The Coleco ADAM Universal Interface (RS-232 / Parallel) — Hardware, Firmware & Provenance

The Coleco ADAM Universal Interface (RS-232 / Parallel)

Hardware identification, firmware reverse engineering, and host-software confirmation of an unreleased Coleco peripheral — from engineering prototype PCB X00257 REV 0 (EPROM "AHSPI REV 1.2", a near-complete firmware missing only the AdamNet STATUS handlers) to the production "ADAM Universal Interface"

ColecoVision & ADAM Archive & Museum · AdamArchive.org · ColecoVisionADAM.com
Hardware analysis and firmware development by Richard DiRocco with Claude (Anthropic). Prototype EPROM disassembly and original 6801 reverse-engineering by **Chris Braymen** (1995) with annotation by **Richard F. Drushel** (1996); ADAM CP/M 2.2 BIOS disassembly and host-driver discovery by Drushel; SPI schematic digital cleanup by Drushel (2005).
Document version 2.0 · 20 June 2026

Provenance & attribution. The 6801 prototype firmware disassembly used throughout this document — including the symbol set (`NET_XMIT`, `XMIT_CU`, `SEND_REPLY`, `JUST_RETURN`, `STAT_MSG_TBL_E/D/2`, `RAM_STAT_MSG_E`, `DEV2D_STATUS`), the routine boundaries, and the identification of the damaged STATUS region — is "99%+" the work of **Chris Braymen**, c. 1993–1995, using chip-reading equipment at Sierra On-Line; the annotation pass and the original missing-routine reconstruction sketch are by **Richard F. Drushel**, Case Western Reserve University, 1996. The ADAM CP/M 2.2 BIOS disassembly — and the discovery of the prototype host-side serial/parallel driver at device 14 / AUX — are also Drushel's, via his own QuickBASIC Z80 disassembler. The SPI schematic digital cleanup is Drushel, 15 July 2005. The prototype EPROM image used here is circa 1993 and is byte-identical to Braymen's SPI.BIN. The Phase 1 STATUS-handler reconstruction (§11) is Richard DiRocco with Claude (Anthropic), 2026 — a completion consistent with the firmware's confirmed idiom, not a recovery of the original bytes.

CONFIRMEDverified from schematic, photo, disassembled opcode, or running reference code INFERREDstrong inference from evidence OPENrequires further investigation DESIGNnew engineering work introduced here

Contents

1. Executive summary

This document establishes, from multiple independent lines of evidence, that a clear-acrylic-cased Coleco engineering prototype — PCB `X00257 REV 0`, carrying an EPROM hand-labelled `AHSPI REV 1.2` — is the engineering precursor to an unreleased production peripheral Coleco called the **ADAM Universal Interface (RS-232 / Parallel)**. The device attaches to the AdamNet bus and gives the ADAM a standard RS-232 serial port and a Centronics parallel port, selectable through a three-position mode switch, with transparent compatibility for existing software via SmartWriter-printer emulation.

The identification rests on five sources that agree with one another: the prototype hardware and its firmware; a Coleco Marketing Communications "New Products" memo describing the finished product; a working emulation of the same device built by the FujiNet project (whose author worked backward from the ADAM CP/M BIOS); the AdamCalc application, which detects the interface at boot; and Chris Braymen's 1995 disassembly of the prototype EPROM, which is byte-identical to the surviving image we hold and proves the firmware is structurally and functionally complete except for a small clobbered region containing the three device STATUS handlers.

Headline findings. The device lives on **AdamNet device 14 (hex $0E)**; it is software-compatible with SmartBASIC and CP/M because it emulates the built-in SmartWriter printer rather than demanding a new driver; only two prototype units are known to exist; and the production unit was advertised internally but never shipped. The prototype firmware is **~97% complete by byte count**: it implements full bidirectional buffered serial through the SCN2661A USART (initialised from a six-byte parameter block, with a 90-byte receive ring at `$008D`, XON/XOFF and RTS/CTS flow control, and concurrent interrupt-driven AdamNet on the 6801's built-in SCI). The **only** missing piece is the 57–59-byte block that held the device $0D / $02 / $0E STATUS responders, clobbered in the surviving image by a misplaced copy of the vector table. The ADAM CP/M 2.2 retail BIOS contains a working

host-side driver for this device (device 14 / AUX in CP/M terms), discovered by Drushel via his own QuickBASIC Z80 disassembler — meaning the production CP/M shipped with host support for an AdamNet peripheral that was never released.

This document also presents **a functionally-correct reconstruction** of the three missing STATUS handlers (§11), assembled and byte-clean against the original image, that emits the expected six-byte AdamNet STATUS response for each of devices $0E, $0D, and $02 under every modelled RAM state. The reconstruction uses only routines and data already present in the surviving ROM, and is one of two plausible completions; the alternative (a TDRE-interrupt path sketched in Drushel's 1996 annotation) is discussed alongside it in §11.

2. What this device is

The ADAM shipped with a closed peripheral philosophy: printers, drives, and the keyboard all spoke AdamNet, Coleco's proprietary half-duplex serial bus, and there was no standard RS-232 or Centronics port for connecting third-party printers, plotters, modems, or terminals. The Universal Interface was Coleco's answer: a single AdamNet peripheral exposing *both* an industry-standard RS-232 serial port and a Centronics parallel port, so that ADAM owners could attach the same printers and modems everyone else used.

Two engineering goals shaped the design. First, the unit had to present itself on AdamNet as a device the ADAM already understood, so that existing software — SmartBASIC, CP/M, SmartWriter — would drive it without modification. It achieves this by **impersonating the built-in printer** and by intercepting SmartWriter's output, rather than by introducing a brand-new device type that software would have to be taught about. Second, it had to be switchable, so the user could choose whether output went to the serial port, the parallel port, or straight through to the original SmartWriter printer.

3. The production product: ADAM Universal Interface

A Coleco Marketing Communications "New Products" memo documents the intended retail product. It describes a unit that connects RS-232 or Centronics peripherals — printers, plotters, modems, and input devices — to the ADAM through AdamNet, supports up to two units simultaneously, and is software-compatible with SmartBASIC, CP/M, and other software written to use it. The finished product carried a clean white/cream enclosure labelled **ADAM UNIVERSAL INTERFACE** with an ON indicator LED.

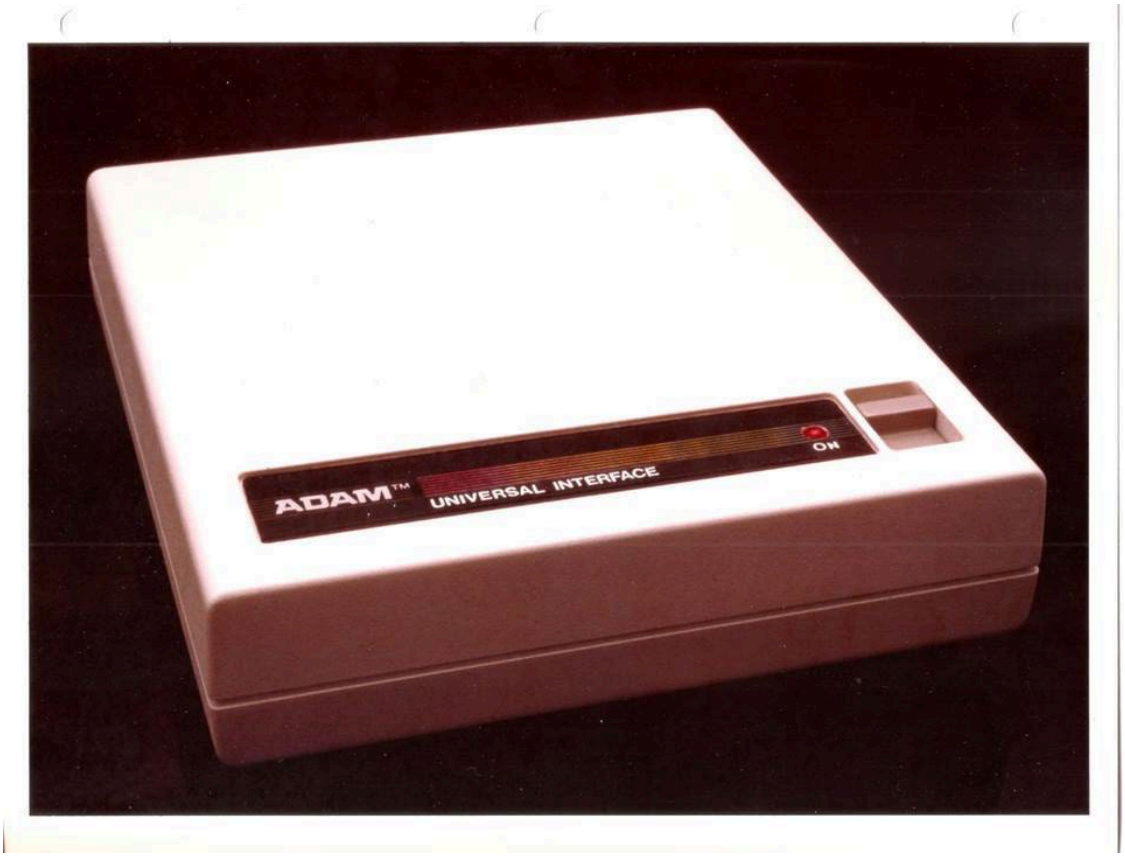

Figure 1. The production ADAM Universal Interface (Serial / Parallel) as pictured in Coleco materials — the finished enclosure that the X00257 prototype was engineering toward. The unit never reached retail.

The codename on the prototype EPROM, `AHSPI`, is consistent with "ADAM Host Serial Parallel Interface" INFERRED — the working engineering name for what marketing would present as the Universal Interface.

4. The engineering prototype: PCB X00257 REV 0

The surviving prototype is a Coleco-manufactured engineering board — plated-through-hole, full silkscreen, solder mask, with the part number `X00257` and revision `REV 0` printed beside the Coleco logo — housed in a hand-cut clear acrylic case. It is not a hand-built breadboard; it is a real pre-production board with a small number of engineering modification wires on the solder side.

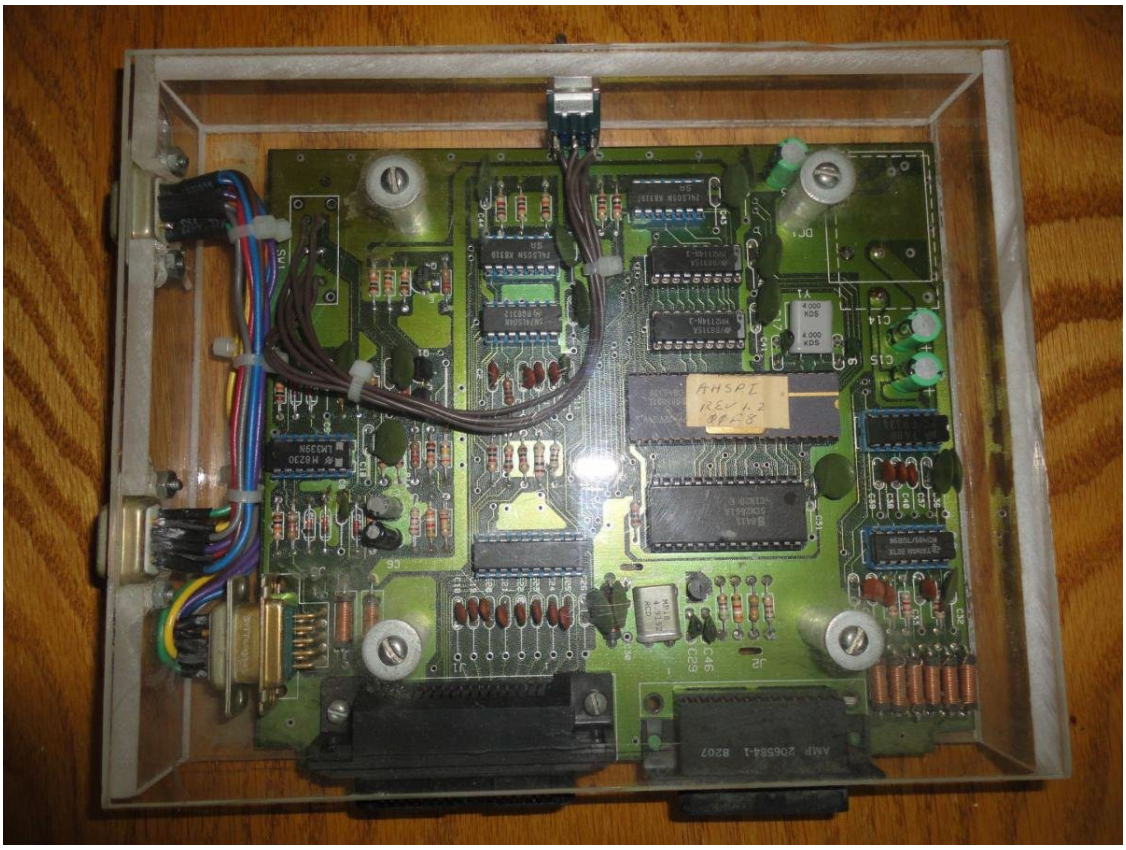

Figure 2. Top-down view through the acrylic lid. The socketed 24-pin EPROM carries an orange sticker reading `AHSPI` / `REV 1.2` / `00F8`. Board silkscreen designators (`U...`, `J1–J3`, `Y1`, `Y2`) are clearly visible.

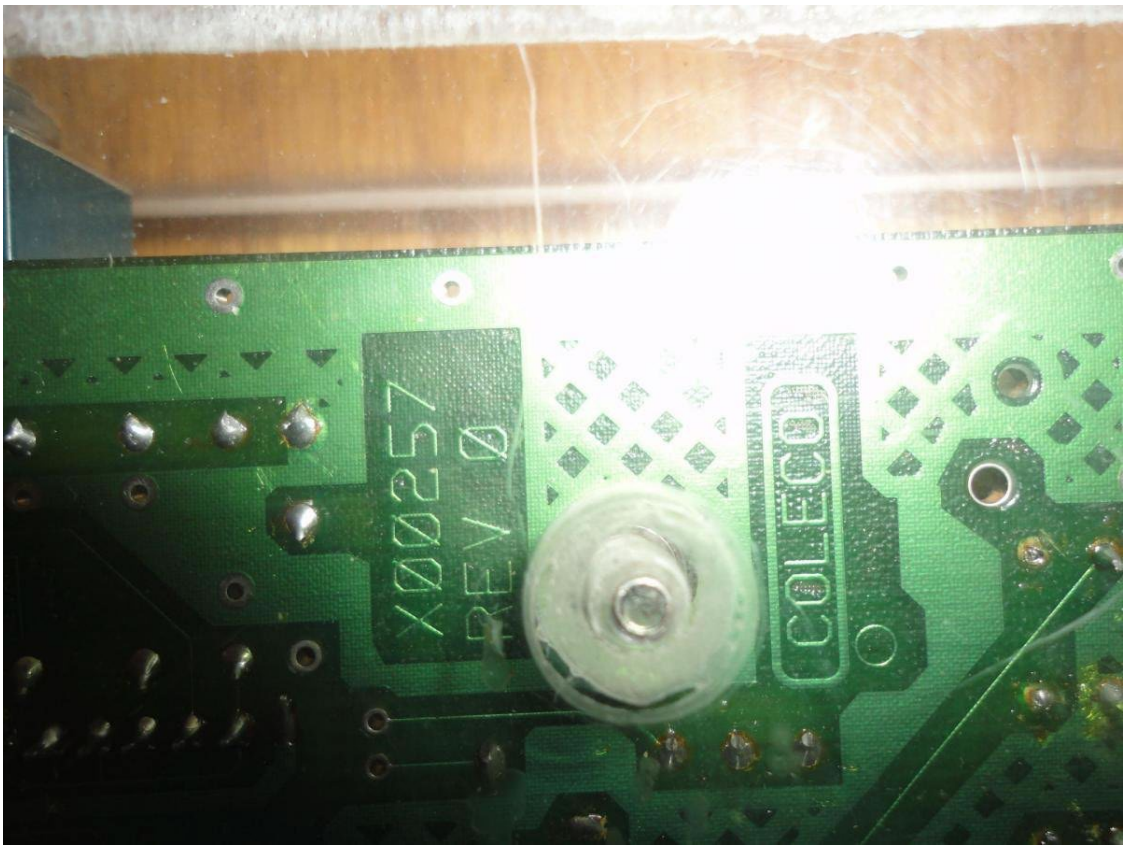

Figure 3. Solder-side silkscreen: part number `X00257 REV 0` with the boxed COLECO logo, positively identifying a Coleco engineering board.

Attribute	Finding	Status
Enclosure	Clear acrylic, screw-fastened, hand-cut panels	CONFIRMED
PCB	`X00257 REV 0`, Coleco-manufactured (silkscreen, solder mask, PTH)	CONFIRMED
EPROM label	Three hand-written lines: `AHSPI` / `REV 1.2` / `00F8`	CONFIRMED
Codename	“ADAM Host Serial Parallel Interface”	INFERRED
Label line 3	`00F8` — firmware entry hint; reset vector targets `$F800`	INFERRED
Units known	Two prototypes known to exist	CONFIRMED

5. Integrated circuits and bus interface

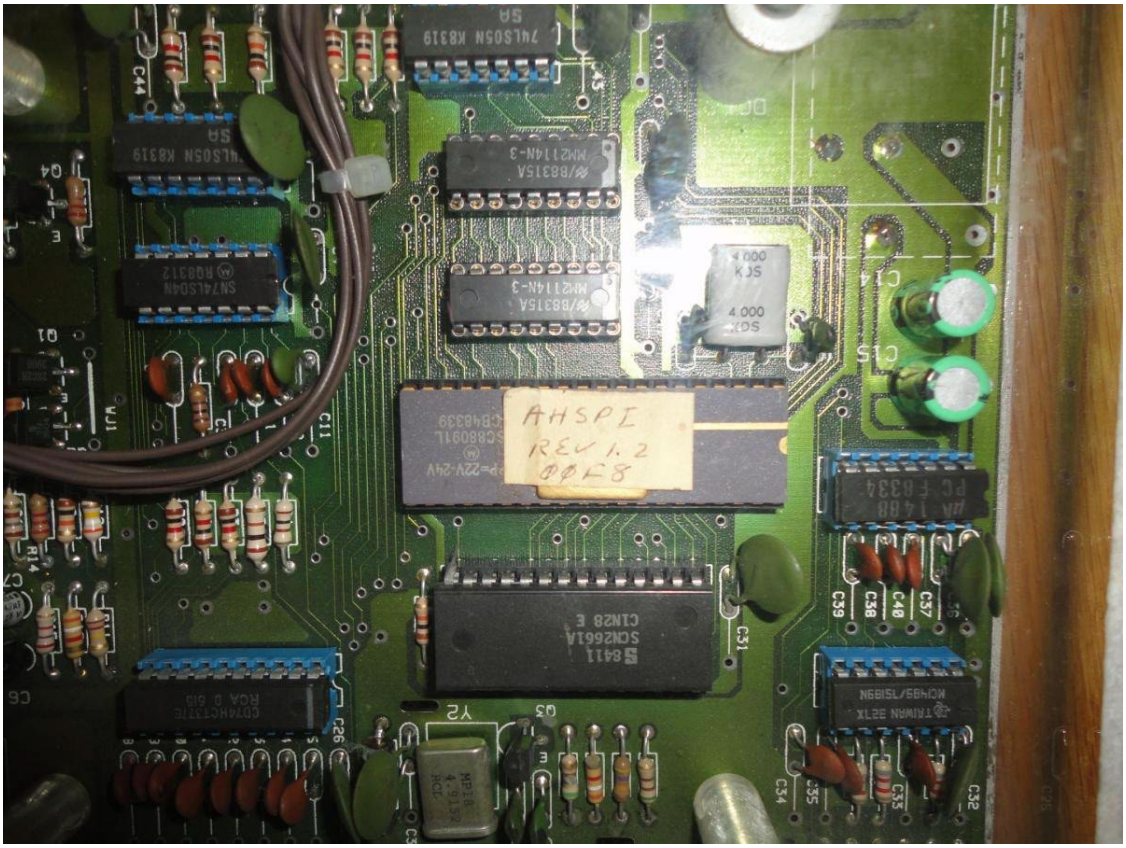

Figure 4. IC close-up. The EPROM sticker, the Signetics SCN2661A USART (date code 8411), two MM2114N-3 SRAMs, 74LS04/74LS05 logic, the 4.000 MHz crystal, and the µA1488 RS-232 driver are all visible.

Function	Device	Role	Status
MCU	Motorola MC6801	System microcontroller; AdamNet node logic and port control	CONFIRMED
Firmware ROM	`SC8060IL` (2716 family, 24-pin)	Mapped at `$F800–$FFFF`	CONFIRMED
USART	Signetics `SCN2661A` (date 8411)	RS-232 serial port engine, mapped at `$0100–$0103` (DATA / STATUS / MODE / COMMAND)	CONFIRMED
SRAM ×2	National `MM2114N-3`	Working RAM (1 KB)	CONFIRMED
Parallel latch	`CD74HC373E` (schematic specifies 74LS273)	Centronics 8-bit data latch	CONFIRMED
Decode logic	`74LS05` open-collector inverters	Wired-AND address decode for UART/RAM/IO selects	CONFIRMED
RS-232	`µA1488`	TTL → RS-232 (±12 V)	CONFIRMED

Function	Device	Role	Status
driver			
RS-232 receiver	1489-equivalent	RS-232 → TTL	CONFIRMED
MCU clock	Crystal Y1 4.000 MHz	6801 clock (E = 1 MHz); AdamNet bit timing	CONFIRMED
USART clock	Crystal Y2 4.9152 MHz	SCN2661A baud-rate generator	CONFIRMED

USART address decode and access. The SCN2661A occupies four byte locations at $0100–$0103 (DATA, STATUS, MODE, COMMAND), with RAM filling $0110–$01FF. Address lines A2 and A3 are *not* decoded, so the four registers also appear at the mirror image addresses $0104/$0108/$010C, $0105/$0109/$010D, etc. — an artefact of the 74LS05 wired-AND decode that is visible in Braymen's disassembly. The firmware accesses the USART extensively: INIT_UART programs mode/baud from a six-byte parameter block whose format matches the CP/M BIOS Init_Aux_Device entry byte for byte; XMIT_UART polls TX_RDY at $0101 bit 0; RCV_UART drains the receiver into a 90-byte ring buffer at $008D–$00E6 with inbound XON/XOFF detection and a 75-character high-water flag. Both XON/XOFF and RTS/CTS flow control are implemented, with state tracked via the SS_* bitmap at $00F2. AdamNet runs interrupt-driven on the 6801's built-in SCI (SPI_MAC at $FBED) while the user UART is polled in the main loop — a clean separation of the time-critical bus from the user-data path. CONFIRMED

6. Connectors and the three-position mode switch

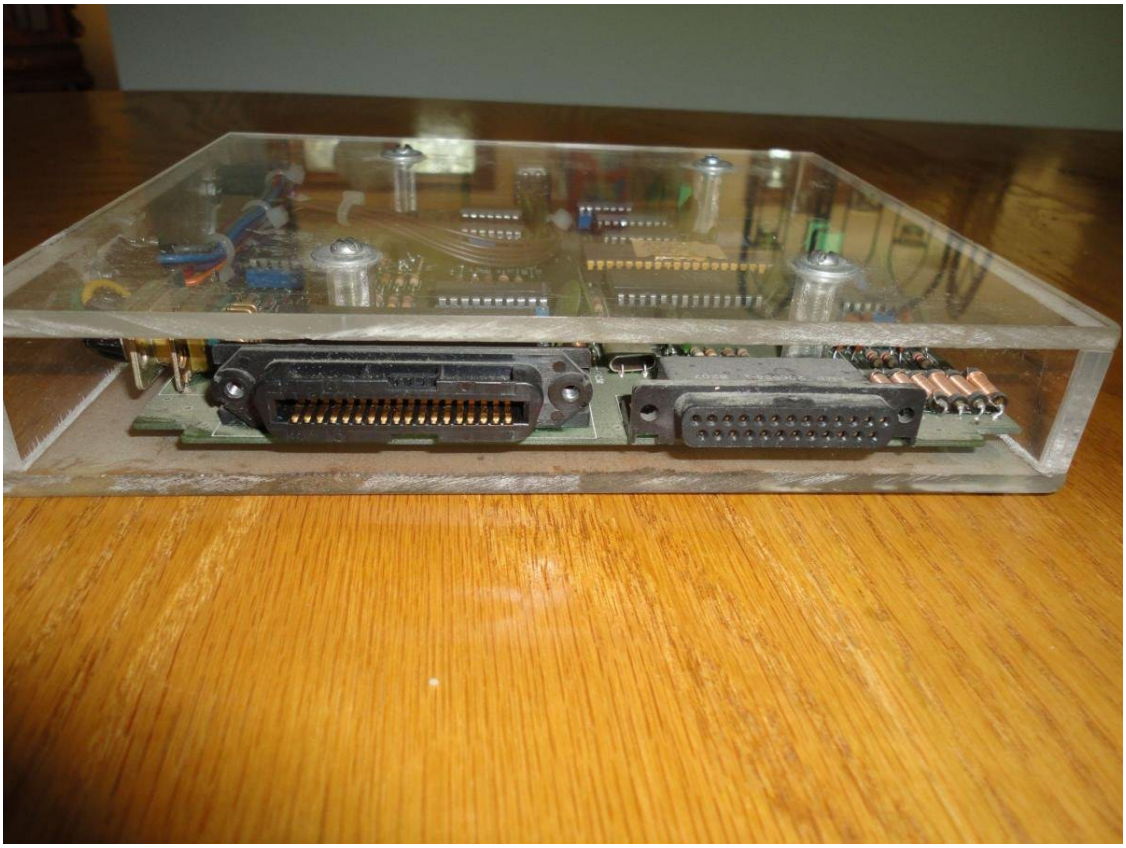

Figure 5. Rear panel: J1 36-pin Centronics (left) for parallel output; J2 25-pin D-shell (right) for RS-232.

Figure 6. Front panel: 9-pin AdamNet + external-power connector (the prototype uses a 9-pin connector, not the usual RJ12).

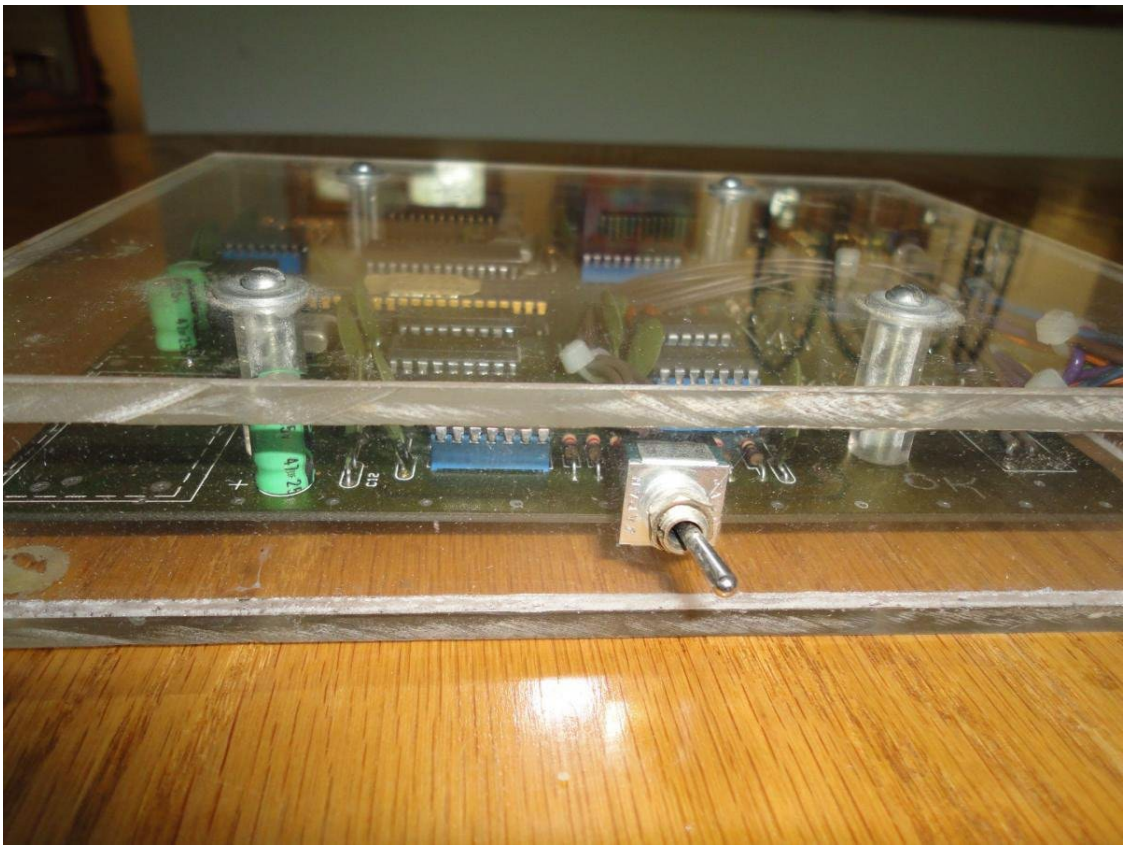

Figure 7. The three-position mode switch on the side panel selects RS-232 / PARL / SmartWriter by driving bits of the 6801's port.

The front 9-pin connector is a notable departure from AdamNet convention. Standard AdamNet peripherals use an RJ12 jack; this device instead uses a 9-pin connector that carries AdamNet signalling *and* external power. The reason is the SmartWriter-intercept function: the switch shunts the SmartWriter printer's input and output so the unit can sit between the ADAM and its printer, selecting whether SmartWriter output is passed through to the original printer or redirected out the new serial/parallel ports.

Note — switch location. A recorded walk-through of the prototype describes the mode switch as being "on the bottom." High-resolution photographs of this unit place the toggle on a *side* panel. The photographic evidence is taken as authoritative here; the discrepancy is most likely a casual description of a different handling orientation.

7. AdamNet device identity: device $0E

On the AdamNet bus, every peripheral answers at a four-bit device address. Roll-call after reset has the ADAM send a CONTROL.STATUS request to each address; a device that responds with a valid status packet is given a Device Control Block and considered present. The Universal Interface answers at **device 14, hexadecimal $0E**.

This is established three independent ways: the prototype firmware dispatches its serial-port STATUS handler from the device-$E vector and carries a device-$E response template in ROM; the FujiNet emulation of the device places it at $0E; and a logic-analyzer capture of a real ADAM boot shows the device answering the status request at address $E. The address-emulation strategy that gives software compatibility means the unit *also* presents printer (device $2) and SmartWriter (device $D) identities for its parallel and pass-through modes.

AdamNet address	Role in this device	Switch position
$0E (14)	RS-232 serial port (SCN2661 USART)	Serial
$0D (13)	Centronics parallel port (74LS273 latch)	Parallel
(no AdamNet device claimed)	SmartWriter printer pass-through	SmartWriter

Device-ID assignment. Application-level evidence from the AdamCalc boot device-probe table at `$32AB` (see § 9) shows that the Universal Interface presents two independent AdamNet device IDs — `$0E` labelled *Serial* and `$0D` labelled *Parallel*. The third switch position is a SmartWriter pass-through that does not claim an AdamNet address; `$02` on AdamNet remains the standard ADAM printer device and is not part of the Universal Interface.

8. Independent confirmation: FujiNet and CP/M

The FujiNet project — an open-source ESP32-based AdamNet peripheral emulator — implemented a working emulation of this exact device and published it in December 2021, describing it as "the unreleased AdamNet Serial Interface for the ColecoVision ADAM… that was prototyped but never released by Coleco." Crucially, the FujiNet author states that the original ROM dumps were corrupt, so the emulation was reconstructed **by working backward from the ADAM CP/M BIOS implementation** of the serial device.

That detail matters in two ways. First, it independently corroborates the prototype's identity, hardware (Signetics 2651 USART, 6801 + EPROM, DB-25 serial, 36-pin Centronics), the device-$0E address, and the "two known to exist" count. Second, it means the ADAM CP/M BIOS itself contains a working host-side driver for this device: by default CP/M maps the TTY and reader/punch devices to the serial port, so a CP/M user can redirect output to the interface with `PIP` or move the console to it for an 80-column serial terminal. The marketing memo's claim of CP/M compatibility is therefore demonstrably real.

Reference implementation. Because CP/M is the authoritative host driver and FujiNet's firmware is open source, the AdamNet node behaviour the production firmware must implement — the exact STATUS response and the SEND / RECEIVE / READY command flow for device $0E — exists as readable, running code. That code is the specification against which the firmware in §11 is written and verified.

9. Host-side detection: AdamCalc

The AdamCalc spreadsheet application (Coleco, 1984) detects the interface during its boot device scan. When the serial interface is present at device $0E, AdamCalc recognises it and draws an on-screen indicator for it — confirmation that period software was already aware of the device, even though AdamCalc does not yet make functional use of it.

AdamCalc's boot image carries a custom TMS9918 graphics set containing device glyphs and text labels. Extracted directly from the application's data, that set includes a keyboard/device icon and legible text fragments including `Serial` and `Parallel` — independent evidence, from inside shipping software, that the serial/parallel interface was a known quantity. The live-emulator capture below shows both labels rendered on the boot screen, alongside the seven default-detected device icons.

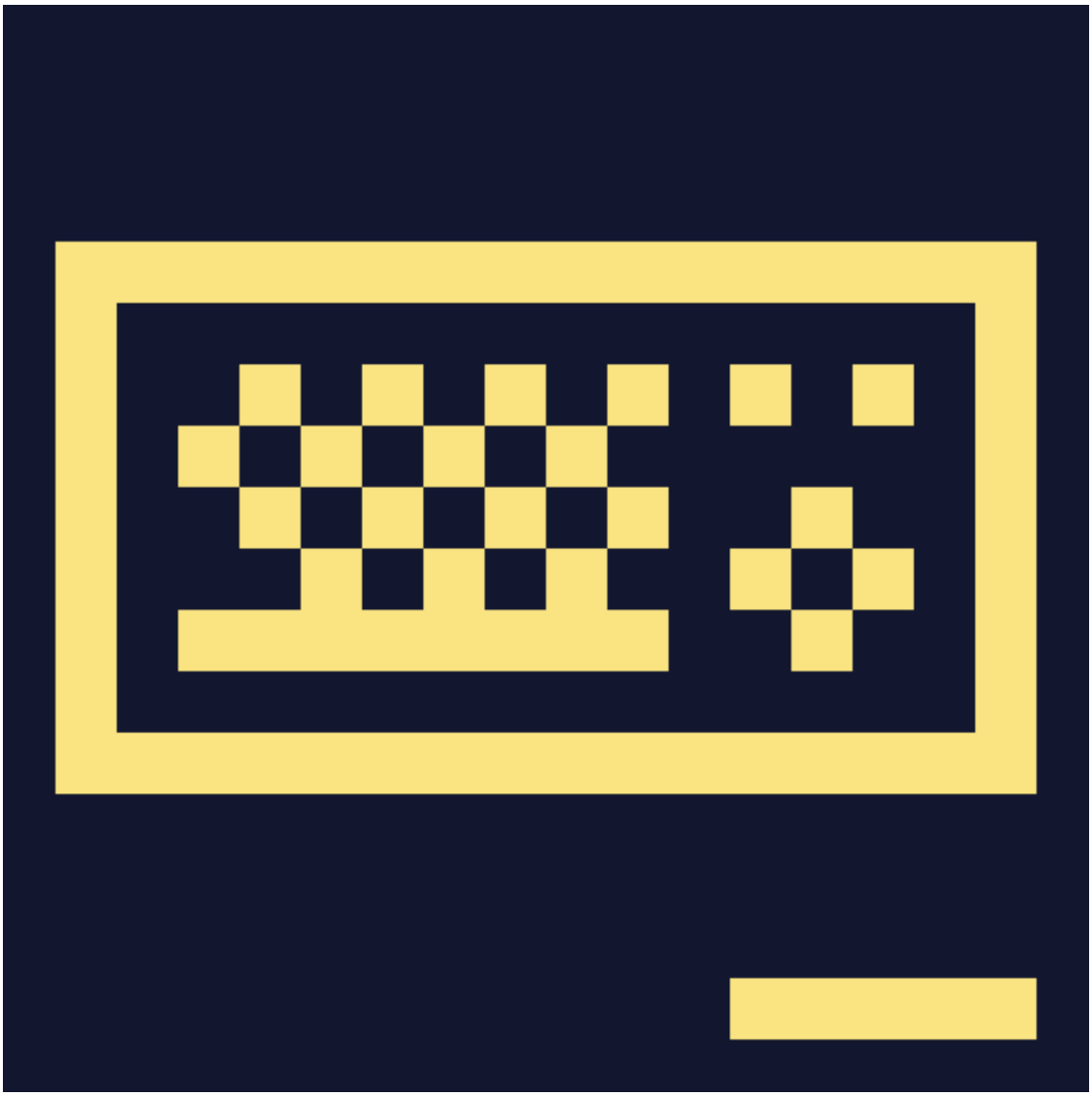

Figure 8. A device glyph reconstructed pixel-for-pixel from AdamCalc's boot graphics data.

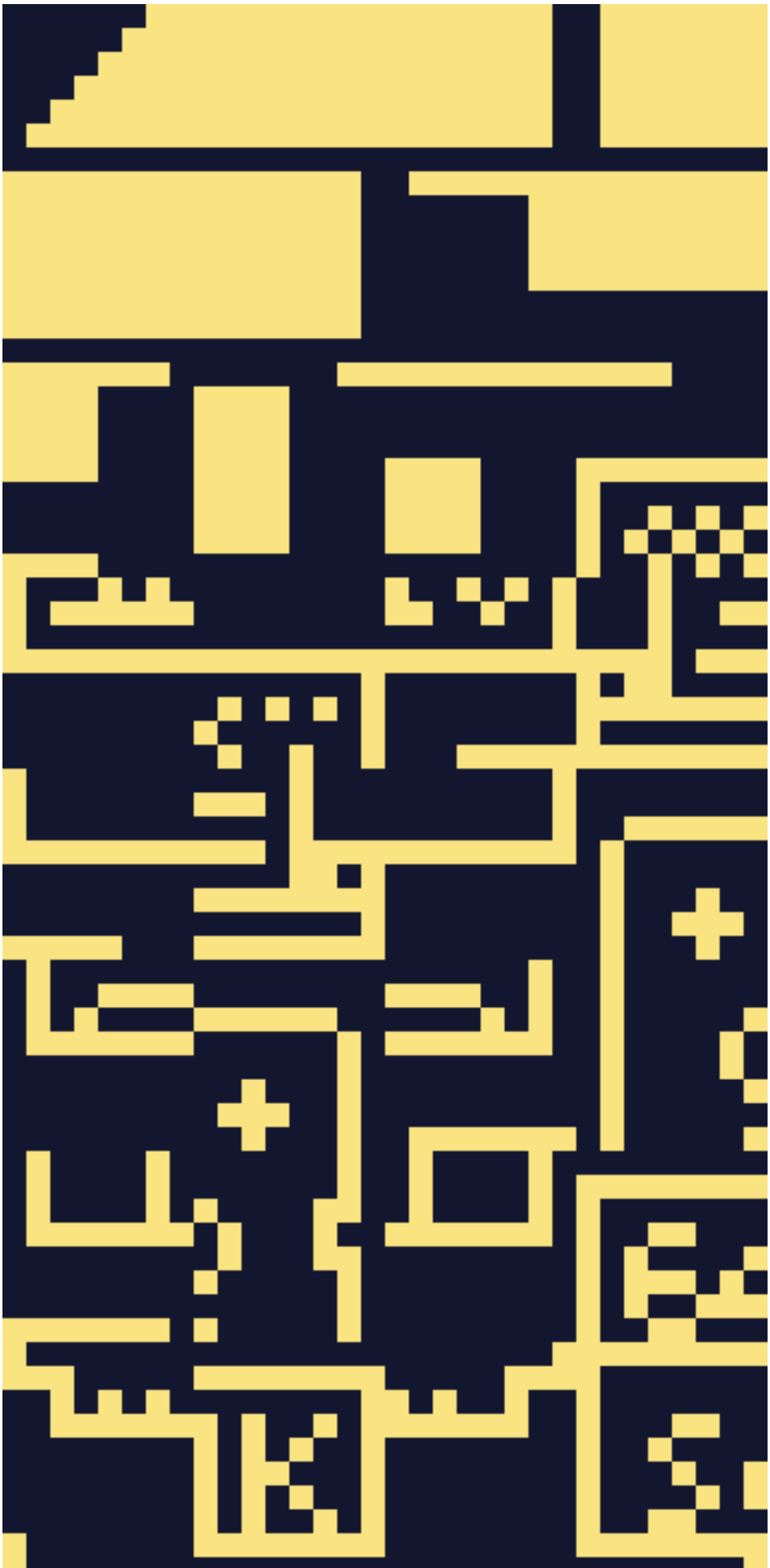

Figure 9. The custom graphic set from AdamCalc, showing device pictograms and text labels including “Serial” and “Paral.”

Figure. AdamCalc boot screen captured from a live emulator session after defeating the per-device AdamNet presence check at `$3256` (a two-byte patch at file offset `0x0A56`: `B7 20` → `37 C9`). All nine entries of the device-probe table at `$32AB` now draw their icons. The bottom row reveals the previously hidden *Serial* and *Parallel* labels.

Icon for device `$0E`. Label *Serial* — the RS-232 half of the Universal Interface.

Icon for device `$0D`. Label *Parallel* — the Centronics half of the Universal Interface. CONFIRMED

The probe table at `$32AB` in the AdamCalc binary is a nine-entry list of `(device-id, icon-code)` pairs. Two of those entries name the Universal Interface explicitly: `$0E` with icon labelled *Serial* and `$0D` with icon labelled *Parallel*. This is independent, application-level corroboration that the device exposes *two* AdamNet identities — one per port — rather than a single identity selected by the front-panel switch. The third (SmartWriter) switch position is a pure pass-through that does not claim an AdamNet address. CONFIRMED

Note on the figures. These are faithful renderings of the raw character data stored in the AdamCalc image; the exact on-screen composition is set at run time by the program's name-table writes and is best captured from a live emulator. The glyph and label *content* shown here is taken directly from the binary.

10. The firmware gap: the missing STATUS handlers

The prototype EPROM is **~97% complete**. Disassembly — Chris Braymen 1995, with annotation by Drushel 1996 — shows a structurally and functionally complete firmware: working AdamNet byte transmitter (`NET_XMIT` at `$FF2C`), command dispatcher, polled bidirectional buffered serial through the SCN2661 with full flow control, reset and interrupt vectors. The only missing code is a 57–59-byte block holding the three device STATUS responders, and that block is not *erased* — it is **clobbered by a misplaced copy of the ROM's own tail vector table**. The exact byte pattern in the surviving image:

```
$FD36-$FD5F  42 bytes of $FF
$FD60-$FD6F  stray DUPLICATE interrupt-vector table:
             FB ED  F8 00  F8 00  F8 00  F8 00  F8 00  F8 00  F8 00
             (byte-identical to the real vector table at $FFF0)
$FD70        one orphan $DF byte
$FD71-$FD73  7E FE 24  =  JMP JUST_RETURN   (surviving tail of the lost
STATUS_E path)
$FD74        SEND_NACK_2 — normal code resumes
```

Two byte-counts have circulated for this region (Drushel's ASM header reads 59 bytes; his cover note says 57). Both are right: the damaged span runs from `$FD36` to somewhere in `$FD6E–$FD70` depending on whether one stops counting at the stray vector block or includes the orphan `$DF`. The dispatcher targets that fall into this region are still valid:

Dispatch	Target	Device	State in prototype
`$FC4F: JMP $FD44`	`$FD44`	$0E (Serial)	inside clobbered region
`$FCB3: JMP $FD3D`	`$FD3D`	$02 (SmartWriter intercept)	inside clobbered region
`$FC80: JMP $FD36`	`$FD36`	$0D (Parallel)	inside clobbered region

The ROM does still contain the static STATUS templates the handlers were meant to read from, sitting in the same neighbourhood and untouched by the clobber:

```
$FF70:  8E 10 00 00 00     ; STAT_MSG_TBL_E  (device $E, max msg 16)
$FF75:  82 10 00 00 00     ; STAT_MSG_TBL_2  (device $2, max msg 16)
$FF7A:  8D 01 00 00 00     ; STAT_MSG_TBL_D  (device $D, max msg  1)
```

The boot path also exists: `INIT_RAM` copies `STAT_MSG_TBL_E` into a six-byte RAM-resident message at `RAM_STAT_MSG_E` (`$00ED`), and the main-loop housekeeping for the serial port (`DO_CLR_E`) already maintains the two dynamic bytes that complete it (the xmit-size hint at `$00F1` and the `SS_*` flag byte at `$00F2`). The shared device-$0D/$02 dynamic status byte at `DEV2D_STATUS` (`$00FE`) is likewise maintained by the existing parallel/SmartWriter state machine. **The clobbered handlers had nothing to invent;** they only had to walk these

tables and send the bytes. This is the concrete shape of the "prototyped but never finished" description the community has carried for years — not abandonment, just one small block lost in an otherwise shipping-ready ROM.

11. Phase 1 firmware: a functionally-correct reconstruction DESIGN

This section presents an assembled, byte-clean completion that fills the three missing STATUS handlers. The original Coleco bytes are unrecoverable from the surviving image, so what follows is a *reconstruction*, not a recovery — one that emits the protocol-correct six-byte response for each device while changing nothing outside the clobbered region.

Design

On a CONTROL.STATUS request, an AdamNet node must reply with six bytes: a four-byte fixed prefix taken from the ROM template (response opcode + max-message length lo/hi + xmit code), followed by two dynamic bytes the device's own state machine maintains. For device $0E the dynamic pair is the current xmit-size hint (`$00F1`) and the `SS_*` flag byte (`$00F2`); for devices $0D and $02 the prefix is followed by a fifth zero byte and the shared `DEV2D_STATUS` byte at `$00FE`. The handlers walk these existing tables and send the bytes via the firmware's confirmed transmit path:

- Three in-gap 3-byte dispatchers in the clobbered region at `$FD36` / `$FD3D` / `$FD44` — each a single `JMP` to its corresponding sender.
- Three sender routines in the formerly-unused tail of ROM at `$FF7F+`. Each loops over its source bytes via `NET_XMIT` (the confirmed AdamNet byte transmitter at `$FF2C`), then chains into the firmware's existing cleanup tail — either `SEND_REPLY` (which itself ends in `JSR XMIT_CU / JMP JUST_RETURN`) or that same tail inline.
- No new infrastructure, no new RAM variables. Every routine and table referenced is already present in the surviving ROM, in `INIT_RAM`'s output, or in the main loop's housekeeping.

The three senders (assembled at $FF7F+)

```
SEND_STAT_D    ; $FF7F  - device $0D (Parallel)
        LDAB    #5                  ; five template bytes
        LDX     #STAT_MSG_TBL_D     ; $FF7A
SSD_LOOP:
        LDAA    0,X
        JSR     NET_XMIT
        INX
        DECB
        BNE     SSD_LOOP
        LDAA    DEV2D_STATUS        ; $00FE - sixth (dynamic) byte
        JMP     SEND_REPLY          ; tail: XMIT byte + XMIT_CU + JUST_RETURN

SEND_STAT_2    ; $FF92  - device $02 (SmartWriter intercept)
        LDAB    #5
        LDX     #STAT_MSG_TBL_2     ; $FF75
```

```
SS2_LOOP:
        LDAA    0,X
        JSR     NET_XMIT
        INX
        DECB
        BNE     SS2_LOOP
        LDAA    DEV2D_STATUS        ; $00FE
        JMP     SEND_REPLY

SEND_STAT_E    ; $FFA5  - device $0E (Serial)
        LDAB    #6                  ; six bytes from RAM (prefix already
merged)
        LDX     #RAM_STAT_MSG_E     ; $00ED
SSE_LOOP:
        LDAA    0,X
        JSR     NET_XMIT
        INX
        DECB
        BNE     SSE_LOOP
        JSR     XMIT_CU             ; final-byte cleanup
        JMP     JUST_RETURN         ; re-enable RX int, RTI
```

In-gap dispatchers

```
$FD36 (STATUS_D):  JMP SEND_STAT_D    ; $7E FF 7F, then $FF padding
$FD3D (STATUS_2):  JMP SEND_STAT_2    ; $7E FF 92, then $FF padding
$FD44 (STATUS_E):  JMP SEND_STAT_E    ; $7E FF A5, then $FF padding
```

58 bytes change in 2048; every changed byte was `$FF` padding in the surviving image. Interrupt vectors at `$FFF0–$FFFF` and the three status-template tables at `$FF70–$FF7E` are byte-identical to the original.

OPEN An alternative completion: the TDRE-interrupt path

Drushel's 1996 annotation pass sketched an alternative reconstruction that drives the STATUS reply through the 6801's SCI transmit-data-register-empty interrupt: arm `SCI_TR_CS=$0F`, set an `int_flag` to `tdre_int`, send byte 1, return, and let the TDRE ISR walk a `data_index/COUNT` through the remaining five bytes. Both designs emit the same six bytes; they disagree on the mechanism.

	Drushel 1996 sketch	Phase 1 (this document)
Form	DESIGN Commented pseudocode in `SPI-RFD.ASM`; never assembled	DESIGN Assembled, verified, byte-clean
Mechanism	TDRE-interrupt driven, returns to main loop after byte 1; ISR streams the rest	Blocking via `NET_XMIT`, matching the confirmed `LETS_SEND_DATA` path already used by `CLR_E`
Infrastructure needed	A TDRE branch in `SPI_MAC`, plus RAM vars `int_flag`, `data_index`, and a	None — every routine called (`NET_XMIT`, `XMIT_CU`,

	Drushel 1996 sketch	Phase 1 (this document)
	`statout` state — *none of which exist in the ROM*; symbols appear only in the comments	`SEND_REPLY`, `JUST_RETURN`) is confirmed present
Implied loss	If original, >59 bytes were clobbered (the new RAM/branch infrastructure has to fit somewhere too)	Consistent with the 57–59-byte damaged region as measured

Parsimony argument. The symbols in the Drushel sketch (`int_flag`, `data_index`, `tdre_int`, `stat_pkt_size`, `statout`) have zero real EQU definitions in the disassembly — they appear only in comments. `SPI_MAC` contains no TDRE-continuation branch. For the TDRE design to have been the original, more than 59 bytes would have to be missing. The blocking version above needs nothing that isn't already there, and matches the way the rest of the shipping firmware sends multi-byte packets. We present it here as the more defensible *completion*; we do not claim it is "the" original. This question is one of the open items in §14.

12. Verification

A data-flow simulator walks the patched binary through the SCI receive ISR's STATUS dispatch for each device under a representative set of RAM states — capturing the exact byte stream the firmware will emit on the AdamNet line — and compares against the expected six-byte response. All six modelled scenarios pass:

Device	Scenario	Bytes transmitted	Tail path	Result
$0E (Serial)	idle, RX ready (`SS_RX_RDY`)	`8E 10 00 00 10 08`	XMIT_CU → JUST_RETURN	PASS
$0E (Serial)	15 bytes buffered, high water (`SS_HI_WATER`)	`8E 10 00 00 10 04`	XMIT_CU → JUST_RETURN	PASS
$0E (Serial)	XOFF received from remote (`SS_GOT_XOFF`)	`8E 10 00 00 10 40`	XMIT_CU → JUST_RETURN	PASS
$0D (Parallel)	ready	`8D 01 00 00 00 01`	SEND_REPLY → XMIT_CU → JUST_RETURN	PASS
$0D (Parallel)	not ready	`8D 01 00 00 00 00`	SEND_REPLY → XMIT_CU → JUST_RETURN	PASS
$02 (SmartWriter intercept)	ready	`82 10 00 00 00 01`	SEND_REPLY → XMIT_CU → JUST_RETURN	PASS

Dispatcher verification: `$FD36 → $FF7F`, `$FD3D → $FF92`, `$FD44 → $FFA5`, all confirmed targets. Status templates at `$FF70–$FF7E` and interrupt vectors at `$FFF0–$FFFF` are byte-identical to the original image.

Scope of "verified." This is data-flow correctness only: the patched firmware reads the right bytes from the right places, sends them through the confirmed transmit path, and returns cleanly to the firmware's normal continuation point. OPEN Two further verification passes are still required: (i) cycle-accurate 6801 instruction-level simulation against the new layout, and (ii) a real-hardware scope check of the AdamNet bit-cell timing on a physical prototype unit, to confirm the reconstructed handlers meet the bus's response deadline at full bit rate. Until those two pass, the reconstruction is functionally correct but not silicon-proven.

13. Performance: the architectural answer

The FujiNet author noted that the emulated serial path is slow — output emerges one character at a time with noticeable overhead — and invited the community to make it faster. The bottleneck is structural: FujiNet emulates the entire AdamNet node on an ESP32 and bridges the data through a TCP socket, so every byte crosses a general-purpose network stack.

The Coleco prototype answers the same question architecturally. Braymen's disassembly shows a deliberate separation of concerns: AdamNet runs *interrupt-driven* on the 6801's built-in SCI (`SPI_MAC` at `$FBED`), where bit-cell timing is critical and missed deadlines mean a dropped node; the user-facing SCN2661A USART is *polled* in the main loop, where bursts of latency simply absorb into the 90-byte receive ring at `$008D` and the XON/XOFF / RTS/CTS flow control state at `$00F2`. This is the right architecture for the chip and the bus, not a deficiency: it gives AdamNet first claim on CPU cycles while letting the user UART run at its own pace. That is precisely the property a faithful native-silicon implementation preserves and a ported software-emulation cannot.

14. Open investigations and next steps

Item	Status
Cycle-accurate 6801 instruction-level simulation of the Phase 1 patched binary	OPEN
Real-hardware bit-cell timing of the STATUS handlers (scope check on a physical unit)	OPEN
Resolve the STATUS architecture against Drushel's 1996 TDRE-interrupt sketch — specifically, whether he recalls a TDRE-driven transmit path in any 6801 ADAMnet device; affirmative recall would imply >59 bytes were lost	OPEN
Reach Chris Braymen to confirm the historical credit chain firsthand; last known sighting at a single-digit ADAMcon	OPEN
OCR the original Coleco source listings for the Master 6801, ADAM printer, and tape drive from the ADAM Technical Manual; verify each against Braymen's	OPEN

Item	Status
disassembly of the corresponding ROM image; relocate Drushel's `DAISY4.LST` (his reconstruction of the missing first page of the printer listing) from the `KEYBOARD/` to the `PRINTER/` subdirectory of the 6801 archive	
SST 256K solid-state tape drive (ADAMcon 5 demo unit; 32×8K SRAM, battery-backed, four stacked wire-wrapped boards): contact Bob Slopsema for the schematic, never publicly released, and any additional history	OPEN
Function of the second front-panel 9-pin connector on the built unit (schematic shows only one)	OPEN
Phase 2: complete SCN2661A serial driver and SEND / RECEIVE / READY command flow for device $0E, using the CP/M BIOS host driver (Drushel) and FujiNet source as the behavioural spec	DESIGN
Optional: merge Braymen's 1995 source plus the Phase 1 patch into a single reassemblable `SPI.ASM` that produces a complete 2 KB image	DESIGN

15. Sources

Source	Contribution
Prototype PCB X00257 REV 0 and EPROM "AHSPI REV 1.2"	Hardware, IC complement, connector set
Chris Braymen, 6801 prototype-SPI disassembly, c. 1993–1995 (`SPI.ASM` + `SPI.BIN/HEX/S19`)	Authoritative disassembly, symbol set, table layout, identification of the damaged region; byte-identical EPROM image. Provided to the project by Drushel, June 2026, with blanket permission to use.
Richard F. Drushel, annotation pass on Braymen's disassembly, 1996 (`SPI-RFD.ASM`)	Routine-by-routine commentary; original missing-routine reconstruction sketch (the TDRE-interrupt alternative discussed in §11)
Drushel, ADAM CP/M 2.2 BIOS disassembly (`CPM.ASM` + `DEV14.ASM`)	Discovery and working-out of the prototype-only serial/parallel host driver at device 14 / AUX, via Drushel's own QuickBASIC Z80 disassembler — the host-side counterpart that confirms the device shipped supported in retail CP/M
Drushel, SPI schematic digital cleanup, 15 July 2005	Hand-traced clean schematic of the prototype, made from a contemporary photocopy
Coleco Marketing Communications "New Products" memo	Production product identity, feature set, software-compatibility claims
FujiNet project (Thomas Cherryhomes) — videos and open-source firmware	Device $0E confirmation, "two known to exist," CP/M-BIOS-derived behaviour, reference STATUS response and command flow
AdamNet Protocol Reference (Cherryhomes + Coleco	NM_STATUS packet format, roll-call sequence, timing deadlines

Source	Contribution
EPS #227)	
AdamCalc (Coleco, 1984)	Period host-side detection of device $0E; extracted device glyphs and “Serial/Parallel” labels
ADAM CP/M 2.2 (AJM/Coleco)	Retail-shipping host driver (Drushel, see above); TTY / reader / punch mapped to the serial port

Revision history

Version	Date	Notes
1.0	earlier 2026	Initial publication
2.0	20 June 2026	Major revision following receipt of the Braymen/Drushel 1992–2005 archive. Corrected the “USART never accessed” framing (the USART is accessed extensively; §5). Recast the firmware gap as a 57–59-byte clobbered region anatomy with vector-table-duplicate overlay (§10), not an erased abandonment. Replaced the Phase 1 reconstruction with Braymen-symbol-set design using `NET_XMIT` / `SEND_REPLY` / `RAM_STAT_MSG_E` / `DEV2D_STATUS` infrastructure already present in ROM, and added the Drushel-1996 TDRE alternative for comparison (§11). Updated verification to 6/6 data-flow scenarios across three RAM states for device $0E (§12). Reframed performance as architectural answer rather than completion goal (§13). Added provenance & attribution block; added Braymen, Drushel BIOS, Drushel schematic, and Drushel CP/M-host-driver sources (§15).

Prepared for the ColecoVision & ADAM Archive & Museum. Phase 1 firmware artifacts: `SPI_phase1.bin` (patched 2 KB image), `SPI_phase1.asm` (assembly source, drop-in addition to `SPI-RFD.ASM`), `SPI_phase1_diff.txt` (byte-level diff against original), `SPI_phase1_verify.py` (data-flow verifier), `SPI_phase1_verify_report.txt` (6/6 PASS report).