

PAUL HAGSTROM, KANSASFEST 2021

FUNNY MODE

ON THE APPLE ///

IMPROVEMENTS: APPLE II + PLUS

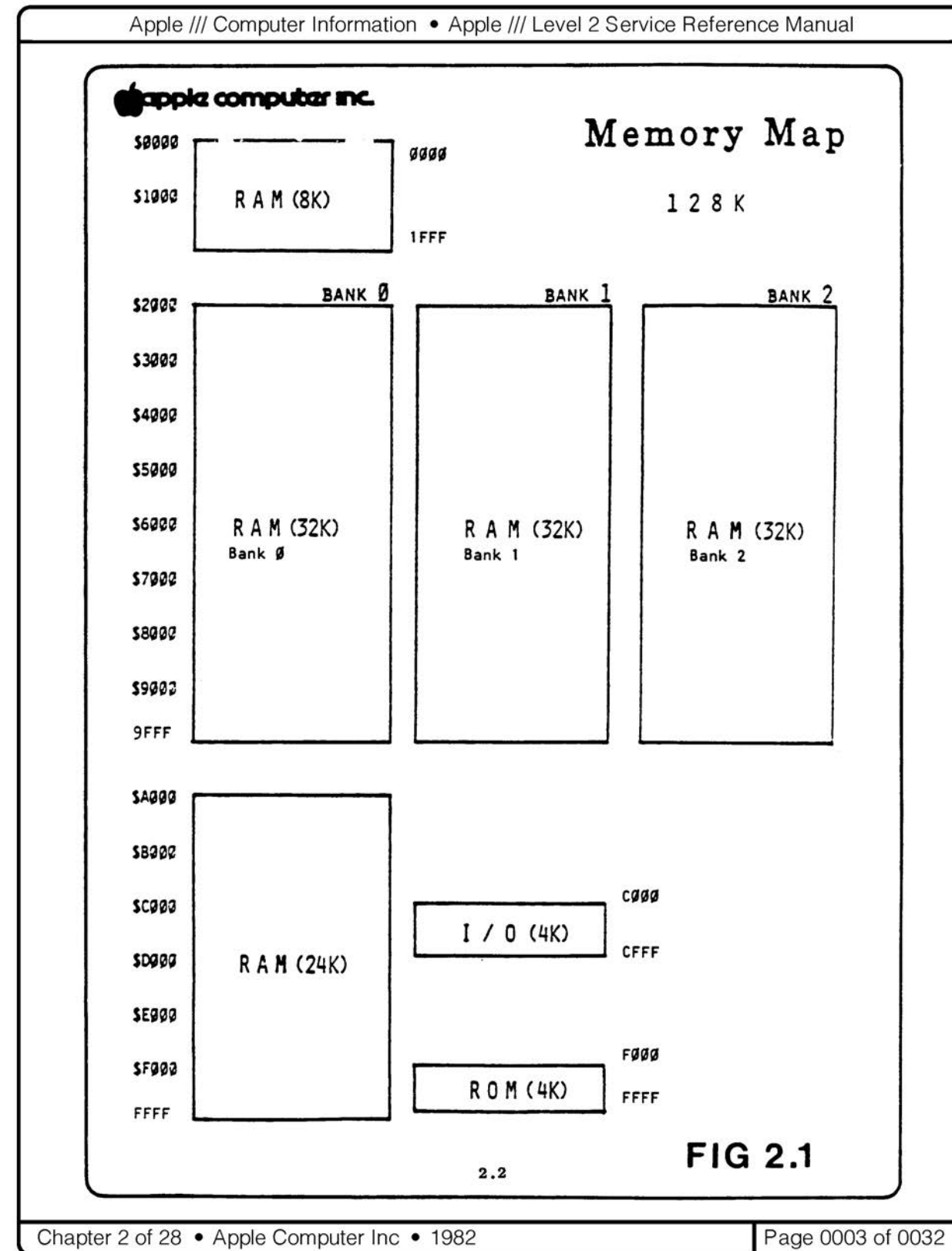
- 128K-256K RAM
- 1.4MHz to 1.8MHz CPU
- Serial port, Silentype printer port
- RGB output, 6 bit audio
- Clock nearly built in
- Chainable disk drives, one built in.
- Better keyboard, numeric keypad
- 80 columns, lowercase, new graphics modes
- Emulation mode for Apple IIs circa 1980. 48K Apple II Plus.

EMULATION MODE!

- I can buy an Apple /// and still have all that Apple II software available!
- Maybe I could write cool Applesoft programs that used the clock, 256K of memory.
- Except, no. The main features of the Apple /// are not available in emulation mode. You have 48K.
- Apple is mean.

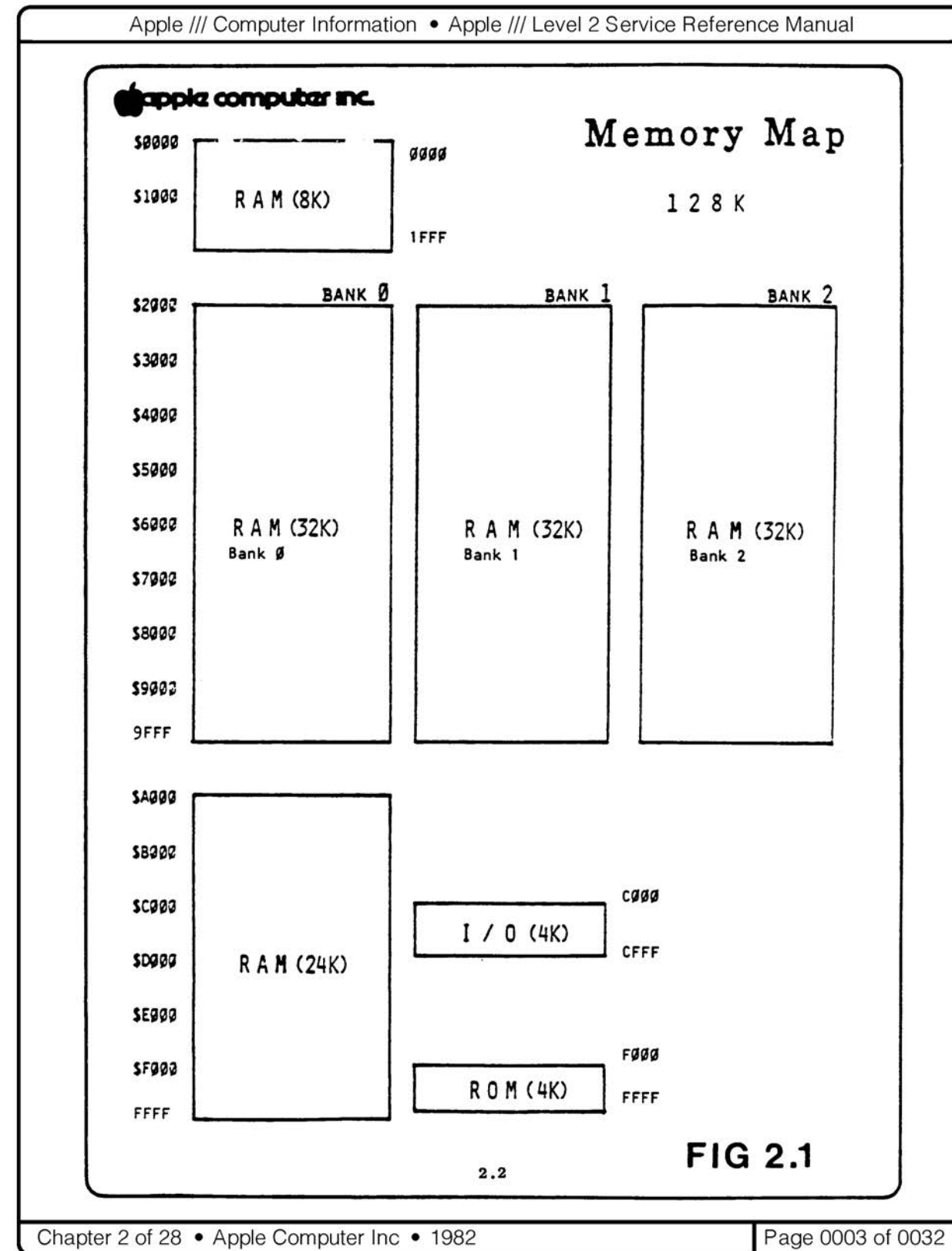
128-256K

- 6502 can see 64K at once.
- But we want more.
- Bank switching: 6502 refers to something within 64K, but Apple /// positions that 64K window over a larger RAM area.



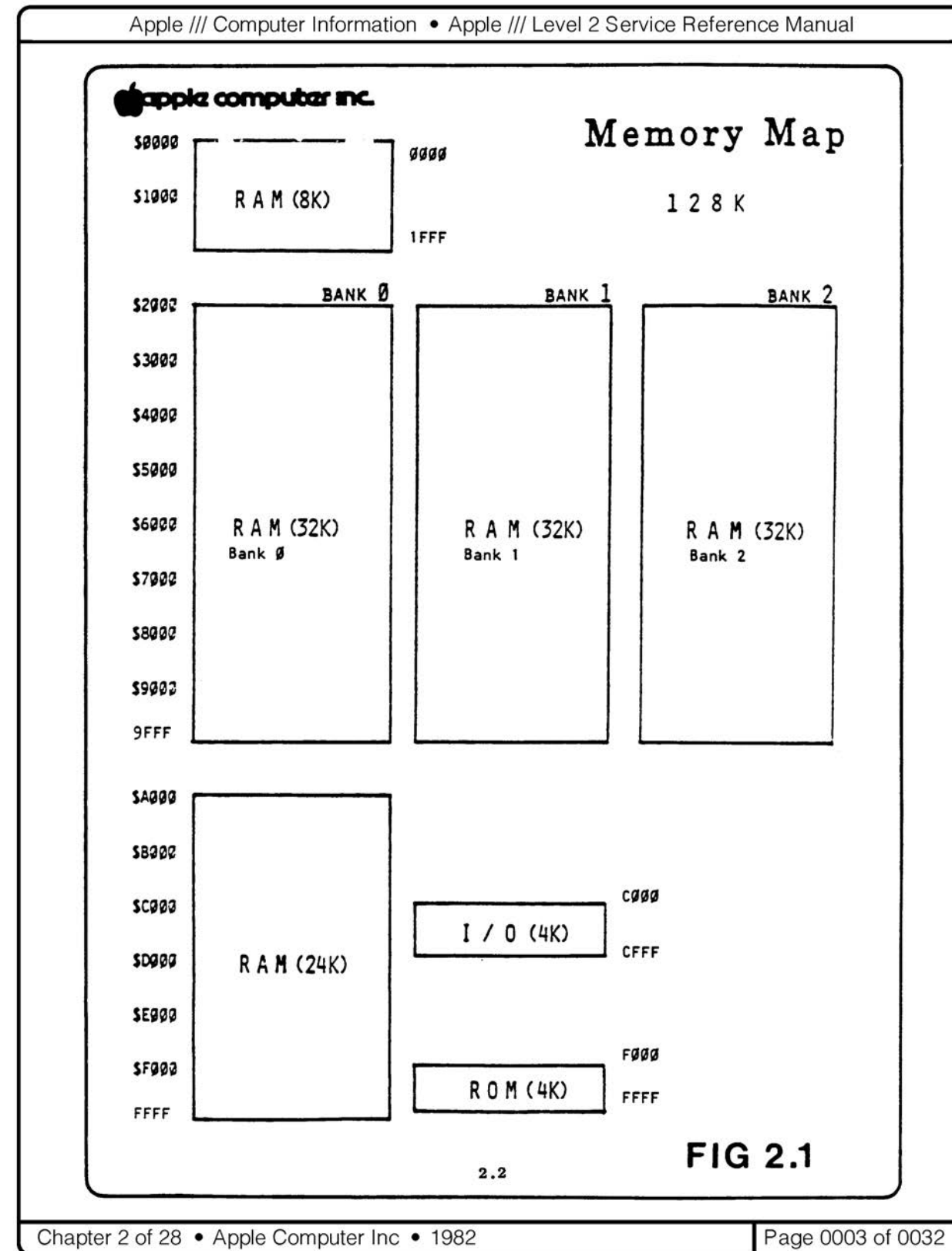
CONTROL VIA REGISTERS

- Versatile Interface Adapter. 6522. I/O device. Controls machine behavior. Similar to (same as?) softswitches.
- **FFEF**: bank register. F0=bank 0, F1=bank 1...
- **FFDF**: environment register. CPU speed, \$Cx I/O exposed, \$Fx ROM/RAM, upper write protect, ...
- This is how you control the fancy things that Apple /// does.



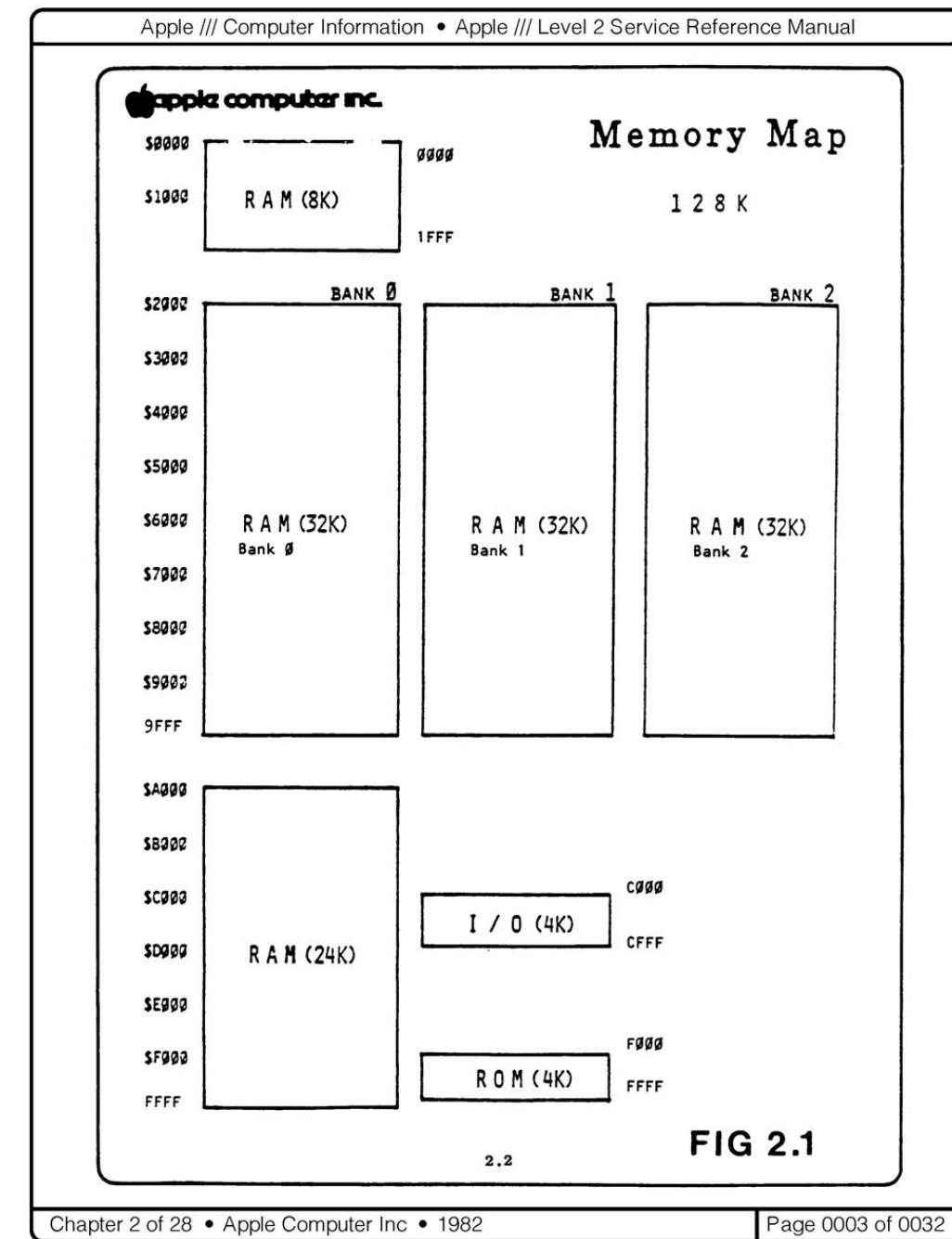
THE LOWER 48

- Compared to the Apple II, the D000-FFFF space is (almost) just RAM (if you hide the A3 ROM). But you could put the A2 ROM in there and lock it.
- Many of the I/O switches are the same, in Cxxx.
- The lower 48K can pretty much act like an Apple II, with a little bit of translation of I/O switches enabled in hardware, and ROM loaded into D000-FFFF. That's Apple II Emulation mode.



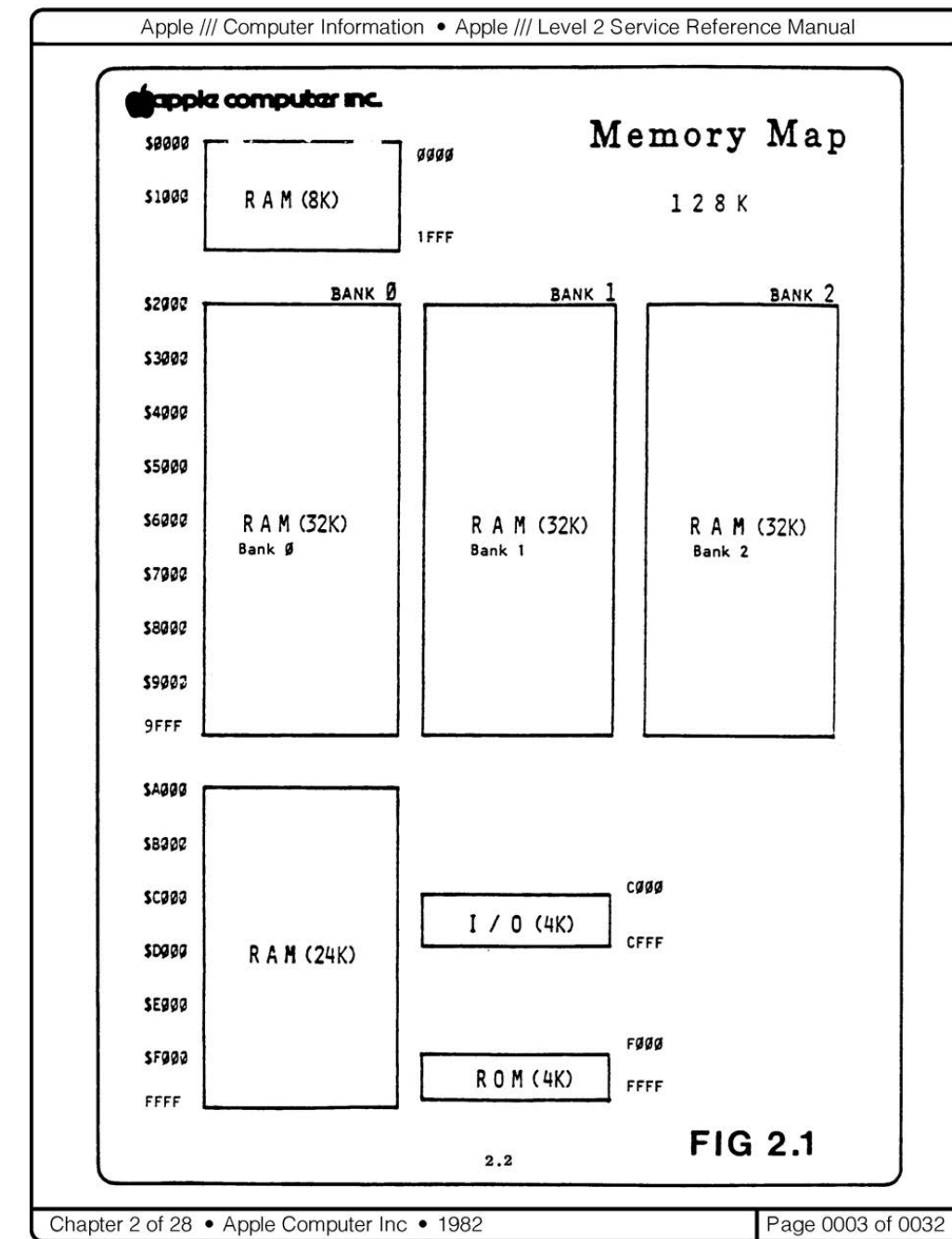
MORE CHIPS TO MAKE IT WORSE

- Why can't it run 64K software?
- Memory banking in the LC is different
 - D0-DF RAM 1, RAM 2, ROM
 - E0-FF RAM, ROM
- High parts of the FF page are effectively soft switches on the /// and must be dodged.
- Cxxx soft switches are not identical (though mostly the same).



MORE CHIPS TO MAKE IT WORSE

- Probably would not have been impossible, but would not have been trivial either.
- Seems not just the addition of a spiteful hardware lock, even if it makes a more dramatic story.
- Apple II emulation mode protects the ROM area, translates the I/O switches where needed, fills in missing graphics modes (text, mixed GR/text).



APPLE][EMULATION DISK (1.1)

- To get to emulation mode, you boot a pre-boot disk.
- Provides options for the emulated machine.
- Put in new disk, boot.
- The bargain: you are protected from injuring yourself on the Apple ///, but you can't reach the Apple ///'s features.
- Largely because VIAs are inaccessible.

A screenshot of a computer window titled "Apple /// [apple3] - MAME 0.233 (LP64)". The window displays the "apple II EMULATION MODE" menu. The menu lists several options with their current settings in boxes: LANGUAGE: APPLESOFT, INTEGER BASIC; CARD: SERIAL, COMMUNICATIONS; BAUD RATE: 110, 300, 600, 1200, 2400, 4800, 9600, 19200; LINE FEED: ENABLED, DISABLED; LINE WIDTH: 40, 72, 80, 132, 255, CHARACTERS; CARRIAGE RETURN DELAY: ON, OFF. At the bottom, there is a list of control keys and their functions: RETURN for BOOT apple II DISK, ESCAPE for RESTORE DEFAULTS, [F1] & RETURN for SAVE CONFIGURATION TO EMULATION DISK, and arrow keys for SELECTION KEYS.

```
Apple /// [apple3] - MAME 0.233 (LP64)
apple II  EMULATION MODE

⇒ LANGUAGE:  APPLESOFT  INTEGER BASIC
CARD:       SERIAL    COMMUNICATIONS
BAUD RATE:   110    300    600    1200
             2400   4800   9600   19200
LINE FEED:   ENABLED   DISABLED
LINE WIDTH:  40    72    80    132  255 CHARACTERS
CARRIAGE RETURN DELAY:  ON  OFF

[RETURN]      - BOOT apple II DISK
[ESCAPE]      - RESTORE DEFAULTS
[F1] & [RETURN] - SAVE CONFIGURATION
                  TO EMULATION DISK
[↑],[↓],[←],[→] - SELECTION KEYS
```

EMULATION DISK ORGANIZATION

When you hit return, the appropriate segments are loaded into high memory, the machine control registers are set for Emulation mode, memory above \$C000 is write protected, and control transfers to the Apple II auto-start routine.

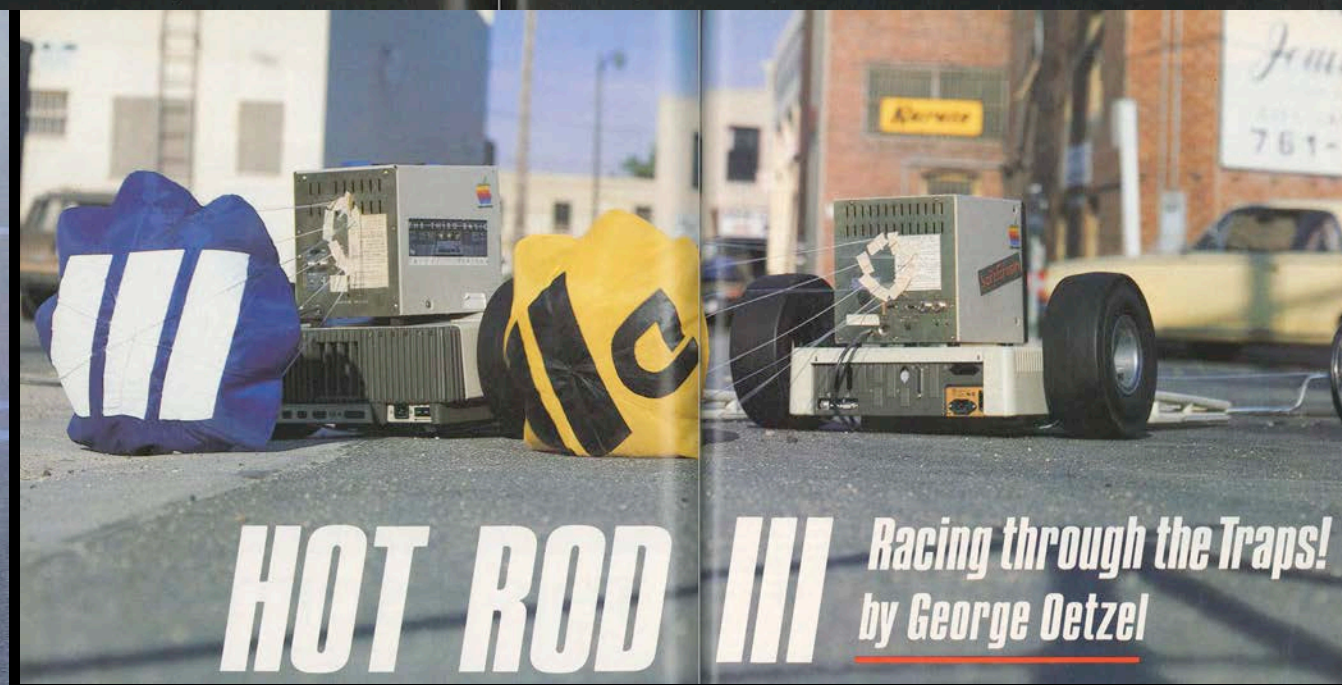
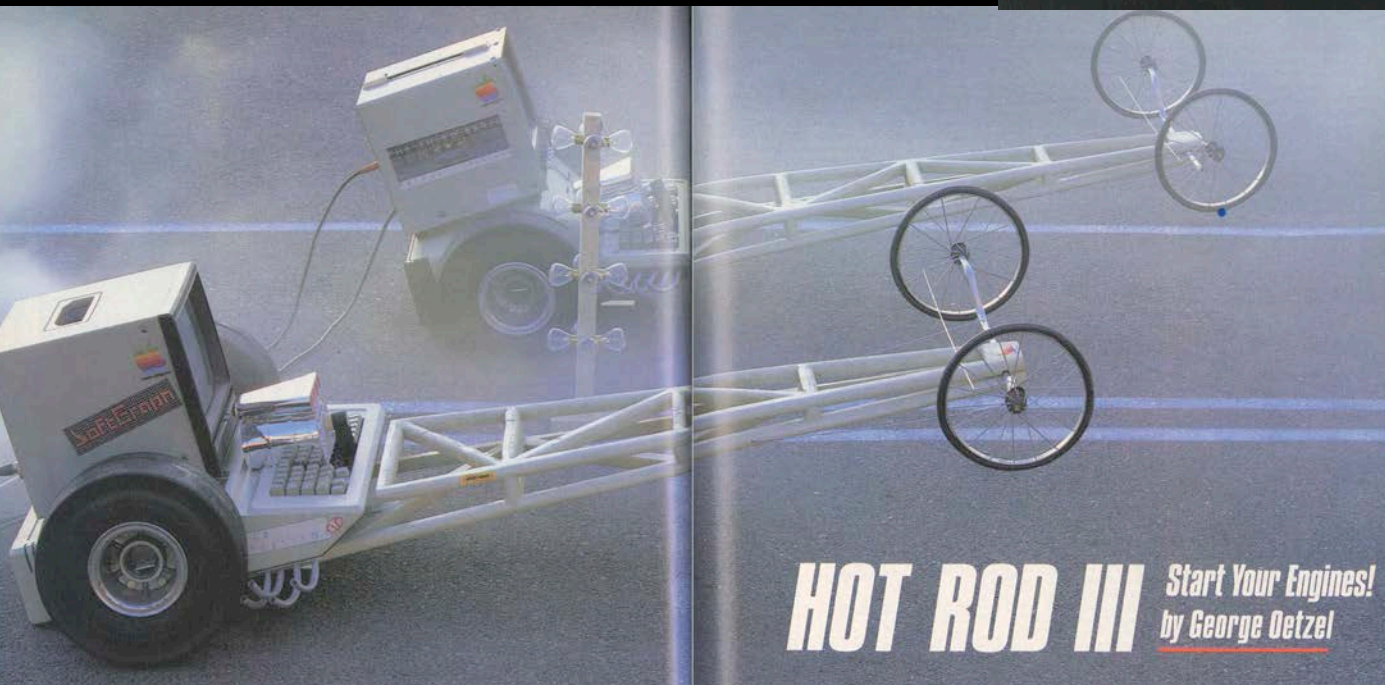
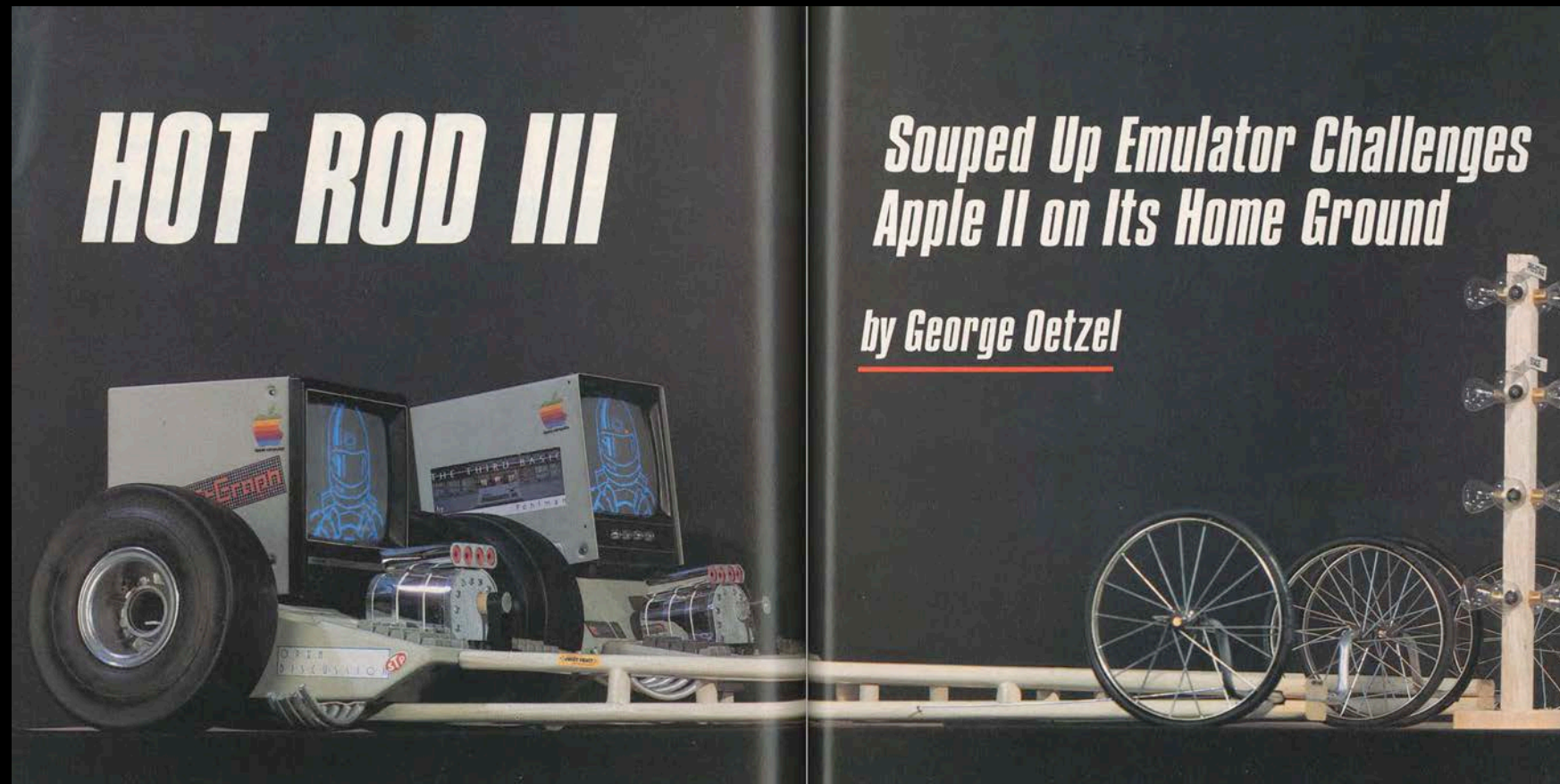
Destination Address	Boot Address	Disk Block	Description
Integer Basic Image			
C500–C5FF	2000–20FF	10	Slot 5 (Comm card) ROM
C600–C6FF	2100–21FF	10	Slot 6 (disk) ROM
C700–C7FF	2200–22FF	11	Slot 7 (Comm card) ROM
C800–CFFF	2300–2AFF	12–15	Expansion I/O ROM (empty)
D000–D7FF	2B00–32FF	15–21	Programmers aid #1
D800–DFFF	3300–3AFF	21–23	D8 ROM (empty)
E000–F7FF	3B00–52FF	23–29	Integer Basic
F800–FFFF	5300–5AFF	29–2D	Autostart Monitor
Applesoft Basic Image			
C500–C5FF	5B00–5BFF	2D	Slot 5 (serial card) ROM
C600–C6FF	5C00–5CFF	2E	Slot 6 (disk) ROM
C700–C7FF	5D00–5DFF	2E	Slot 7 (Comm card) ROM
C800–CFFF	5E00–65FF	2F–32	Expansion I/O ROM (empty)
D000–F7FF	6600–8DFF	33–46	Applesoft Basic
F800–FFFF	8E00–95FF	47–4A	Autostart Monitor

Table 1. Address guide to the two Basic images after booting the Emulation disk. All addresses and disk blocks are hexadecimal values.

If they're loaded from disk, they can be changed on disk. You can rewrite your ROM, Applesoft, disk/serial/comm card firmware. You can actually also change the font.

GEORGE OETZEL: HOT ROD III (SOFTALK JUL, AUG, SEP 1983)

- Three part series about tweaking Apple II software to run on the Apple /// with access to Apple /// hardware.

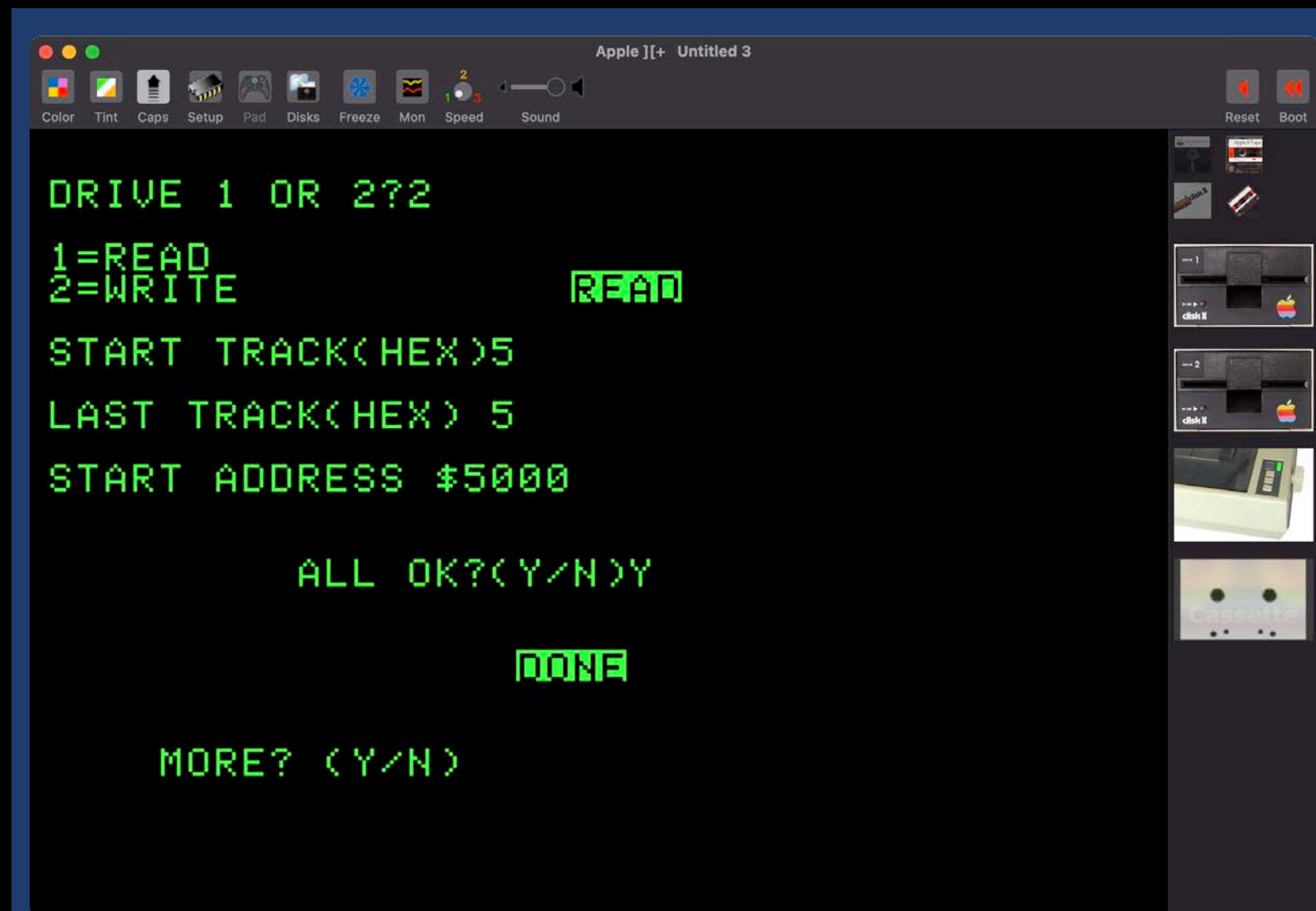


GEORGE OETZEL: HOT ROD III (SOFTALK JUL, AUG, SEP 1983)

- **Part 1:** Modifying the Apple II Emulation disk to modify ROM: Reset to monitor instead of to reboot. Provides TRACKMOVER track-at-once loader/saver.
- **Part 2:** Swapping in ROM routines to handle Apple /// game controllers and patching games to use them.
- **Part 3:** Changing the character set, handling lowercase display and input. Expanding memory a bit, speeding up the processor, "hybrid mode."

TRACKMOVER

- *Softalk* type-in program, included on the Washington Apple Pi **EMM 11b** disk (translated to Applesoft).
- Surprisingly useful. I ran it almost exclusively in an Apple II emulator, since it is DOS 3.3 Applesoft.



WHAT THE EMULATION DISK DOES

- After setting parameters, the emulation disk loads in the appropriate ROM images to \$C0-FF.
- Sets the environment variables.
- Turns on emulation mode (no more access to VIAs), boots.

```

; $FFD0 is the zero-page control register.
A56F - A9 00      LDA #$00          ; Select zero page = 0.
A571 - 8D D0 FF   STA $FFD0

; $FFDF is the environment control register.

A574 - A9 FC      LDA #$FC          ; Select environment—
A576 - 8D DF FF   STA $FFDF         ; discussed below.

; $FFE3 selects memory bank and I/O status.

A579 - AD EF FF   LDA $FFE3         ; Retain same
A57C - 8D EF FF   STA $FFE3         ; memory bank.

; $FFE3 is the data direction register for the
; A port of the E VIA. Set the
; Emulation bit to output status, so the
; STA $FFE3 will turn on Emulation.

A57F - AD E3 FF   LDA $FFE3         ; Set the Emulation mode
A582 - 09 40      ORA #$40          ; bit in data direction
A584 - 8D E3 FF   STA $FFE3         ; control register.
A587 - AD EF FF   LDA $FFE3         ; Reselect memory
A58A - 29 B0      AND #$B0          ; bank 0, and turn on
A58C - 8D EF FF   STA $FFE3         ; Emulation mode.
```

FUNNY MODE

(SILLY MODE, HYBRID MODE, SATAN MODE)

- The experiment: [What if we did NOT turn on Apple II Emulation mode?](#) Retaining access to the VIA registers and the general features of the Apple ///?
 - access to memory beyond 64K (including ROM) works quite differently. Things written for the Apple II are not going to address it correctly. And some hazard using indirect ZP.
 - VIA registers sitting at an important end of the monitor ROM.
 - Graphics/text handling (additional modes) differs.
 - Apple II stuff was largely written essentially right on the metal.

BOB ETHEREDGE VS. FUNNY MODE

Internally, a well-known case was made against encouraging use of "funny mode." Partly against for support/marketing reasons, but outlines technical challenges.

July 17, 1981

To: Dave Paterson, Mike Kane, Wil Houde, Ken Victor

Cc: Rob Campbell, Barry Yarkoni, Taylor Pohlman, John Santler, Steve Bareilles, Bob Martin, System Software, Applications Software

From: Bob Etheredge

Subject: Funny Mode and SOS - Implications for the Apple ///

Abstract

The purpose of this document is to outline some of the potential costs associated with DOS 3.3 applications operating in Apple /// native mode (the so called "funny mode"). It is my contention that tacit approval of this software environment threatens the financial success of the Apple /// system, both in the short and the long terms.

It is my contention that encouraging OEM software developers to use funny mode is both a short cut "panic" response to insufficient system sales, and a major threat to the success of the Apple ///, short-term and longterm. What follows is an analysis of the dangers funny mode poses to the Apple ///.

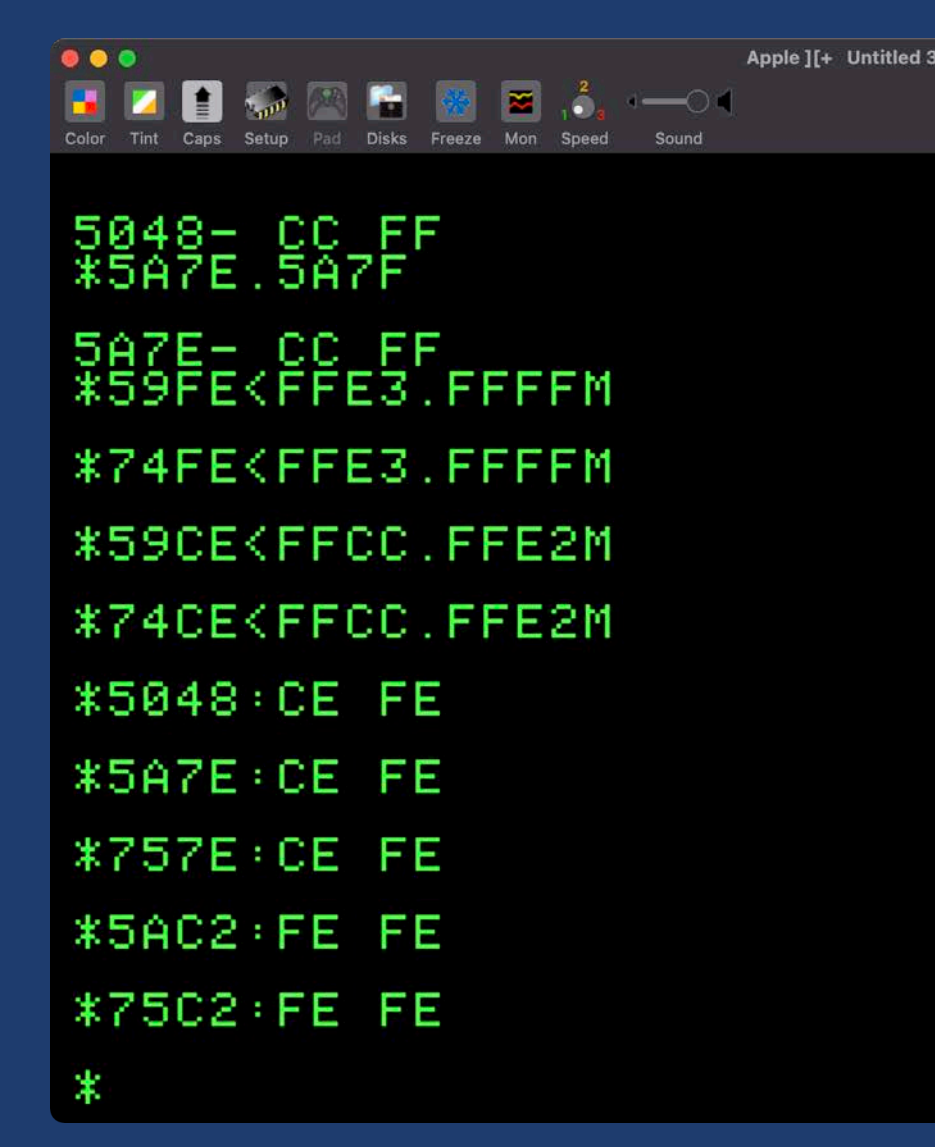
BUT LET'S TRY ANYWAY

- Let's see how it goes. What problems arise, how can we get around them?



MONITOR PROBLEMS

- In the monitor ROM, the location of the VIA registers is right in the middle of the command table. So you can CALL-151, but because part of the table is missing, it is not going to understand you.
- So, [move the tables](#). The /// has no cassette port, so we can use the cassette read and write areas. (Oetzel outlined what needs changing, I adapted it slightly.)
- TRACKMOVER: read track 5 to \$5000, 9 to \$7000, fix, write back out

A screenshot of an Apple II monitor window titled "Apple][+ Untitled 3". The window has a menu bar with icons for Color, Tint, Caps, Setup, Pad, Disks, Freeze, Mon, Speed, and Sound. The main area displays green text on a black background, showing hex addresses and data. The text is as follows:

```
5048- CC FF
*5A7E .5A7F

5A7E- CC FF
*59FE<FFE3 .FFFFM
*74FE<FFE3 .FFFFM
*59CE<FFCC .FFE2M
*74CE<FFCC .FFE2M
*5048:CE FE
*5A7E:CE FE
*757E:CE FE
*5AC2:FE FE
*75C2:FE FE
*
```

TWEAK ENVIRONMENT REGISTER

- Emulator disk sets: **FC**
 - 1MHz, \$Cxxx I/O, video on, reset enabled
r/o high memory, normal stack, \$Fxxx from RAM.
- If we want access to more memory (or to swap in INT on the go or something), we can unlock it: **F4**

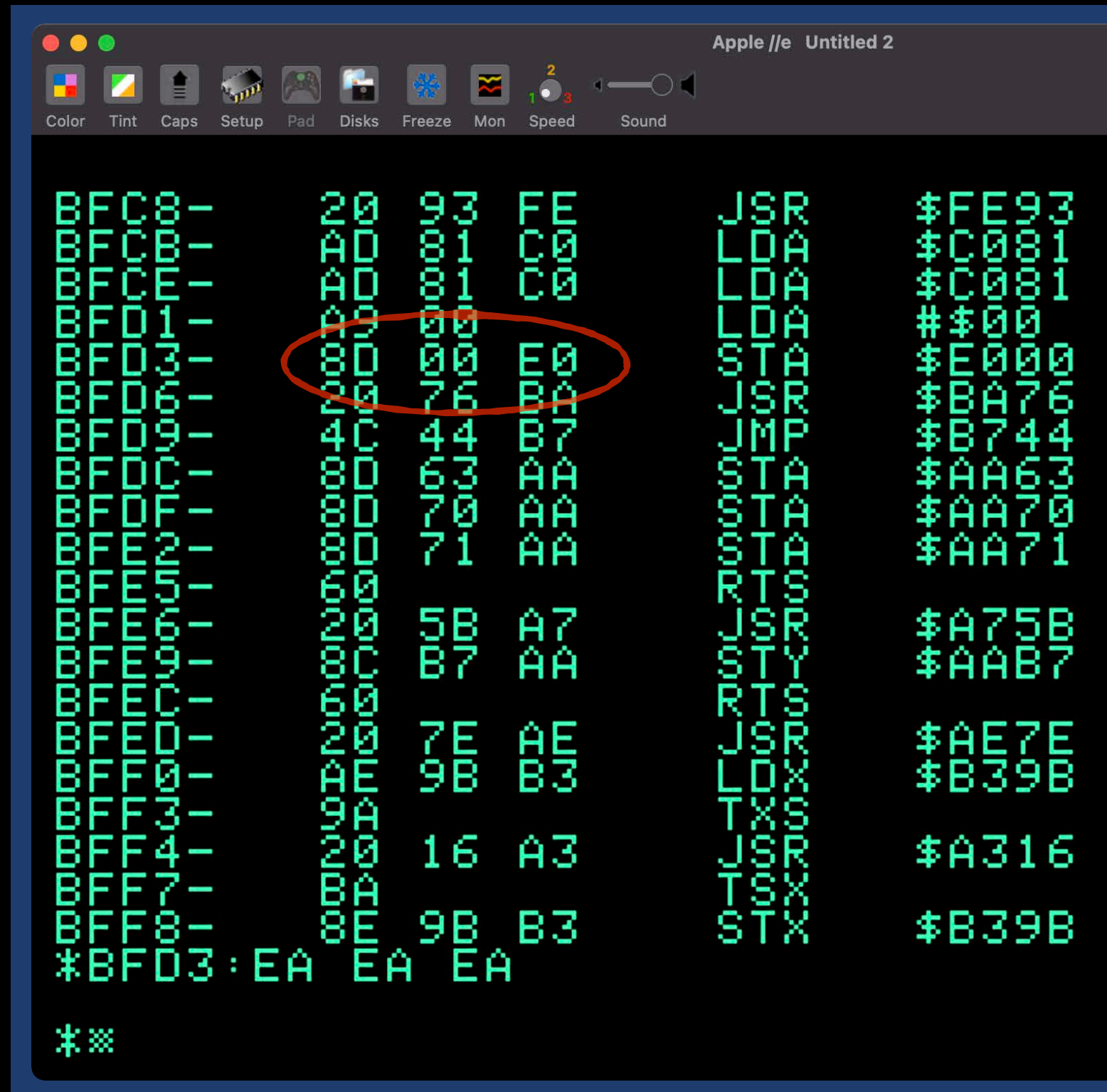
Value	Bit	Function	Bit = 0	Bit = 1
01	0	F000..FFFF	RAM	ROM
02	1	ROM#	ROM#2	ROM#1
04	2	stack	alternate	normal (true 0100)
08	3	C000..FFFF	read/write	read only
10	4	reset key	disabled	enabled
20	5	video	disabled	enabled
40	6	C000..CFFF	RAM	I/O
80	7	clock speed	2MHz	1 MHz

Note: ROM#2 doesn't exist.

Table 6. Environment register (\$FFDF).

MAKE DOS 3.3 LESS DANGEROUS

- DOS 3.3 will explicitly store a 00 in \$E000 as it boots, to force the language card to reload.
- This will damage BASIC and we don't have a language card. We don't want this.
- NOP NOP NOP the code
 - BFD3:EA EA EA
- Format a new blank disk



AND NOP OUT EMULATION MODE

- TRACKMOVER:
read track 0 to
\$4000.
- 4574: F4
- 4582: EA EA
- Write back out.

```

; $FFD0 is the zero-page control register.
A56F - A9 00      LDA #$00      ; Select zero page = 0.
A571 - 8D D0 FF   STA $FFD0

; $FFDF is the environment control register.
A574 - A9 FC      LDA #$FC      ; Select environment—
A576 - 8D DF FF   STA $FFDF     ; discussed below.

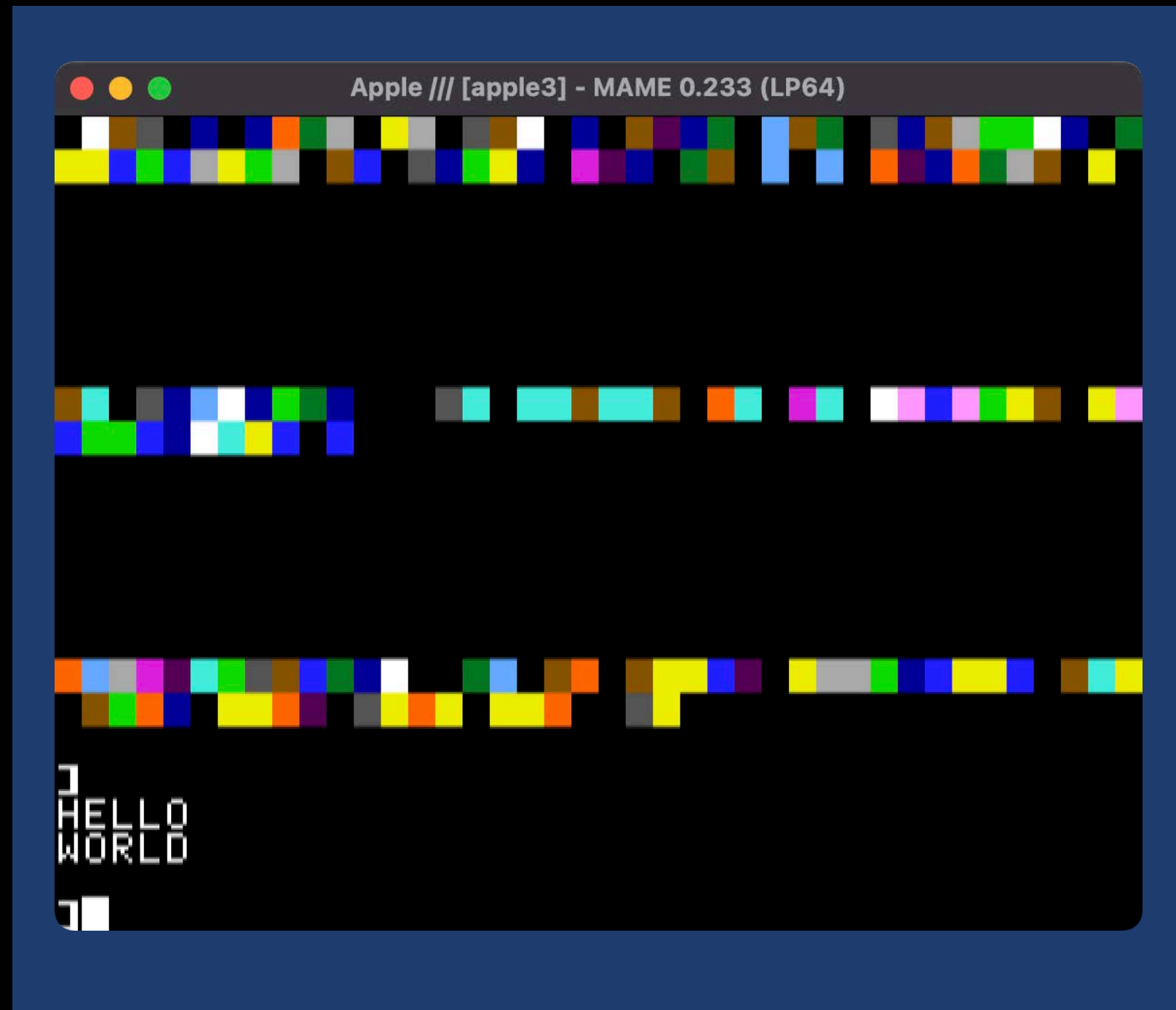
; $FFE3 selects memory bank and I/O status.
A579 - AD EF FF   LDA $FFE3     ; Retain same
A57C - 8D EF FF   STA $FFE3     ; memory bank.

; $FFE3 is the data direction register for the
; A port of the E VIA. Set the
; Emulation bit to output status, so the
; STA $FFE3 will turn on Emulation.

A57F - AD E3 FF   LDA $FFE3     ; Set the Emulation mode
A582 - 09 40      ORA #$40      ; bit in data direction
A584 - 8D E3 FF   STA $FFE3     ; control register.
A587 - AD EF FF   LDA $FFE3     ; Reselect memory
A58A - 29 B0      AND #$B0      ; bank 0, and turn on
A58C - 8D EF FF   STA $FFE3     ; Emulation mode.
```

HOW'S IT GOING SO FAR?

- Interesting. Ok, it boots. But it looks kind of oddly colorful. Why?
- Text page 1 is \$400 just like on the Apple II.
- Text page 2 is at \$800 **and controls the foreground and background color of the character on page 1.**
- We could clear it to white on black, but... BASIC programs live at \$800.



MOVE BASIC TO START AT \$C00

- We can tell Applesoft to put its programs at \$C00 instead of \$800.
- This does that, part of a larger embedded utility designed to run at boot time. Moves BASIC to \$C00, initializes the program space (NEW), then sets the colors to B&W.
- I call this in a HELLO EXEC that BRUNs this and then runs HELLO2.

[illegible]

FUNNY UTILITIES

- I adapted Oetzel's utilities into an Applesoft &-handler (AMPERSILLY). BRUN it and it
 - Sets up the & handler
 - Relocates BASIC to \$C00
 - &1/&2 for 1MHz/2MHz speed
 - &F/&N for video off/on
 - &L set background of line in \$40 (64) to pattern in \$E3 (227).
 - &C clear text background to white on black (set it all to \$F0).

NOW WHAT?

- I've tried a few things as proofs of concept.
 - Detecting shifted keys by looking at the \$C008 keyboard flags.
 - Speeding up and slowing down the processor (don't try to do I/O at 2MHz!).
 - Beeping with the built-in Apple /// beeper at (\$C040).
 - Scrolling graphics, bouncing a ball in the colors of page 2 without affecting (or while independently scrolling) the text of page 1.
 - Swapping in other banks of memory (but beware: you need to have \$9D00-\$9FFF in banks you switch in if anything like character I/O talks to DOS).
BRUN BANKDOS500,A\$500 will copy 9D00-9FFF from bank F0 to bank F1.

OTHER THINGS YOU CAN DO

- Oetzel's articles cover:
 - Changing the font (Apple /// font is replaceable in software), you can use fonts from the DOS Toolkit.
 - Making Reset dump you to the monitor rather than reboot the machine so you don't have to go through the Emulation boot step again.
 - Allowing for lowercase display and input by checking the shift key and adding lowercase fonts, removing FLASH characters.
- Also I fiddled with:
 - Putting the emulator ROMs and DOS 3.3 on the same disk so you don't have to swap disks. Uses something like half the disk. See also Martin Haye's 2017 HackFest entry.

CAN I RUN AIRHEART ON AN APPLE /// LIKE SUPER FAST?

- This is not really a route to just running Apple II stuff on an Apple ///, mostly.
- It's providing access to the Apple /// hardware within an Apple II framework, but things written for DHGR, or for 64K/128K RAM cards can't address Apple /// memory. It was baroque on the Apple II, too.
- Best option is to *rewrite* Apple II things, but maybe minimally, to address memory or graphics modes properly.
- But maybe as interesting might be just writing new things on the Apple ///, but with some of the comforts of the Apple II available.