-shaped jog. With the RC in place the dots become **solid, continuous wireframes** — an analog fix for a digital-graphics problem. `(An analog CRT scope is essential — DSOs render X-Y vectors poorly.)`

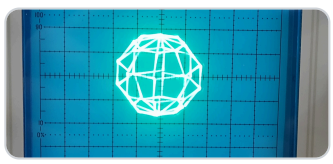
04 THE PROGRAMS

CUBE · TORUS · SPHERE — rotating 3-D wireframes, each drawn as a **single continuous stroke** (a retrace-minimising path for the cube; an **Eulerian circuit** of the mesh for the torus and sphere), so the pen never lifts. Fixed-point rotation uses the INS8070's hardware **16-bit MPY** (and **DIV** for the cube's perspective). About **0.7 KB each**.

★ **VECTOR SHOOTER** — a wholly original, **playable arcade game**: a turret fires at a descending grid of hexagonal aliens; clear the wave for a flashing vector **"WIN"**, get overrun for a big **"X"**, then the next wave drops in. Played from the hex keypad **or the console SA/SB buttons** (both = fire); **F1/F2/F3** show an aliens-remaining bar; selectable **3/6/9 aliens × speed**, with a per-wave **difficulty ramp** — all in 1 KB.


Cube

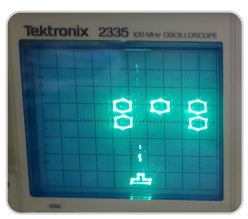

Torus


Sphere

▶ **Watch it live.**
A YouTube demo of the MC6400 drawing solid 3-D vector wireframes on the Tektronix 2335.
youtu.be/bAo9eb0MnI4

05 HOW CLAUDE BUILT IT

- **Hand-wrote raw INS8070 machine code** — the MasterLab is programmed directly in machine language; no compiler, no OS.
- Built a from-scratch Python toolchain: `asm8070.py` (a two-pass **INS8070 assembler**), `ins8070.py` (a **cycle-accurate simulator** with a virtual DAC + keypad, validated against the real monitor ROM), and `render.py` (an **X-Y scope renderer** that models the analog beam → PNG/GIF) — so a program is *seen* before any hardware.
- **Diagnosed "dots, not lines" as an analog problem**, not a software one — deriving the RC slew-limit fix.
- Modeled the INS8070 gotchas that bite: **A is the low byte of EA** (so `LD A` clobbers a 16-bit result), and unsigned compares via the **carry flag** (branches only test A).
- Verified in the simulator, **then on a real MC6400 + Tektronix 2335** — iterating with Michael over photos of the live scope.
- **Learned the machine** from its 1983 German manual (which we translated to English) and used Thorsten Brehm's MasterLab emulator as a reference oracle.



Beyond the demos: the **interactive vector shooter** on the 2335 — turret, a bullet in flight, and a wave of hexagonal aliens, played from the MC6400's SA/SB buttons.

Programs & tooling by Claude (Anthropic, Opus 4.8) · with **Michael Wessel**, on real hardware · code, write-up & build docs: **github.com/lambdamikel/philips-mc6400-vector-graphics** · loaded via **PicoRAM Ultimate** · Philips MasterLab © 1983, work © M. Wessel 2026

```
DAC / SCOPE MAP
E000 ← X (hold)   ·   E001 ← Y (commit both)
E002 ← Z blank    ·   E003 ← frame marker
draw = stream VERTICES; the RC draws the LINES
```