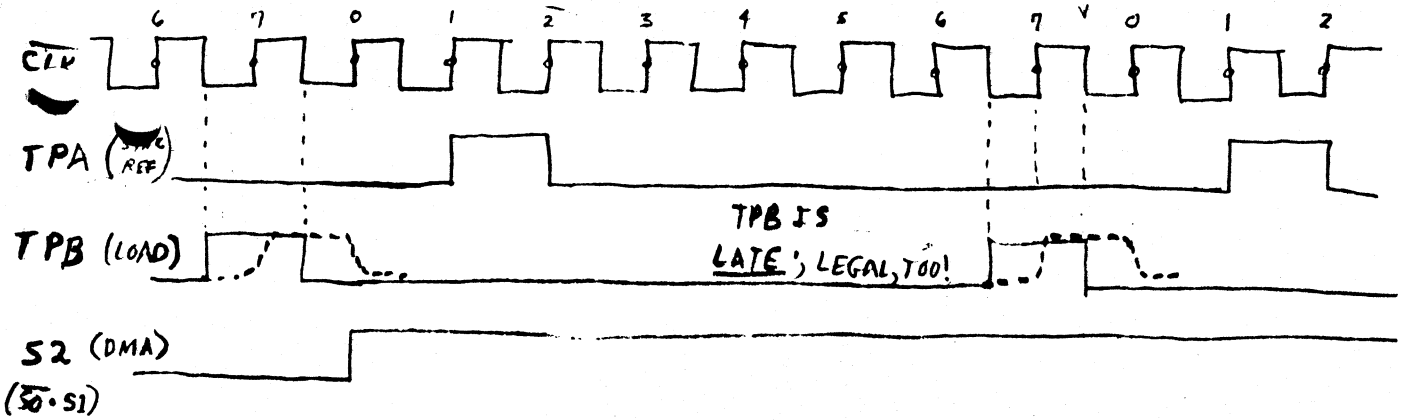
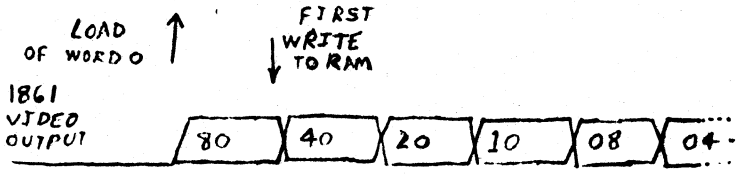
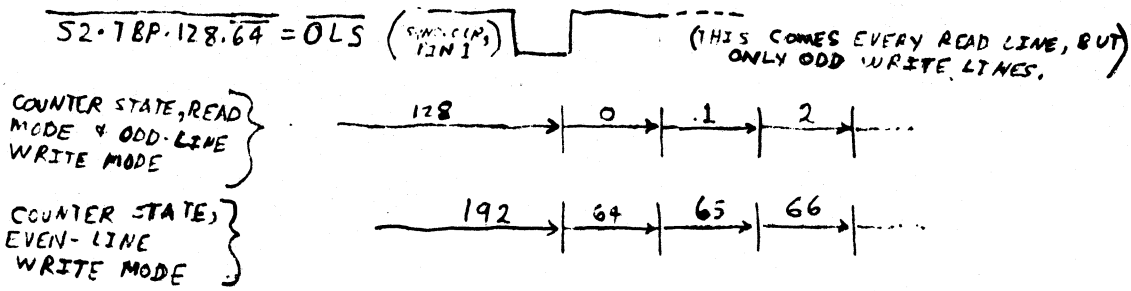
VIPER

VOLUME 1 FEBRUARY ISSUE 7







$2.00

AN ARESCO PUBLICATION

EDITORIAL

"Those not busy bein' born are busy dyin'" goes the Bob Dylan song. We try to grow a little with each issue of The VIPER, to do more of what works and less of what doesn't. A case in point is our recent attempt to provide royalties for authors of excellent VIP software by selling, rather than publishing their works. This concept had been very well received in our other publications, but we received more complaints than orders from VIPers. LIFE and BASEBALL will continue to be available, but we won't make any more additions to the library.

Our recent issues have each centered around a theme, such as Games, Printers, or Editors. I guess this is our potpourri issue. Our feature article is one I never thought I'd see - a hardware modification to provide double-resolution horizontal displays. Since this is a pretty complex construction project, we'd like to hear from anyone who makes up a PC board for the circuit - we'd be happy to distribute them to VIPER readers.

I'm already putting the next issue together. It will have a great Lunar Lander program, and an article on getting CHIP-8 up on the ELF II. We'll also have the conclusion of the Studio II conversion series. See you next month.

Rick Simpson

SUBSCRIPTION RATES, ADVERTISING RATES AND OTHER ESSENTIAL INFORMATION

The VIPER is published ten times per year and mailed to subscribers on the 15th day of each month except June and December. Single copy price is $2.00 per issue, subscription price is $15.00 per year (all ten issues of one volume). Dealer prices upon request. Outside of Continental U.S. and Canada, add $10.00 per subscription for postage ($1.00 for single copy).

The VIPER is an Aresco Publication, edited by Rick Simpson. For information contact Editor, VIPER, P.O. Box 43, Audubon, PA 19407.

CONVERTING THE STUDIO II PART 2

Constructing the Card

The schematic for the plug-in PROM card is shown in Figure 3. It uses a Harris 7641 fusible-link PROM (tri-state outputs). The circled letters are the edge-connector pins. (5/2 x 8).

The program for the PROM is shown in Figure 4. Since the Studio II uses primarily CMOS logic, the 4050s were included on the card to buffer the address lines. The 4042 and 4001 latch and decode the high-order addresses (400 through 05FF). The diode to CE1 disables the Studio II RAMS whenever the PROM is addressed.

Before starting the construction, make sure the card you're using will fit into the Studio II socket, clearing the aluminum grounds pins on each side.

When using the card, the PROM will draw an additional 100 mA or so from the Studio II power supply, putting the total drain close to the 250 mA maximim of the 9V power supply. On most Studio IIs, this will cause no problem unless you have a marginal power supply. If you have problems and suspect the supply, measure the 9V with the prom card plugged in. If it has dropped to 7V or less, you'll need to try another power supply.

Note: Keep in mind that most of the ICs used on the plug-in card are CMOS and thus can be damaged by static electricity. Use care when handling the card under dry humidity conditions.

Using the PROM Card

Once the programmed PROM card is inserted in the Studio II, and of four "modes" may be selected via keyboard A:

Load Mode (Key 1)

When key 1 is pressed, the Studio II waits for the starting address of the program you intend to load. You enter this address in hex notation, using the A keyboard for digits 0 through 9 and the B keyboard for digits A through F (A=1, B=2, C=3, D=4, E=5, F=6). An overlay for the B keyboard, which you can make from thin, clear plastic, will make it easier. The address wil be displayed on the TV. The normal start address is 0800, and your program can normally exend

from there to address 08FB. There is also RAM from address 0900 to 09FF: this RAM is normally bit-mapped onto the display as described earlier, but may be used for program storage if your program doesn't need a full display. Programs as complex as a "Lunar Lander" program can be written in this small memory by making use of subroutines that are in the plug-in PROM and which can be called by your program. These routines will be described next month.

After you enter the four-digit start address, you next enter the first byte of the program. Then continue to key in your code through the keyboards, one byte after another, until you're done. You'll notice that the address display automatically increments itself as you enter each byte, and that each data byte is displayed as it is entered.

Memory-Read Mode (Key 2)

When key 2 is pressed, enter the address of the memory location you wish to examine. Pressing any key on keyboard A will then display the byte stored at that location.

Continuing to press any key on the A keyboard will sequentially step through memory, displaying its contents and incrementing the address display at the same time.

Run Mode (Key 3)

The program that has just been loaded can be run by pressing key 3 and then entering the four digit start address of the program. The program will start running as soon as the last digit of the start address has been entered, and will use R9 as the program counter.

Shift Mode (key 4)

Suppose you have loaded a program and then want to add several bytes in the middle of it. Pressing key 4 and then entering a four-digit start address will allow you to shift the entire program, beginning with this address, one location higher in memory. The shifting is done by pressing either key 8 or key 9 on the A keyboard. Press key 8 to shift page 8 (0800-08FB) or press key 9 to shift page 9 (0900-09FF). Each key press will shift the entire program after the shift start address one location higher in memory.

When you have completed your entries in one mode and wish to switch to another mode, press key 0 on the B keyboard. DO NOT PRESS "CLEAR"!!!

Pushing "CLEAR" will destroy part of the program you have just entered!

After key 0 has been released, you may select the next mode. Pressing key 0 on the B keyboard also causes the control program in thePROM to store most of the 1802 internal 16-bit registers in memory so they can be examined using the Menory Read Mode. This is very useful when trying to debug a program.

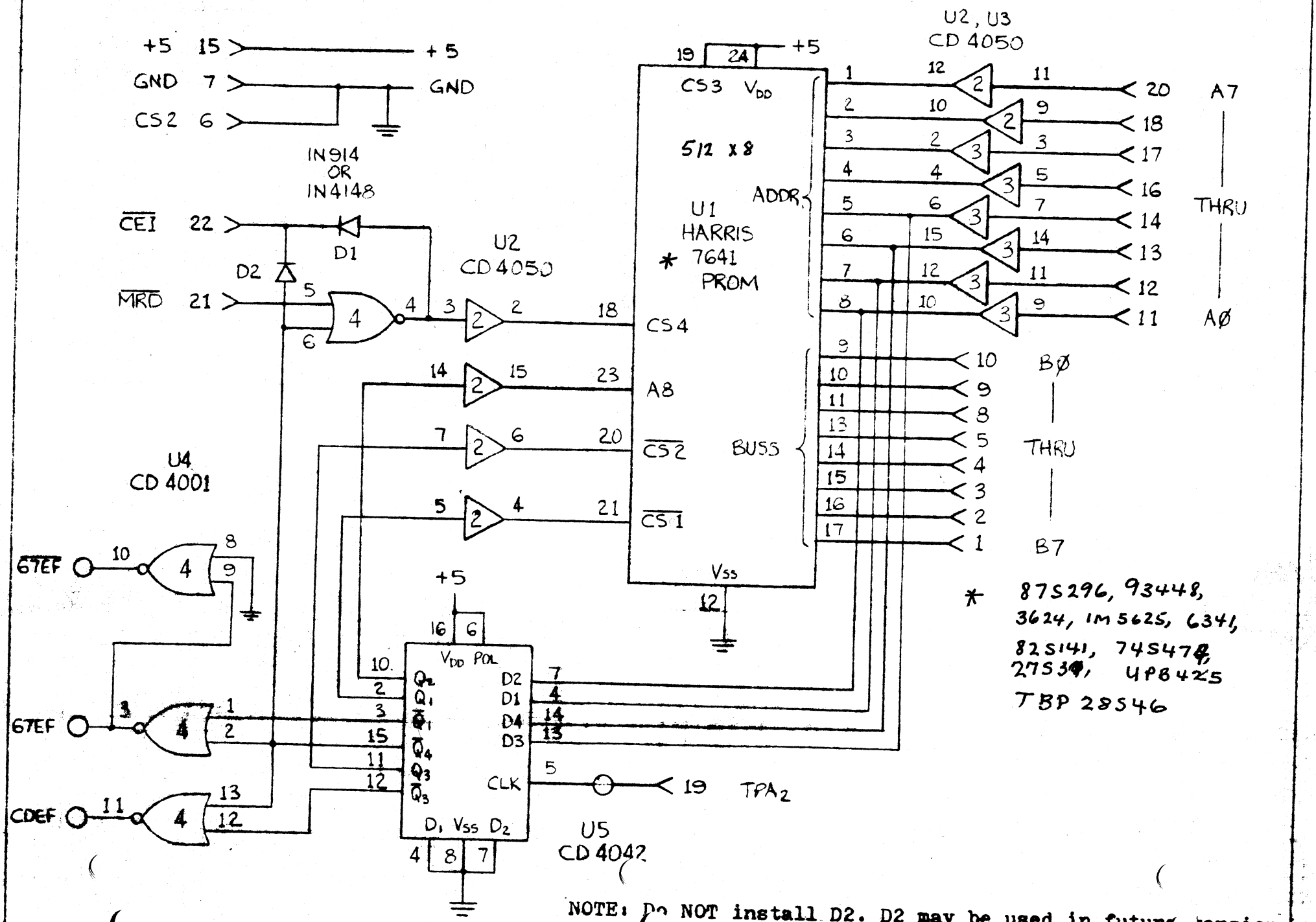
All registers except R0,R1, R2, and R8 are stored in the RAM at addresses 09B3 to 09CF. The last digit of the address is the register number; if the third address digit is B, it's the low-order register address. For example, 09B5 contains the contents of 1802 register R5.0 (the low-order 8 bits of register R5), While 09C5 contains the contents of register R5.1 (the high order 8 bits of register R5).

NEXT MONTH - Programming the System

ADDRESS	DATA (HEX)							
0400	0402	F804	B1B5	F8AA	A1F8	ODA5	D5F8	09BB
0410	BDB2	F800	ABA2	22F8	05B3	BC7A	E2E2	F874
0420	A3D3	30E2	D3F8	DCAD	D3D3	86B9	D3F8	D2AD
0430	D3D3	86A9	97FB	0132	7497	FB03	3245	97FB
0440	0432	8A30	46D9	F874	A3D3	09A6	F8D6	A3D3
0450	F8D6	ADD3	F874	A3D3	1999	A6F8	D6A3	D3F8
0460	DOAD	D389	A6F8	D6A3	D3F8	D2AD	D397	FB01
0470	327C	304A	D3F8	D6AD	D3D3	8659	19F8	70A3
0480	D3F8	D6AD	D3D3	8659	3059	F874	A3D3	94B6
0490	F8FC	A646	5626	E686	7316	89F3	263A	93E2
04A0	308A	OFOF	OFOB	030F	7270	C422	7822	528B
04B0	A09B	B028	E2E2	80E2	20A0	E220	AOE2	20A0
04C0	E23C	B622	F800	5262	37D0	2284	5262	30A8
04D0	F805	B8F8	43A8	37D6	3CD8	34DA	72F8	2852
04E0	30A9	94B7	F870	A330	24XX	XXXX	XXXX	XXXX
04F0	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
0500	3039	222A	3E20	2434	2628	2E18	141C	1012
0510	F080	F080	F080	8080	F050	7050	F050	5050
0520	F080	F010	F080	F090	F090	F010	F010	F090
0530	F090	9090	F010	1010	1060	2020	2070	A0A0
0540	F020	20F8	D1A2	F809	B2BD	F8D8	73F8	9F52
0550	92B1	82A1	22D1	7321	2141	21FF	0151	FB82
0560	3A55	F804	B1B5	F8AA	A1F8	CFAD	1D30	F2D5
0570	F8FF	A638	93A6	E214	8452	6222	3682	3F77
0580	FC09	FA0F	B47B	F804	A888	3A89	F804	A836
0590	8C37	8C88	30BD	97D5	96AC	0CAC	F805	A7F8
05A0	005D	8DFC	08AD	4C5D	8DFC	08AD	2787	3AA6
05B0	31BA	7B8D	FF2F	AD94	3099	7A30	E33A	8F7A
05C0	86FB	FF32	C730	9794	B630	74XX	XXXX	XXXX
05D0	XXXX	XXXX	XXD5	86F6	F6F6	F6B6	86FA	0FB4
05E0	3097	XXD5	96FE	FEFE	FE52	94F1	A630	6FXX
05F0	XXXX	F800	5D8D	FBFF	3A6C	F80D	A5D5	XXXX

Figure 4 - PROM Listing

FIGURE 3 - STUDIO II ROM CARD Schematic



NOTE: Do NOT install D2. D2 may be used in future version.

DOUBLE-BUFFER SPEEDUP HARDWARE FOR 64X128 GRAPHICS WITH THE COSMAC 1802 AND THE 1861 VIDEO CHIP

The accompanying diagrams and tables describe a "ping-pong" or double-buffer memory system which can be added on to an essentially unmodified 1802-1861 combination to double the available graphics resolution in both axes, with square picture elements. The add-on buffer is usable either with a VIP and the CHIP-10 interpreter, or with an ELF or other COSMAC system with machine language programs.

The complete system uses 18 ordinary integrated circuits, 14 and 16 pin DIP's. Two are CMOS hex buffers, two are high-speed 256X1 TTL RAMs, and the rest are common 7400 TTL. LS TTL could be used instead, and the package count could be reduced slightly by using data selector or ROM logic for control. The packaging consisted of wire-wrap sockets fastened with 2-56 screws to an OK Machine & Tool HB-1 Hobby Board; this board fits the 44-pin Expansion Interface socket on the VIP (left rear). If your VIP has no sockets, the OK CON-1 connector, designed for wire wrap, will fit with only slight dogleg bending of the wirewrap pins. Tie the bus strips on the card into interleaved grids, one for +5 volts and one for ground, with frequent cross-connections; distribute at least 5 or 6 0.01 uf disc ceramic bypass capacitors around the board, with short leads, between +5 and ground buses. Provide more D.C. power; the board takes about 300 ma with standard TTL. I use the stock VIP power supply just for the buffer board; the VIP itself (with 4K memory) now runs from a separate 1-amp supply.

Functionally, the buffer is simple: a pair of 128-bit memories (256-bit stock RAMS used inefficiently), with address counters and control. One of the two memory systems (called "Left" and "Right" on the schematics) accepts and stores two successive 64-bit lines of video output data from the 1861 at its standard 1.76 mHz rate. Meanwhile, the other memory system is putting out high-resolution video at a 3.52 mHz rate; it outputs the same information twice, on two successive TV scan lines. The stored video came, of course, from the 1861 on the previous two scan lines; the twin memory systems swap roles every two scan lines. The role swapping is controlled by the 7474, which functions as a 2-bit counter clocked by the horizontal sync pulses. The Q and $\overline{Q}$ outputs of this counter select either the 1.76 mHz or the 3.52 mHz clock for each of the 8-bit address counters (74163's), and switch the memory chips to either READ or WRITE mode as appropriate. The memories are Fairchild 93421's, very fast (35 nsec) TTL units with tri-state outputs.

The basic concept and operation of the buffer are, as described above, pretty straightforward; the device data sheets will clear up any subtle details. The control of the address counters is, unfortunately, much more complicated and hard to follow. Study the two tables which describe the sequence of counter states, STOP states, and control signals.

The basic requirement is that the counters start at exactly the right time, i.e., the beginning of each scan line, in the proper initial state (0 or 64), and stop counting and hold exactly 64 counts later (when writing) or 128 counts later (when reading out) as the video scan line ends. A pulse signal can easily be derived from COSMAC state code signals SC0 and SC1 gated with timing pulse TPB; the first occurrence of this pulse coincides with the beginning of the actual video output (left edge of the display window). This signal (called S2 TPB on the drawings) thus serves nicely to start the counters; however, there is no easy way to derive a corresponding signal at exactly the right time to stop the counters. S2, the DMA state, ends 1/8 line too soon. The solution chosen is to have the counters stop themselves when they reach the proper state; as noted in the table, this happens whenever the MSB=1 in state 128 or 192. The MSB is fed back through an inverter to the counter-enable input to do this.

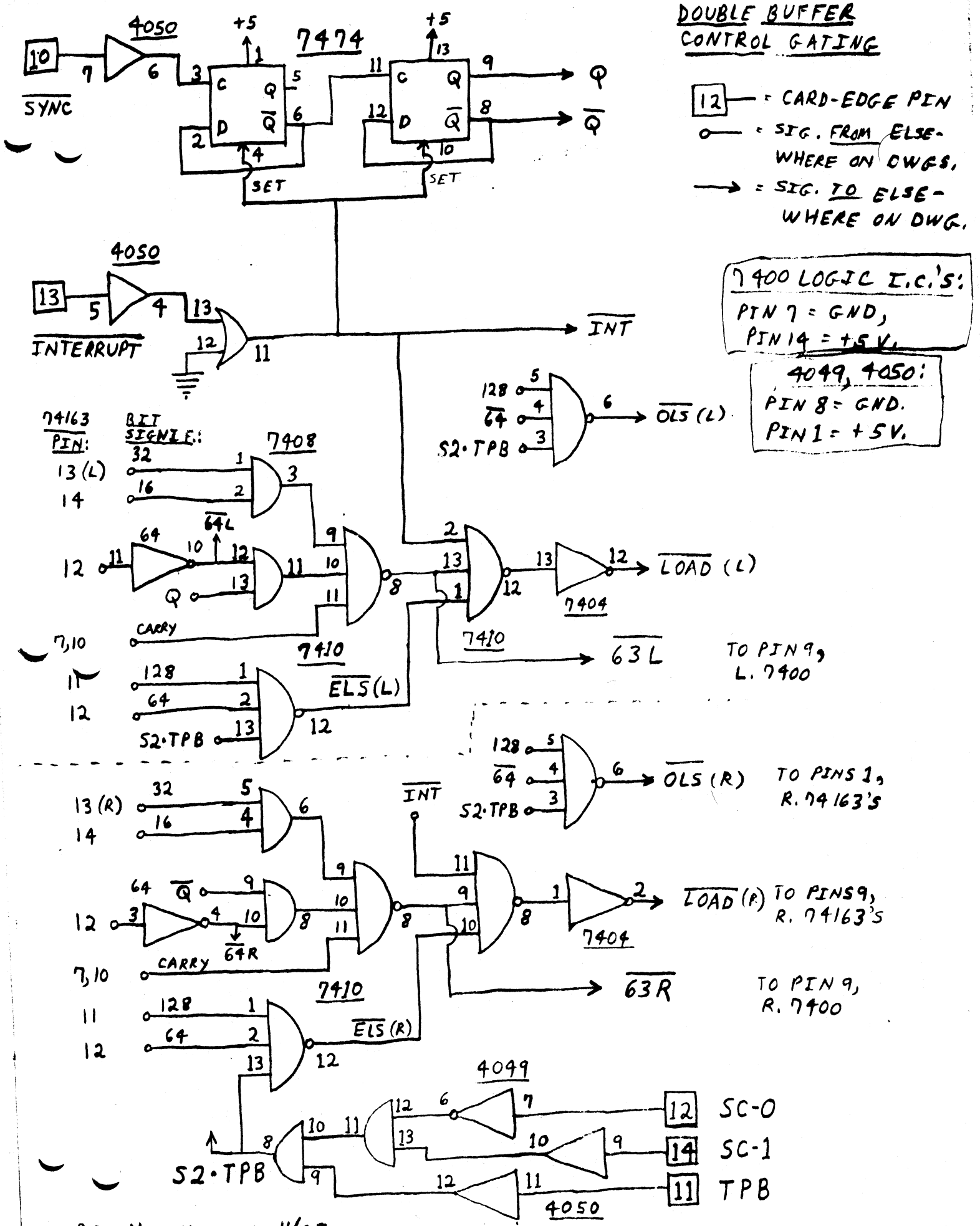
When in the read (output) mode, the counter reaches the proper stop state (128) naturally, after counting up from 0 to 127 in one scan line. When in the write (input) mode, the counter must stop after state 63. When a logic gate array senses state 63, the top two data-input bits of the counter are made = to 1, and the LOAD input is made active (=0), causing the counter to go to state 192 on the next clock pulse and stop there.

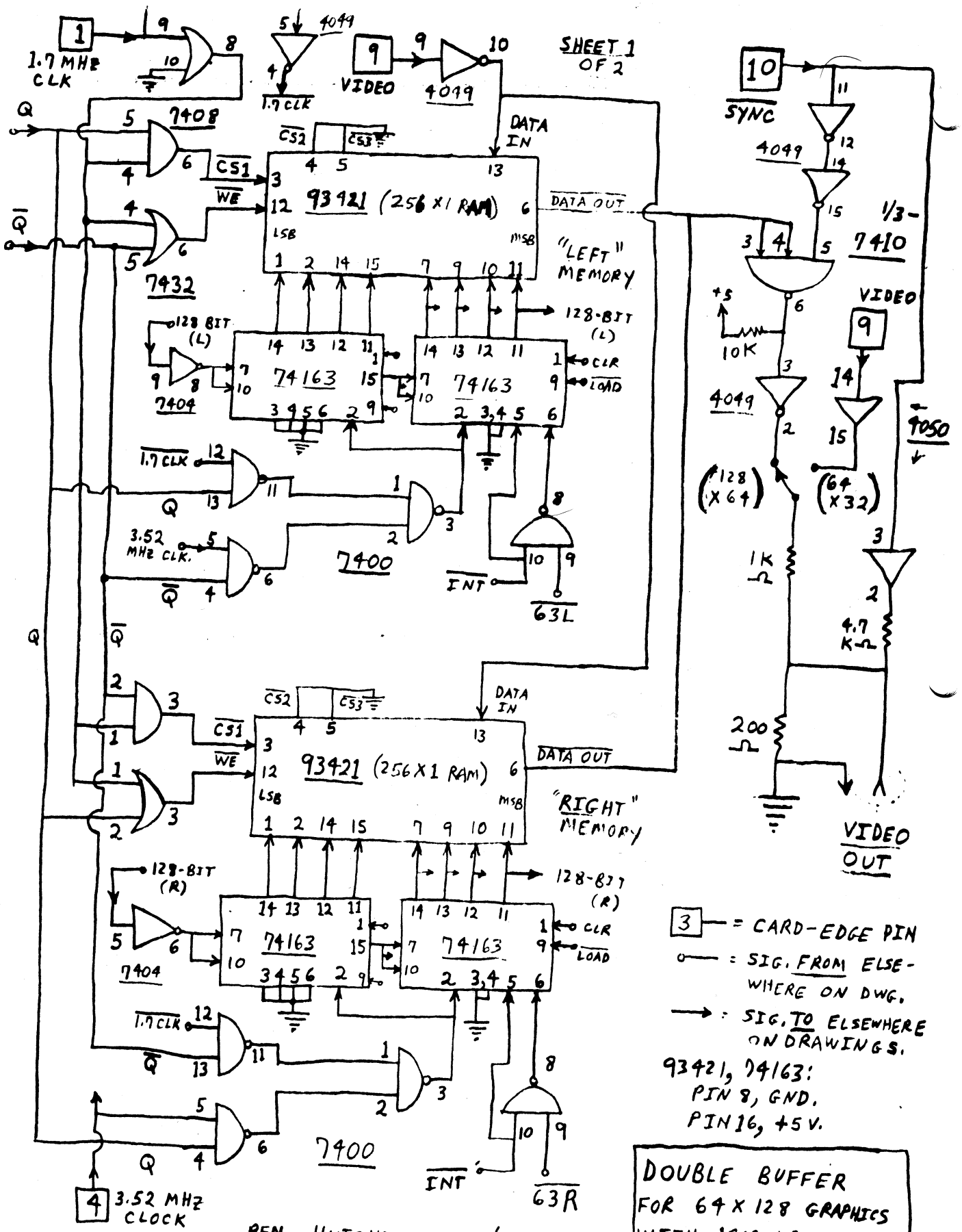
The S2 TPB signal actually occurs 8 times during each scan line, but only the first occurrence is wanted. Furthermore, the counter must be either set to 0 (if it's stopped in state 128) or to 64 (if it's stopped in state 192). These requirements are met by gating S2•TPB so that it only causes counter reset or jam set to 64 when the counter is already stopped at either 128 or 192 respectively.

The INTERRUPT signal from the 1861 occurs at the beginning of the video frame, and sets both address counters and the ping-pong counter to the proper initial states to start everything off on the proper foot. The first (top) line seen on the screen is actually the last line from the previous frame, sometimes with some displacement. This effective loss of one line out of 64 seemed to me not worth the trouble to fix.

The only actual mod needed to the VIP hardware is to lift the 200 ohm video output summing resistor from ground so that the 1861 output signals will have enough voltage swing to drive the CMOS buffer gates. Standard 32X64 operation can be had either via the switch with the board plugged in, or via the normal VIP video output by re-grounding the 200 ohm resistor, with or without the card plugged in.

As indicated briefly in accompanying waveform sketch, the TPB signal does not occur sharply coincident with the 1.76 mHz clock edges, as drawn so prettily on all the RCA data sheets for the 1861 and 1802. It happens delayed by at least half the period of the fast clock; furthermore, if you pore over the fine print of the data sheet, such a delay is apparently quite legal! The solution was to invert the 1.76 mHz clock fed (via gating) to the 74163 counters (using a 4049 section, shown at the very top of the drawing). Without this, the high-resolution picture is split by a dark vertical line, with the halves left/right swapped.


DOUBLE BUFFER CONTROL GATING
12 = CARD-EDGE PIN
SIG. FROM ELSEWHERE ON DWGS.
SIG. TO ELSEWHERE ON DWG.
7400 LOGIC I.C.'S:
PIN 7 = GND,
PIN 14 = +5V.
4049, 4050:
PIN 8 = GND.
PIN 1 = +5V.
4050
7474
+5
SYNC
SET
INTERRUPT
INT
74163 PIN:
BIT SIGNLE:
7408
7410
7404
64L
CARRY
OLS (L)
LOAD (L)
63L
ELS (L)
TO PIN 9, L. 7400
OLS (R)
TO PINS 1, R. 74163's
LOAD (R)
TO PINS 9, R. 74163's
64R
63R
TO PIN 9, R. 7400
4049
SC-0
SC-1
TPB
S2·TPB
- BEN HUTCHINSON, 11/78


SHEET 1 OF 2
1.7 MHz CLK
7408
7432
7404
7400
4049
VIDEO
SYNC
1/3-7410
7050
93421 (256 X 1 RAM)
"LEFT" MEMORY
"RIGHT" MEMORY
74163
DATA IN
DATA OUT
CS1
CS2
CS3
WE
128-BIT (L)
128-BIT (R)
CLR
LOAD
INT
63L
63R
10K
1K
4.7 K
200
(128 X 64)
(64 X 32)
VIDEO OUT
3.52 MHz CLOCK
3 = CARD-EDGE PIN
o— = SIG. FROM ELSEWHERE ON DWG.
→ = SIG. TO ELSEWHERE ON DRAWINGS.
93421, 74163: PIN 8, GND, PIN 16, +5V.
DOUBLE BUFFER FOR 64 X 128 GRAPHICS WITH 1802 + 1861
BEN HUTCHINSON 11/78

CHIP-10 INTERPRETER FOR THE COSMAC VIP

The CHIP-10 interpreter is so named because it provides the full set of graphics and other instructions implemented by CHIP-8, but with a screen resolution of 64 square elements vertically by 128 elements horizontally; this requires 1024 rather than 256 bytes of display refresh memory, requiring 10 rather than 8 address bits. The interpreter was implemented by modifying the initialization, screen erase, and "show" (DXYN) routines of the CHIP-8 interpreter.

Since the 1861 video chip is incapable of more than 64 elements of horizontal resolution, a hardware add-on is necessary to get the 128-element horizontal resolution offered by CHIP-10. This device is conceptually simple: a pair of 128-bit memories, with address counters and control. One of the pair accepts and stores two successive 64-bit lines of video output data from the 1861 at its standard 1.76 mHz rate. Meanwhile, the other memory is providing the high-resolution video output, at a 3.52 mHz rate; it outputs the same information twice, on two successive TV scan lines. The stored video came, of course, from the 1861 on the previous two scan lines; the twin memory systems swap roles every two TV scan lines. This approach involves essentially zero modification of the original VIP hardware, and instant reversion to the standard 64X32 format whenever desired (e.g., to use the ROM operating system).

Naturally, the standard display refresh interrupt routine for 64X32 display, in the ROM, can't be used; it repeats the same information for four successive TV lines. A new routine is provided, and its entry address (0073) loaded into R1 during initialization. The software thinks that it and the 1861 are making a display which is 64 horizontally by 128 vertically, with elements 4 wide by 1 high; thus the routine at 0073 is modelled after the routine for this purpose on the RCA data sheet. It differs in that it includes the timer incrementing required by various CHIP instructions, and (most importantly!) a simple dumb delay loop that postpones return to the main program until the end of the TV frame. Without this, strange and unpleasant display strobe effects occur, plus erratic behavior of some CHIP-10 instructions.

Note that since CHIP-10 uses the top four pages for display refresh, Y=3 in a system with 2K of memory, while Y=B in a system with 4K of memory. Obviously, in a 2K system with CHIP-10, only page 2 is available for CHIP-10 instructions, plus a few leftover bytes on pages 0 and 3 usable for subroutines, tables, images, and the like. This is plenty for even a fairly elaborate cursor drawing game, or for the kaleidoscope, or for an improved space intercept game. With 4K, of course, CHIP-10 really comes into its own, with 8 pages available for user programs. Or, you can switch between two separate 4-page display buffers for smooth animation or LIFE, and still have room for very elaborate user programs (4 pages).

In a 4K system, it is convenient to load the DXYN routine on page B as a separate step. This could have been avoided, of course, by using page 2; however, CHIP-8 programs coded for page 2 could not then be used without identifying and changing all the branch and table addresses.

Some of the space vacated on page 0 by the eviction of DXYN is needed to accommodate the expanded, 4-page erase routine, and (as noted above) for the new interrupt service routine starting at 0070 (entry at 0073). This still leaves 89 bytes from 0091 to 00DD (hex) available for subroutines, tables, or the implementation of new instructions. One simple use for 16 bytes of this is to put in a fast, simple interrupt routine for display refresh which does not increment timers or wait for the frame end. This is usable only with machine-language programs, but speeds up such routines (notably LIFE) by about 30%. This routine is included in the listing. If used, R8, R9, and RB.0 become available in machine language routines.

I would be most interested to hear from anyone who has written programs in CHIP-10 to interestingly exploit the improved resolution. (Or, for that matter, in machine language). Or, if you have hardware or software difficulties, I will do what I can to answer your questions.

Ben H. Hutchinson, Jr.

CHANGES TO VIP KALEIDOSCOPE PROGRAM
TO MAKE IT WORK WITH CHIP-10

Address	Now	Change to:
0205	1F	3F
0207	0F	1F
0245	E0	C0
0249	1F	3F
0251	F0	E0
0255	0F	1F
025D	1F	3F
0265	1F	3F
026B	3F	7F

CHIP-10 INTERPRETER: LISTING

ADDRESS	HEX CODE	Mnemonics and Comments
0000	91 FF 03 BB	GHI,SMI-03,PHI: Get top page, sub.3, put in RB
04	FF 01	SMI-01: Calc. Y (1 page below display start)
06	B2 B6	PHI: Set R2 and R6 =Y (Working RAM page)
08	F8 CF A2	LDI-CF, PLO: Stack origin ← OYCF
0B	F8 73 A1	LDI-73, PLO: R1.0=73, entry for interrupt routine
000E	90 B1	GHI,PHI: Set R1.1=00
0010 to 005C	SAME AS ORIGINAL CHIP-8	
005D	03 for 2K system, 0B for 4K system	(working-RAM page, where new DXYN routine lives)
005E to 006C, and 006E-006F	SAME AS ORIGINAL CHIP-8	
006D	00	Because new DXYN starts at OY00
0070	-	spare
0071	42	LDA-R2 (Restores D from stack, increments R2)
72	70	Return (increments R2, restores X and P)
0073	C4	NOP, 3 cycle, for timing. ENTRY POINT
74	22 78 22 52	DEC, SAV,DEC,STR: Save Y,P,and D
78	19	INC R9
79	F8 00 A0	LDI-00,PLO-R0
7C	9B B0	GHI,PHI: DMA start page. Ready for DMA
7E	98 32 85	Same as original operating system display routine, except branch addresses
81	AB 2B 8B B8	
85	88 32 8C	
88	7B 28 30 8D	
8C	7A	
8D	34 71	If EF1=1, Br. to 0071 (Frame over)
8F	30 8D	Go to 008D: Tight loop to stall 'till frame end

CHIP-10 INTERPRETER LISTING, CONT.

Address	Hex Code	Mnemonics and Comments
0091	42	LDA: Restore D from stack, increment R2
92	70	Return. Restore X,P from stack, incr. R2
0093	C4	NOP, 3 cycle, for timing. ENTRY ADDRESS
94	22 78 22 52	Dec, SAVE, DEC, STR: Save X,P,D on stack
98	F8 00 A0	LDI-00, PLO-R0
9B	9B B0	GHI-RB, PHI-R0 : DMA START PAGE
9D	F2 E2	SEX-2: 2 cycle NOPS for sync (X=2 already)
009F	30 91	Branch to 0091 for return

Note: Above routine, 0091 through 00A0 inclusive, is needed only to speed up machine-language routines which use the display, not for normal CHIP-10 operation. To use with LIFE, change 02D4 from 73 to 93. (Also frees R8,R9,and RB.0 for mach.lang.use)

00A1 to 00DD — THIS SPACE AVAILABLE-vacated by eviction of old DXYN. Total of 73_{10} bytes, not counting above 16.

00DE	12 D4	INC,RET: Begin ERASE routine (entry at 00E0)
00E0	9B BF	GHI-RB, PHI-RF: Display start page into RF
E2	FC 04 22 52	ADI-04,DEC,STR: Put limit on stack
E6	93 AF	GHI, PLO; RF at begin. of 4page disp. buffer
00E8	93 5F	GHI,STR: SET DISP. MEM. WORD=0 (R3.1=00)
EA	1F	INC RF
00EB	30 F3	Branch around sub.-return routine
ED	00	(SPARE)

00EE to 00F2 — SAME AS ORIGINAL CHIP-8 (subroutine return routine)

00F3	9F F3	GHI,XOR: Has RF gone 1 page too far?
F5	32 DE	BR.IF 0: If so, done; return
F7	30 E8	BR. : If not, loop to continue erasing

00F9 to 00FF — SPARES (fill with 00 if you want to be neat)

Page 1: Same as original CHIP-8

CHIP-10 DISPLAY ROUTINE (for "Show" instruction, DXYN)

Y=3 with 2K of memory
Y=B with 4K of memory

Address	Hex Code	Mnemonics and Comments
0Y00	06 FA 07 BE	LDN,ANI,PHI:Fetch VX, select LS 3 bits, - RE.1
04	06 FA 7F	LDN,ANI: Get VX, select LS 7 bits
07	F6 F6 F6	SHR: Shift VX right 3 places
0A	22 52	DEC R2,STR: 40,20,10,08 bits of VX to stack
0C	07 FA 3F	LDN,ANI: Get VY, select LS 6 bits
0F	FE FE FE AC	SHL-3,PLO: Move left 3 places, store in RC.0
13	94 7E BC	GHI-R4(=0),SHLC, PHI:Store page-increment MSB
16	8C FE	GLO,SHL: Move one more bit left
18	F1 AC	OR, PLO: OR with LS 4 bits of VX, put in RC.0
1A	9C 7E	GHI, SHLC: Shift o'flow into page incr.(LSB)
1C	52	STR: Put page increment on stack (=0,1,2,or3)
1D	9B F4 BC	GHI,ADD,PHI: Add page incr. to disp. start page
0Y20	45 FA 0F	LDA,ANI: Get CHIP-10 instr. LS byte, select N
23	AD A7	PLO-RD&R7: Put N in RD.0 and R7.0
25	F8 D0 A6	LDI-D0, PLO-R6: Set VX pointer to RAM work area
28	94 AF	GHI-R4, PLO-RF: R4.1=00
2A	87 32 85	GLO,Br. if 0 to 0YD5: R7 counts up to N
2D	27	DEC R7: Since R7.0 found ≠0, above line
2E	4A BD	LDA,PHI:Put 1st word to be written into RD.1
30	9E AE	GHI,PLO: Put LS 3 bits of VX into RE.0 (&inRE.1)
32	8E 32 3E	GLO,Br.if0: RE.0 counts SHR's of word to be writ.
35	9D F6 BD	GHI,SHR,PHI: RD.1 has shifted MS part of word
38	8F 76 AF	GLO,SHRC,PLO: RF.0 has LS part of shifted word
3B	2E 30 32	DEC,Br.: Decrement shift count, loop
3E	9D 56 16	GHI,STR,INC: Write MS part in RAM buff. @0YD0
41	8F 56 16	GLO,STR,INC: Write LS part of image word in next RAM buffer addr.
0Y44	30 28	Br.: Loop, get next word of N, repeat

Note: Code from 0Y00 to 0Y1F and 0Y71 to 0Y7B inclusive is new or heavily revised; rest closely follows CHIP-8, with changes in branch addresses due to relocation and in a few constants.

CHIP-10 DISPLAY ROUTINE (continued)

Address	Hex Code	Mnemonics and Comments
0Y46	00	IDL: Sync;wait for display interrupt
47	EC	SET X=C · RC is display buffer pointer
48	F8 D0 A6	LDI-D0,PLO: R6 to begin. of RAM buffer
4B	94 A7	GHI,PLO: R7.0, collision marker,=0
4D	8D 32 7E	GLO,BR.IF 0: RD.0 is N,#of words displayed
50	06 F2	LDN,AND: "AND" new word with one already displayed
52	2D 32 58	DEC,BR.IF 0: If no common bits, leave R7.0=0
55	F8 01 A7	LDI,PLO: If 1 or more bits match, setR7.0=1
58	46 F3 5C	LDA,XOR,STR: WRITE NEW WORD (XOR w/old)
5B	02 FB 0F	LDN,XRI-0F: Was word at end of 16-word line?
5E	32 6C	BR.IF 0: If so, skip writing LS part in next wd.
60	1C	INC: If not, incr. display address pointer
61	06 F2 32 68	LDN,AND,BR.IF 0: "AND" new LS part w/old word
65	F8 01 A7	LDI,PLO: Set R7.0=1; 1 or more bits match
68	06 F3 5C	LDN,XOR,STR: WRITE NEW WORD,LS part (XOR w/old)
6B	2C	DEC: Move RC back to MS-part-word
6C	16	INC: Move R6 to MS part of next image word
6D	8C FC 10 AC	GLO,ADI,PLO: Incr. addr. by 16 (to word below)
71	9C 7C 00 BC	GHI,ADCI,PHI: Carry into MS address byte
75	E2 52	SET X=2,STR page on stack
77	9B FC 04	GHI,ADI-04: D=limit=last disp.buff.page+1
7A	F3 EC 3A 4D	XOR,SEX,BNZ:Loop if <u>not</u> beyond last page; else fall through to return
0Y7E	F8 FF A6	LDI,PLO: Set R6 to 0YFF, VF storage
81	87 56	GLO,STR: Set VF=R7.0 (1 if new image touched old)
83	12	INC: Increment the stack pointer
0Y84	D4	SEP: Return to "Call" subroutine
0Y85	8D A7	GLO,PLO: R7 was 0 to get here; reset to N,#words
87	87 32 46	GLO,BR.IF 0: If R7=0, done with I pointer reset
8A	2A 27	DEC RA,R7: RA is I pointer, R7 loop counter
0Y8C	30 87	BR: Loop to continue I-pointer reset

CHIP-8 on the ELF-II !

Rick Simpson:

I received a package of the first five issues of VIPER and still can't calm down enough to thank you properly. This is the best collection of ideas and software that has been published. on the 1802 (the greatest micro on the market for doing things.)

Maybe I should explain to you that I am not a VIP owner: I have owned a NETRONICS ELF-II (1802) for about a year & have found very few articles written relating to the 1802 with the exception of Popular Electronics. I purchased the VIP manual about 6 months ago hoping it would have some software for me to use. Much to my surprise, it contained something even better; a language called CHIP-8 and 20 games I could run. There was only one problem, the operating system was designed for ROM & in a different location than my available 4K of RAM. Well, I figured if the code was there in print, it couldn't be too hard to rewrite to work on my ELF-11. About two months later after hand decoding each instruction in the operating system & designing a keyboard to work like the VIP, I solved it. I had my VIP operating system up & running in the last three pages of RAM. Now, only one obstacle remained; to get the interpreter working. Another couple of months & some patching and I finally was able to make an 8 move around on the screen with a CHIP-8 program. I was elated, and immediately put a game in & ran it. The game worked perfect, so I put the rest of them on cassette & started experimenting with CHIP-8. Later, I added a 555 timer type circuit to the Q line so I would get tones with each keystroke. Now I can enter programs with my keyboard (made from a casio calculator) and just listen to the tones & know my program is going to the right little niche.

My next project is to add an expansion connector compatible with the VIP so I can use the VIP add-ons (Tiny BASIC in ROM, in particular).

How about some information on the 1804, and if RCA starts to manufacture the 8085, will peripheral circuits be designed to interface with both the 8085 and 1802?

Keep up the good work and you'll probably be hearing from me again if you don't mind talking to a non-VIP owner.

Sincerely,

Bobby R. Lewis
Waterford, CT

Editor's Note:
As soon as I received this letter I wrote back to Bobby asking for details on his conversion. We just received his information as this issue went to press. His article will appear in VIPER #8.
As for the 1804, RCA still hasn't announced a firm availability date. The 8085 is still further down the line.

VIP BOOTSTRAP LOADER FOR NETRONICS ELF II CASSETTES

by Dave Friedman

This short program lets you load Elf II cassettes into the VIP for re-recording on another cassette in VIP format. Load the bootstrap program into a higher memory page than the highest page you want to load from cassette. Put the C0 HH 00 long branch to the loader at 0000. Reset and run.

Start the tape with the tone control at high. Adjust the volume for a solid glow on the tape light. Then, turn the tone control down until the tape light starts to flicker. Rewind the tape and start it again. Press and release hex key 'B'.

The Elf II program will be laoded into RAM from 0000 for as many pages (PP) as are specified by byte HH17. If there are more pages specified than are on the tape, the routine will not finish so you'll have to watch for the tape light to go out. If there are enought pages of data on the cassette, the routine will fall through to a solid alert tone. Reset and run with key 'C' for the operating system only!! Key in the op-system stack page (0F for 4K or more) and BF. Make note of the byte at 0FBF, then press 'A' for the read mode and advance to 0FCF. The number of errors during loading is the two-byte number 0FCF.0FBF. Hopefully, this will be 0000 or zero. If not, you may need to adjust volume and tone levels but it shouldn't be more than about 0030 if you're anywhere near the correct settings. (Note: The error count will be clobbered by a second reset/run so look at the errors first before looking at the loaded program.)

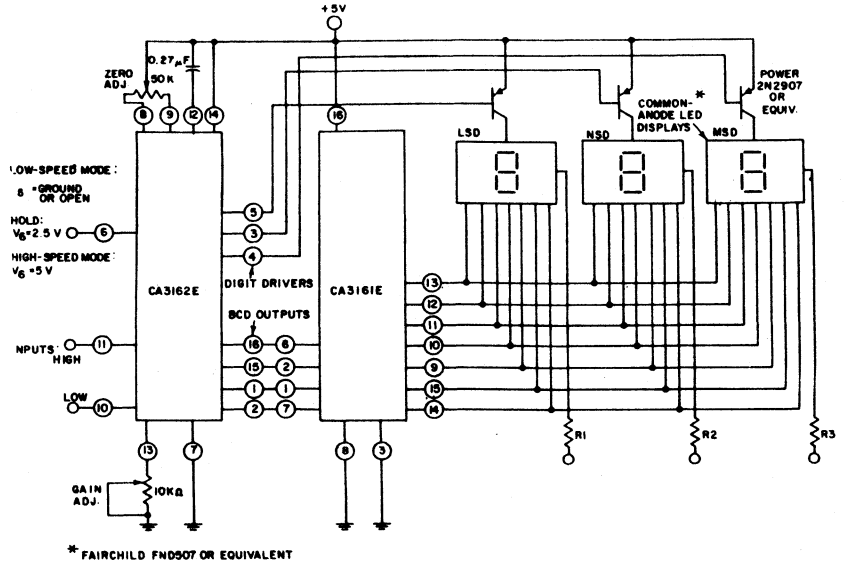
When you've got a zero error load, save it on another cassette in VIP format for easier reloading. Don't forget to replace the long branch at 0000 before trying another load. It is clobbered by the program loaded from cassette using the bootstrap loader.

This loader could be made more convenient to use by adding TV display etc. but then it's used very infrequently!!

```
0000   C0 HH 00          Long branch to loader
-----------------------------------------------
HH00   E0 62 0B          Latch 'B'
  03   3E 03 3703       Wait for keypress
  05   7B 36 05 7A       'Q' on till key released
  09   90 B3 BC          R3 to be PC/RC to be sub. counter
  0C   80 A3 13 D3       R3=HH0D,HH0E,HH0F,HH10/P.C.
  10   F8 00 B6 A6 BF AF R6=0000(Start load)/RF=0000(errors)
  16   F8 PP AE          RE.0=PP Pages
  19   F8 4A AC          RC=HH4A(Cycle read subroutine)
  1C   F8 10 B9          R9=Leader delay
```

```
1F   DC                  Get a bit to DF
20   33 1C               If a '1', reset delay,else....
22   29 99 3A 1F         keep looping until end delay
26   DC 3B 26            Look for a '1' start bit
29   F8 09 A9 A7         R9=09(count)/R7.0=09(odd for parity)
2D   97 7E B7            Save bit in R7.1
30   29                  Decrement the count
31   DC                  Get next bit
32   89 3A 2D            If count 00, parity bit falls through
35   87 F6 33 3A 1F      If parity was even, DF=1(no error)
3A   97 FB FF            Complement the Byte
3D   56 16               Store and increment RAM address
3F   86 3A 26            If not next page, get next start bit
42   2E 8E 3A 26         If not end pages, get next start bit
46   7B 30 46            Stop with Q on as alert signal
49   D3                  Return to caller
4A   F8 0E               'D'=0E>½ cycle of '0',<½ cycle of '1'
4C   3D 4C               Wait for signal zero/minus
4E   3D 57               If not still minus, it's a '0'
50   FF 01               'D' - 1
52   3A 4E               If 'D'=0, it's a '1'
54   17                  Incr. parity count;even ends odd!!
55   F8 80               Load '1' in bit 7 for shift to DF
57   FE                  80 from '1' or 01-0E from '0'
58   35 58               Wait for start of positive ½ cycle
5A   30 49               Branch to 1 byte before entry
```

Three-Digit DVM using RCA chips



A Letter to The VIPER:

Dear Rick:

Sorry to take so long getting this back to you, but I was hoping to have a short article for you. My system has been down while I was working on a cabinet for it, and I haven't been doing any hardware or software development for the past several weeks.

I am presently using an A/D converter from Alpha Electronics (Box 1005, Merrit Island, Florida, 32952). The converter was written up in the September 1978 Popular Electronics. It uses a National Semiconductor 3511 A/D Converter. The board has a 4066 analog switch in front of the converter to provide four computer selectable inputs and runs on a single 5 volt supply.

The reason I was hoping to write an article about it for you, is that there are some problems with the board (at least in my opinion). The 5 volt supply is used for the converters reference directly, with no stabilizer of any type. I consider this a very poor design, as the converter chip is a 3-1/2 digit model. It is also set up so that the computer has to sit in a loop and wait for data, in spite of the fact that the chip itself is designed for microprocessor applications, and provides an end-of-conversion signal, and has addressable storage for each of the 3-1/2 digits. Finally, the board only allows converting positive voltages, in spite of the fact that the chip itself can do negative conversions as long as the ground is allowed to float in relation to the voltage it is measuring.

If you are willing to wait a few weeks (my cabinet is to the painting stage--almost done, I am working on a new board with wat I consider a more realistic implementation of the chip, and would be happy to do an article on the hardware and software implementations of both converter boards.

I don't want to give the impression that the Alpha Electronics kit is a bad deal. I learned a lot about the chip using their board. I just feel that for my own applications (I am doing some temperature monitoring) the chip can be more useful in a different configuration.

A few questions now:

1. Do you know if anyone is planning a number cruncher board for the VIP using the new RCA multiply/divide chip? Also, do you know the pricing on the chip, and if RCA (or anyone else) has any software for it.

2. Have you any idea about how much would be involved in interfacing the number cruncher board from Quest electronics to a VIP. It is supposed to be for any 1802 system, but they don't say anything about the bus they use (it's in the latest issue of Radio-Electronics).

3. Are there any plans in the wings to follow up the Tiny BASIC in ROM with a more advanced BASIC (or perhaps a more structured language).

If so, would the implementation use a number cruncher chip, and would it (hopefully) be available in ROM.

I hope the questions are not to much. I really intended this letter to be a simple reply to your questions about my A/D, but I'm afraid I've been bitten pretty badly by the VIPER, and am on my way to a much larger system than I originally intended the VIP to turn into.

Sincerely,

Larry Rowland

Thanks for the info, Larry. We'll look forward to your article since A/D conversion is high on the interest list of our readers.

The RCA M/D chip has not yet been released and we have no price availability information.

Do any of our readers have experience with the Quest board that they would be willing to share?

Several enhancements are planned for Tiny BASIC. RCA won't say anything more on future languages at this time.

R.S.S.

Dear Sirs:

Please enter the following items for sale in your newsletter:

Netronic ELF II with all edge connectors
Giant Board
4K RAM Board
5V 1A wall Mount Power Supply
Tiny Basic on ELF Cassette
Tiny Basic on Paper Tape (Itty Bitty computers)
Pittman's Short Course for 1802
All back issues of Ipso Facto - 1802 Newsletter and More!
All boards assembled and fully socketed 4K RAM not tested - needs larger power supply.
Asking $200 but will negotiate.

Contact Bill Grzanich, 400 Mill Rd. Apt. 30
Addison, IL 60101 (312) 543-0685

LABEL	ADDR.	INSTR.	COMMENTS
	26Ø	A35C	Point to dash
	2	6DØØ	Display X to left end
	4	6EØØ	Display Y to top
DASH LOOP	6	DDE3	Show dash
	8	7EØ7	Move Y down
	A	3E23	Stop at 5th dash
	C	1266	Do next dash down
	E	7DØ6	Move X right
	27Ø	3D3C	Stop at 1Øth dash
	2	1264	Do next column of dashes
	4	A42Ø	ANS (answer) location
	6	6CØØ	ANS array index
RANDOM # LOOP	8	CØØ7	Random # Ø through 7
	A	FØ55	Put # in ANS array
	C	7CØ1	Incr. index
	E	3CØ5	Stop on 5th random #
	28Ø	1278	Get another #
	2	6DØØ	Display X to left end
	4	6EØØ	Display Y to top
	6	6CØØ	1Ø try loop counter
TRY LOOP	8	A42Ø	ANS location
	A	F465	get five # from ANS
	C	F455	Put in working COPY of ANS
	E	6BØØ	Counter for 5 inputs
NEW KEY	29Ø	FØØA	Key → VØ
	2	3ØØF	Test for cancel signal - key F
	4	12AE	Bypass input CANCEL
CANCEL LOOP	6	4BØØ	End of input cancel?
	8	129Ø	Exit - all done
	A	7EF9	Back up display Y
	C	A42A	IN (input) location
	E	7BFF	Decrement loop counter

Dear VIPers: Here is 5-row Mastermind. Complete info was in VIPER #6

LABEL	ADDR.	INSTR.	COMMENTS
	2AØ	FB1E	Add array offset
	2	FØ65	Recover previous input
	4	FØ29	
	6	DDE5	Erase previous input
	8	A35C	dash location
	A	DDE3	Return dash to display
	C	1296	Continue cancel loop
SAVE INPUT	E	6AF8	-8
	2BØ	8AØ4	Add to key value
	2	3FØØ	Test for key < 8
	4	129Ø	Key > 7 → try again
	6	A35C	Dash location
	8	DDE3	Erase dash
	A	A42A	IN location
	C	FB1E	Loop count is IN array offset
	E	FØ55	Input key to IN array
	2CØ	FØ29	
	2	DDE5	Show input value
	4	7EØ7	Move down
	6	7BØ1	Incr. input loop count
	8	3BØ5	Test for 5 inputs
	A	129Ø	Get another input
	C	64ØØ	V4 signal for dotted bar
	E	68ØØ	V8 correct count for this try
	2DØ	6BØØ	"Test for correct" loop counter
DOTTED LOOP	2	A42A	IN location
	4	FB1E	Loop count is array offset
	6	FØ65	Get next IN #
	8	82ØØ	Move to V2
	A	A425	COPY location
	C	FB1E	array offset
	E	FØ65	Get COPY #

LABEL	ADDR.	INSTR.	COMMENTS
	2EØ	83ØØ	COPY # → V3
	2	236Ø	Call Bar subroutine
	4	A42A	IN location
	6	FB1E	array offset
	8	8Ø2Ø	
	A	FØ55	Save (possibly modified) IN#
	C	A425	COPY location
	E	FB1E	array offset
	2FØ	8Ø3Ø	
	2	FØ55	Save (possibly modified) COPY#
	4	7BØ1	Incr. loop count
	6	3BØ5	Are all 5 rows tested
	8	12D2	continue testing
	A	38Ø5	If all correct – end of game
	C	1322	Branch to white bar test
SHOW ANSWER	E	6EØØ	Display to top – Show Answer
	3ØØ	6D3C	Display to right end
	2	69ØØ	Answer loop count
ANS LOOP	4	A42Ø	ANS location
	6	F91E	array offset
	8	FØ65	Get next ANS #
	A	FØ29	
	C	DDE5	Show ANS #
	E	7EØ7	Move down
	31Ø	79Ø1	Incr. loop count
	2	39Ø5	Stop at 5 numbers
	4	13Ø4	Get more #
	6	6Ø2Ø	Blink time
	8	FØ15	Set timer
	A	FØØ7	Test timer
	C	3ØØØ	" "
	E	131A	" "

LABEL	ADDR.	INSTR.	COMMENTS
	32Ø	12FE	End of blink answer loop
WHITE BAR TEST	2	64Ø1	V4 white bar signal
	4	6AØØ	IN loop index - outer loop
WHITE BAR LOOP	6	6BØØ	COPY loop index - inner loop
	8	A42A	IN location
	A	FA1E	array offset
	C	FØ65	Get IN#
	E	82ØØ	
INNER LOOP	33Ø	A425	COPY location
	2	FB1E	array offset
	4	FØ65	Get COPY #
	6	83ØØ	
	8	236Ø	Call Bar subroutine
	A	A425	Copy location
	C	FB1E	array offset
	E	8Ø3Ø	
	34Ø	FØ55	Save (possibly modified) COPY#
	2	7BØ1	Incr. COPY loop index
	4	3BØ5	Test for 5 iterations
	6	133Ø	continue loop
	8	7AØ1	Incr. IN loop index
	A	3AØ5	Test for 5 iterations
	C	1326	continue loop
END OF TRY	E	7DØ6	Move to next column
	35Ø	6EØØ	Move to top row
	2	7CØ1	Incr. TRY loop
	4	3CØA	Is this the 1Øth TRY?
	6	1288	Process next try
	8	12FE	Done - Go to show answer
	A	9ØFØ	Dotted bar / white bar
	C	ØØØØ	Dash
	E	6ØØØ	"

LABEL	ADDR.	INSTR.	COMMENTS
BAR SR	36Ø	523Ø	V2 = V3 ?
	2	1374	NO - Just Return
	4	A35A	Dotted bar location
	6	34ØØ	Ø in V4 means use dotted bar
	8	A35B	White bar location
	A	DDE1	Show bar
	C	7EØ3	Move down
	E	78Ø1	Incr. correct count
	37Ø	62ØE	Cross out IN #
	2	63ØF	Cross out COPY #
	4	ØØEE	Return
Storage	Arrays		
	42Ø	FIVE LOCATIONS	ANS — answer
	425	FIVE LOCATIONS	COPY - working copy of answer
	42A	FIVE LOCATIONS	IN - column input

CORRECTION
to the
CORRECTION

The CHIP-8I code in issue #3 was wrong. So was the correction printed in issue #5. The slip we inserted in most copies of issue #3 was correct and is as follows:

Starting at 01A4:
86 FA 01 3A AC E5 63 D4 E7 45
FA 01 3A F2 63 D4

Starting at 01F2:
3F F2 6B 3F F5 D4

YES! I'd like to subscribe to the VIPER and receive all ten issues of this year's volume! I enclose $15.00 in full payment.

Name _____
(Please print or type)

Address _____

City _____ State _____ Zip _____

Cash, Check, Money Order Enclosed _____

MC/VISA/BAC Number _____ Exp. Date _____

MC Interbank No. _____

Required Credit Card Signature _____

☐ You may let other VIP owners in my area know I have a VIP, so they can contact me.

☐ I am interested in forming/joining (circle one) a VIP Users group

☐ I'd like to see articles in the VIPER about:

_____
_____
_____
_____

MAIL TO: VIPER; P.O. Box 43; Audubon, PA; 19407

VIPER
P.O. Box 43
Audubon, PA 19407

Place Stamp Here

VIPER

P.O. Box 43

Audubon, PA 19407

TO: