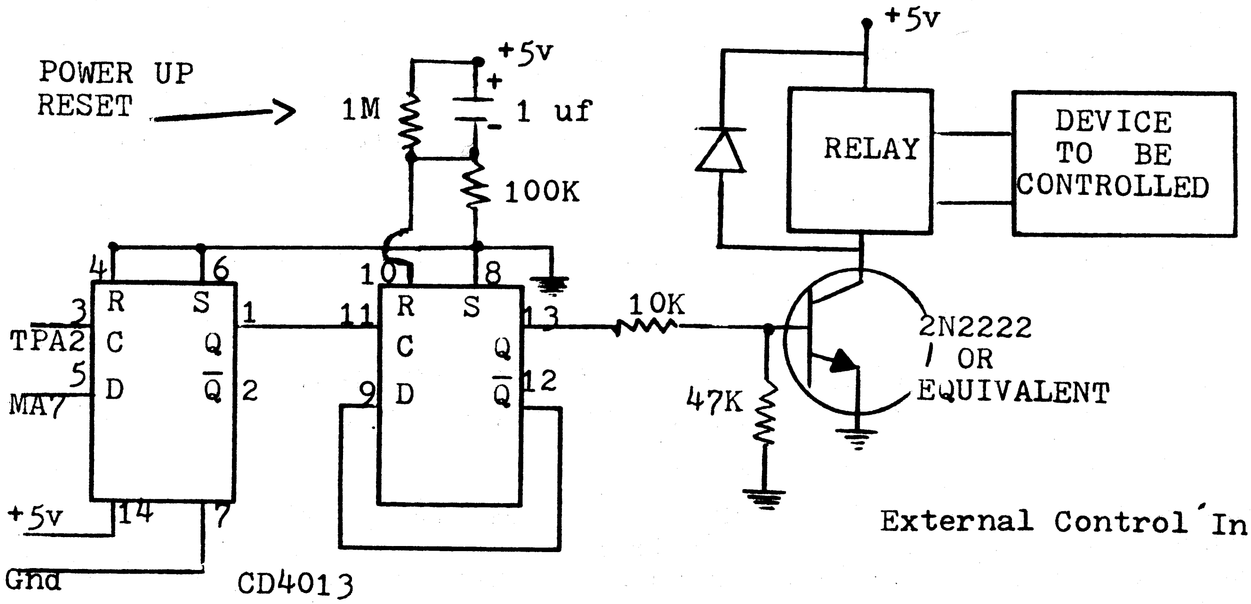
VIPER

VOLUME 1 3/79 ISSUE 8



External Control Interface

SIGN UP NOW FOR VOLUME 2!

$2.00

AN ARESCO PUBLICATION

EDITORIAL

I've seen more and more indications lately that the predicted microcomputer-in-business revolution has been delayed due to lack of interest - acutally, lack of software. RCA seems to be one of the few micro-makers who continues to believe people buy small computers "just for the fun of it".

Good articles continue to come in at a gratifying rate, but we have yet to receive a single article which uses the color, SuperSound, or Simple Sound boards. We would love to publish your SuperSound symphony, or a revision of any VIP game which incorporates color or sound or both. What have you done?

Two important events are coming up for the VIPER, One, our move in APRIL to Columbia, MD; and two, the upcoming renewal campaign for Volume 2 of the VIPER. Details on both can be found elsewhere in this issue.

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).

Readers are encouraged to submit articles of general interest to VIP owners. Material submitted will be considered free of copyright restrictions and should be submitted by the 1st day of the month in which publication is desired. Non-profit organizations (i.e., computer clubs) may reprint any part of the VIPER without express permission, provided appropriate credit is given with the reprint. Any other persons or organization should contact the editor for permission to reprint VIPER material.

Advertising rates are as follows:

1/4 page – 25.	3/4 page – 65.
1/2 page – 45.	full page – 85.

Less than 30% of the VIPER will be available for advertising. Please send camera ready copy in the exact page size of your ad on 8-1/2 x 11 white stock by the 1st day of the month in which you'd like the ad to appear. Photos should be glossy black & white in the exact size to be printed. Payment required with copy.

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

PROGRAMMING THE STUDIO II

When you write your own programs, you must be careful not to conflict with certain registers used by the interrupt routine, which is on the PROM card at addresses 04A8 through 04E1. The restrictions are:

1. R0 is the display pointer.
2. R1 is the interrupt routine pointer.
3. R2 is a stack (storage) pointer, and uses the last four locations in the page 8 RAM (08FC-08FF)
4. R8 is decremented once in each interrupt routine execution (once each 1/60th of a second) so that the main program can use it for timing purposes (delays).
5. RB is transferred to R0 before the display starts, during the interrupt routine. By setting RB in the main program, you can force the display to start anywhere in memory. RB is normally set to 0900.
6. R4.0 contains the last digit (4 bits) latched by the decoder (4515) and is used by the interrupt routine to reset the decoder to that value before control is returned to the main program.
7. Your program starts with R9 as the program counter. It's a good idea to change it immediately to R5 so you can use the PROM subroutines.

The interrupt routine does three things:

1. Sets the starting location in memory for the display and repeats each TV line 4 times.
2. Decrements R8 for delay purposes
3. Checks key 0 on the B keyboard and restarts the main PROM program if key 0 has been pressed.

You may write your own interrupt routine if you want, but it must incorporate the "key 0" feature (item 3) so you can return from your program back to the start of the PROM keyboard by pressing key 0 on the B keyboard.

The sample display program in Figure 5 shows what can be done with the system using only a very short program. It can be used to check out your card. It produces constantly changing video patterns on the TV screen, which grow from the top and bottom and meet in the center of the screen.

Studio II can also be used to provide control of an external device which can be controlled in sync with the TV display. A possible interface is shown in Figure 6. Executing any instruction which reads hex address 8XXX (where X can be any hex digit) will alternately turn the relay on and off.

Figure 3 (published last month) shows the Studio II connector pinout so you can do your own interfacing. MA0-MA7 is the address bus, B0-B7 is the data bus. a high (+5V) in RAM DISA disables all the Studio II game ROMs (IC13 and IC14 in Figure 1, published earlier). $\overline{\text{MRD}}$ and TPA are explained in the 1802 User's Manual.

If you're ambitious, you might want to trace through the PROM program with the idea of improving or expanding it. You could add another shift mode, to allow shifting your program either forward or backward. By the way, when using the shift mode, you must be careful to change any branch instructions that cause a jump from outside the shifted block to inside the shifted block, to reflect the new address of the bytes in the shifted block.

The first instruction on the PROM is a two-byte interpreter instruction (0402) which says "Execute the machine language routine at address 0402". The machine language routine is the entire rest of the PROM.

Addresses 0402-041D are used for register initialization, including setting up R1 to point to the PROM's interrupt routine and setting up R5 as the main program counter. All subroutines are on page 5, so once R3.1 is set to 05, the various subroutines can be called by specifying only R3.0.

KYSC1 is called to check the keyboard for the desired mode, then KYSC2 is called to check for the first two digits of the address. ALPNUM and CONV2 are called to display these digits on the TV. Then all this is repeated for the next two address digits.

The program then branches to the proper section of code, depending on which mode was selected. Load mode starts at 0474, step mode at 0446, run mode at 0445, and shift mode at 048A. Locations 04A2 through 04A7 are unused, and the interpreter goes from 04A8 to 04E1, with 04E9 through 04FF unused. Addresses 0500 through 0542 are used for the symbol table (byte patterns for the 16 hex digits, needed to display them on the TV). Addresses 0543 through 056E and 05F2 through 05FD are used for the register storage routine which is entered whenever key 0 on the B keyboard is pressed. The rest of page 5 is used for the subroutines, with entry points shown in the list of subroutines.

Other unused locations are marked with XX in the program listing in Figure 4. I never had a chance to optimize the program, so it should be possible to find even more free memory space by attempting to rewrite it. Deleting the shift mode would open up 24 more bytes, for example.

Write Your Own Programs

Although you have only a small amount of RAM for your programs, you can still write fairly complex programs by making use of some of the subroutines resident on the PROM card. These subroutines are used mainly to simplify keyboard input and numeric output to the TV. Before using them, you must set the program counter of your main program to R5 (1802 register 5), and and remember that these subroutines use R3 as their program counter.

Load the register to be used as the subroutine program counter (R3 in this case) with the entry point of the routine, and then do a Dn instruction, which sets the program counter to Rn (R3 in this case, using a D3 instruction). These subroutines end with a D5 instruction, which resets the program counter back to R5; which causes a return back to the main program at the point just after the D3 instruction which originally called the subroutine.

The PROM card subroutines are:

KYSC2 Entry at 0570

Scans the keyboard, looking for a 1 byte input (two digits). Returns with MSD in R6.1 and LSD in R4.1. These registers contain the input digit in the form 0X, where X is the input digit. Returns with R3 pointing to entry for the DISP1 subroutine. Adds 09 to the B keyboard entries so that 1, 2, 3, 4, 5, and 6 are converted to A, B, C, D, E, and F, respectively.

DISP1 Entry at 0598

Displays 2 hex digits (one byte). The MSD is taken from R6.1 and the LSD is taken from R4.1. The digits are displayed at the RAM location pointed to by RD, which points to the top left corner byte. Each digit is 6 bytes high in the display. Returns with R3 pointing to the CONV2 subroutine.

KYSC1 Entry at 0574

Scans the keyboard for 1 digit and puts it in R4.1 in the form 0X, where X is the input digit. Returns with R3 pointing to DISP1. Adds 09 to B keyboard entries to perform conversion as in KYSC2.

CONV1 Entry at 05D6

Takes the byte in R6.0 (X_1X_2) and converts it to a form suitable for display by DISP1: $0X_1$ in R6.1 and $0X_2$ in R4.1. Returns with R3 pointing to DISP1.

CONV2 Entry at 05E4

Takes the byte in R6.1 ($0X_1$) and the byte in R4.1 ($0X_2$) and merges them into R6.0 (X_1X_2). Returns with R3 pointing to KYSC2.

SAMPLE DISPLAY PROGRAM

ADDRESS	DATA (HEX)	COMMENTS
08C8	F8 08 B5 B3	R3.1=8
08CC	F8 D0 A5 D5	R5=Program counter =08D0
08D0	F8 F0 A3	R3.0=F0
08D3	87 AA A6 B6	R7.0 → RA.0, R6.0, R6.1 (see note)
08D7	F8 09 BA AC	RA.1=RC.1=9
08DB	F8 FF AC	RC.0=FF
08DE	06 5C	M(R6) → M(RC)
08E0	D3 FF	Call Delay Subr. at 08F0:Delay=FF
08E2	08 5A 1A	M(R8) → D, D → M(RA), RA+1
08E5	2C 8A	RC-1, RA.0 → D
08E7	3A DE	Go to 08DE if D ≠ 0
08E9	16 18 D3 FF	R6=R6+1:R8=R8+1:Call delay Subr.
08ED	30 D7	Go to 08D7
08EF	D5	Return to main program (set PC to R5
08F0	45	
08F1	FF 01	Delay Subroutine
08F3	3A F1	(Program counter = R3)
08F5	30 EF	

Notes: R7.0 is set to zero by the PROM program.
R8 is incremented once every 1/60th of a second.

Figure 5

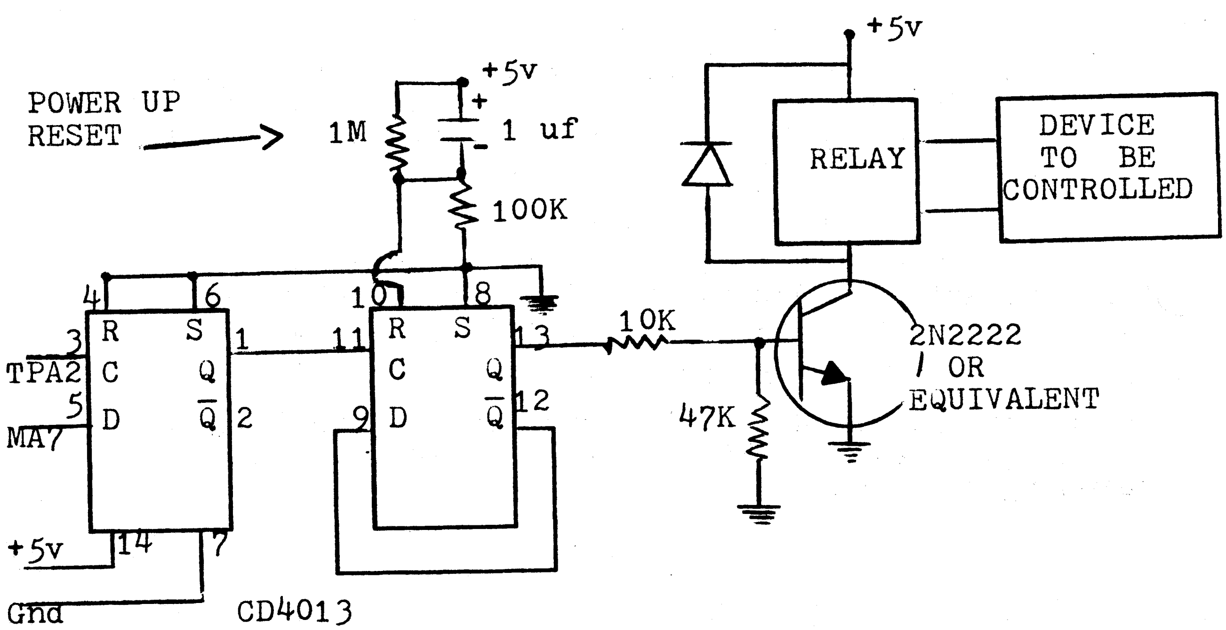


Figure 6 External Control Interface

MODIFY THE ELF-II TO RUN CHIP-8 AND VIP GAMES

by Bobby R. Lewis

(Editor's note: We have received requests from literally dozens of VIPER readers requesting information on how they can use their ELF-IIs to run VIP games in CHIP-8. Since the VIPER is dedicated to users of the VIP rather than to users of the ELF-II, we haven't exactly been what one would call responsive to the inquiries. One reader wrote: "Most of the more elaborate games, etc. seem to have been written in CHIP-8, and at my present level of skill, I'm not up to decoding and re-writing RCA's VIP monitor and CHIP-8 to run on my Super Elf. The manual that came with my machine is confined to very elementary stuff - besides that, it's terrible. They don't know a thing about communication. Help us, VIPER people! We need a "liberator"! " This particular reader reflects the same aura of "desperation" we've seen in several dozen similar letters, and we've been scratching our heads about how such an article would fit into the "exclusively VIP" format we've adopted.

A careful look at Bobby Lewis's article convinced us that the information will be useful to many of our subscribers, even if the idea is to use an ELF II rather than a VIP to start with. However, once the modifications have been made, the user will have an "effective" VIP - and will be among the exclusive membership of the VIPER subscribers! That, to us, was enough of an excuse to justify inclusion of the article in this issue. So here it is - enjoy!)

If you own an ELF-II, you can take advantage of CHIP-8 and RCA VIP games. Although the following information assumes that you have an ELF-II with Giant Board Monitor and at least 4K of RAM, you can possibly use it on other systems if you are familiar with the VIP operations and hardware.

First, let's take a look at some of the VIP features. A basic system contains a 512 byte (2 page) operating system in ROM, addressed at 8000 through 81FF. The operating system normally searches for and uses the highest page of RAM for the operating system display page. The VIP manual[1] contains a hexadecimal listing for the 512 byte CHIP-8 interpreter that must be entered into addresses 0000 through 01FF with the hex keypad. The VIP has no 7-segment displays for data; instead, it uses the video screen in conjunction with the operating system to display addresses and data and to allow modification of the data. The keyboard on the VIP is what I like to call a "dynamic" keyboard. It is perfectly suited to interactive programs and games that require keyboard responses. This is a very simple design to implement on the ELF=II, and full details are included in this article.

Before you start thinking about using the original ELF-II key board, you must be aware of the fact that it latches up 8 bits on the data bus when INPUT is pressed. In addition, the low (or first) digit pressed is shifted when you enter a subsequent digit. Even if you write another routine to read the ELF-II keyboard, you will be pressing three keys to do the function of one on a dynamic-type keyboard. This situation won't allow you to take full advantage of the CHIP-8 games.

Before I discuss the actual conversion, refer to figure 1 for a summary of ELF-II and VIP I/O instructions. The 64 instruction is really doing the same function on both machines, but with the ELF-II, you have the additional feature of displaying the contents of memory on the 7-segment displays. The other mahor difference between the I/O instructions is the hex keypad enables. The ELF-II uses the 6C, which is an input instruction, and the VIP uses a 62, which is an output instruction. This should give you a clue as to why a different keyboard is needed to run CHIP-8.

ELF-II	VIP
61 video off	61 video off
62 available	62 output to keyboard
63 available	63 output port
64 display LEDs	64 Mx-bus, Rx 1
65 available	65 available
66 available	66 available
67 output port	67 available
68 illegal	68 illegal
69 video on	69 video on
6A available	6A available
6B available	6B input port
6C input from keyboard	6C available
6D available	6D available
6E available	6E available
6F input port	6F available

FIGURE 1

I/O INSTRUCTION SUMMARY

The operating system in the VIP actually outputs the low 4 bits of the data bus to a (4 to 16 line decoder) attached to one side of the hex keypad, allowing EF3 to be enabled corresponding to the key being pressed. So actually, the VIP keypad only inputs EF3, not data. The ELF-II, on the other hand, latches up 8 bits, 4 bits at a time, and inputs this information to the data bus when the input switch (EF4) is pressed. Figure 2 contains a summary of the EF flag usage for both systems.

ELF-II	
EF1 video interface	EF1 video interface
EF2 cassette interface	EF2 cassette interface
EF3 available	EF3 hex keypad
EF4 hex keypad	EF4 available

FIGURE 2 - EF FLAG USAGE

HARDWARE MODIFICATIONS

Refer to figure 3 for details to implement a VIP type dynamic keypad to your ELF-II. The keyboard circuit can be hooked directly to the data bus, bit it is easier to use the existing output port. The numbers in () are actual pin numbers at the output port on the Giant Board. Although a CD4514 should be used, I used the TTL equivalent (74154) with no problems. Since pins 6, 7, and 8 are not presently used on the output port, you can jumper +5V, ground, and EF3 or EF4 to them from anywhere on the Giant Board. Use EF4, if not already in use, and you already have a diode for use at the hex/term switch location. These are the only hardware modifications required to your ELF-II

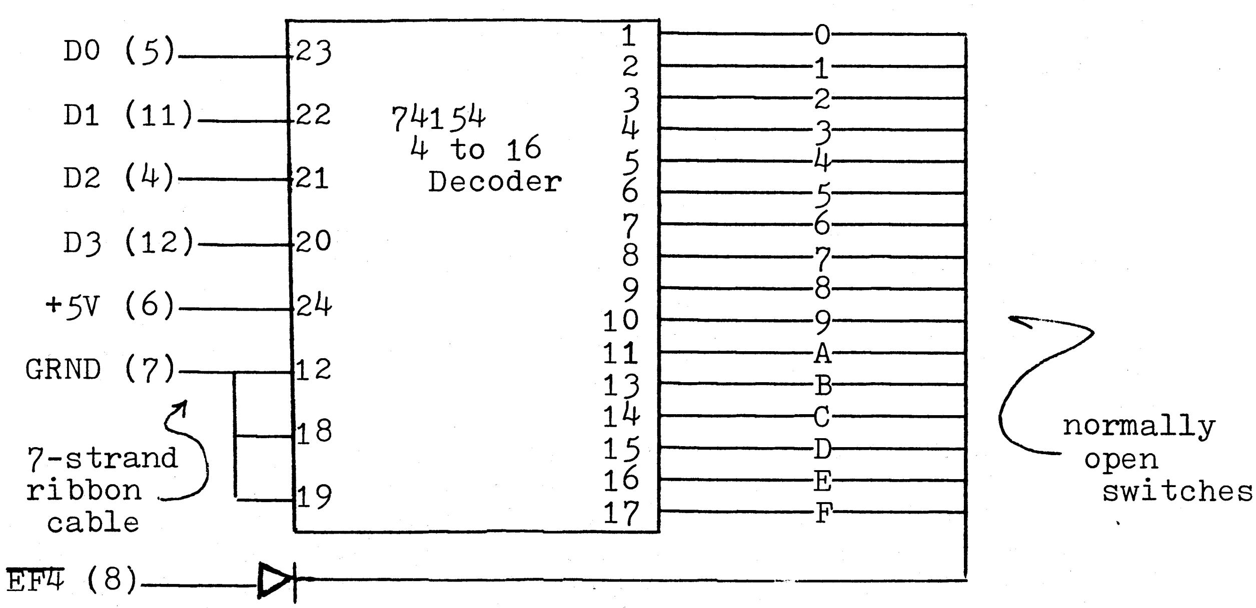


FIGURE 3 - HEX KEYBOARD

(This article will be continued next month, in issue #9. Don't go 'way! - Terry)

GRAPHIC LUNAR LANDAR

by Udo Pernisz

The Graphic Lunar Lander has the following features:

1. Real-time lander simulation, with a time interval of one second used as the increment for integrating the lander's equation of motion (i.e., cycle time is one second). A "beep" audibly counts seconds.

2. Realistic data for both lander and moon: 1.67 m/sec^2 for the moon's gravitational acceleration; 1.5 metric tons for the lander's mass, including 150 kg of fuel (which is low enough to permit taking the lander's masss as constant); 1500 m/sec exit velocity for the hot gases at the jet nozzle (which corresponds to approximately 2000 degrees K of gas temperature in the burner chamber).

3. Initial values of:

Height	2095 m
Velocity	147 m/sec
Fuel	447 units (taken as equivalent to 150 kg)

   Range of fuel rate is selectable from keypad (0-F), with zero lander acceleration by pressing key 5. Keyed in settings are self-repeating until a new key is pressed.

4. Data display: Remaining fuel, velocity, height above ground (all in decimal numbers up to 4 digits); range of velocity is -255 through +255 m/sec; range of height is 0 to 7167 (decimal) meters (expandable).

5. Graphic display: Lander picture switches from "far" (one picture element) to "near" (6 x 8 picture) on crossing a height of 256 meters while descending. It switches back while ascending through 256 meters. The height resolution is 128 m for the "far" picture, using a rounding routine, and 8 m for the "near" picture.

In addition to the lander itself, a moon surface is displayed with a landing site indicated and an optional scal of height appears on the left edge of the display area.

Messages are for "touch-down" and "crash", with illustrative graphics. In case a player uses an excessive amount of fuel, an "overflow" message is displayed if the velocity range is exceeded.

6. The Graphic Lunar Lander runs in the standard CHIP-8 Interpreter, and only uses 3 pages of RAM (0200 - 04FF). The range of displayable height values can be extended by using all or part of page 5, although settings as described don't need it. It may be desired (for training purposes!), in which case, you might want to change the initial velocity and fuel settings. Other trainer options include a larger cycle time and a different touch-down velocity window.

Operation:

After switching from RESET to RUN, the height scale and lunar surface are displayed together, with the initial settings of fuel, velocity, and height. The lander appears at the top of the screen. The simulation is started by pressing any key on the keypad. The value of the pressed key is taken as the "burn" rate - the number of fuel units used per second. A value entered will be used as long as another key is not pressed.

A standard simulation ends with either a "crash" or a "touch-down", and then a new game can be started by pressing key 0.

Subroutine Location:

Moon surface and scale:	02BE-02D5
Multiply and round	02D6-02DF
MLS - Divide by eight	02E0-02E8
Switch from "near" to "far"	02EA-02F9
Compute Fuel	0300-031F
Compute Velocity	0320-033F
Compute height; detect End-Of-Game	0340-0395
Display fuel	039C-03A9
Display velocity	03AA-03B7
Display height	03B8-03C5
Display digits	03E4-03FF

Data location:

Control: V9 to VE	02FA-02FF
Initial Fuel	0400-040F
Initial Velocity	0410-041F
Initial height	0420-042E
Lander pictures (3)	042F-0442
Text and other graphics	0447-0496
Dynamic storage of fuel, velocity, and height	0498-04C7
Hex/Decimal conversion table (incomplete)	04C8-04FF

Other interesting locations:

Minus sign for velocity	0417
Key entry scratch pad	0497

ADDRESS	CODE	COMMENTS	REMARKS
			Initialization Routine
0200	6F00	VF=0	Jump and count variable
0202	A400	I=0400	Set pointer to initial values
0204	FF1E	I=I+VF	jumping sequentially to fuel, velocity, and height

ADDRESS	CODE	COMMENTS	REMARKS
0206	FE65	V0:VE=MI	:Read data into variables for
0208	A498	I=0498	:transfer into dynamic storage
020A	FF1E	I=I+VF	:area
020C	FE55	MI=V0:VE	:Write initial values into dyna-
020E	7F10	VF=VF+10	:mic storage area and increment
0210	3F30	SKIP;VF=30	:jumper. Perform three times,
0212	1202	GOTO 0202	:for fuel, velocity, and height.
0214	00E0	ERASE	
0216	22BE	DO 02BE	:Display moon surface and scale
0218	A2F1	I=02F1	:Read control data into V9-VE
021A	FE65	V0:VE=MI	:by using dummy data for V0-V8
021C	239C	DO 039C	:Display initial fuel
021E	23AA	DO 03AA	:Display initial velocity
0220	23B8	DO 03B8	:Display initial height
0222	A42F	I=042F	
0224	DAB1	SHOW 1MI @ VA, VB	:Display lander picture, "far"
0226	F00A	V0=KEY	:Wait for a key entry to start
0228	1240	GOTO 0240	:Jump into proper place of main
			:loop
			MAIN LOOP: START
022A	FC15	TIME=VC	:Time delay to obtain 1 second
022C	F007	V0=TIME	:cycle for time interval and
			:ticking sound
022E	3001	SKIP;V0=1	:Exit delay loop with V0 set,
0230	122C	GOTO 022C	:to "beep" the sound
0232	F018	TONE=V0	:Time tone signal (exact, if no
			:key pressed)
0234	700F	V0=V0+F	:Set V0 to largest value of keys+1
0236	70FF	V0=V0+FF	:Subtract 1 from V0 in a loop
0238	40FF	SKIP; V0 .NE. FF	:Exit keypad scan if no key was
023A	13DE	GO 03DE	:pressed, and get previous value
023C	E09E	SKIP; KEY=V0	:Exit scan with pressed key in V0,
023E	1236	GOTO 0236	:Else loop back and check next key
0240	A497	I=0497	:Write value of pressed key into
0242	F055	MI=V0:V0	:memory
0244	8800	V8=V0	:Pass value of key to V8, which
			:is the general data variable
			:used for all parameters
			DISPLAY FUEL
0246	239C	DO 039C	:Display fuel data (to erase it)
0248	2300	DO 0300	:Calculate new fuel value
024A	A498	I=0498	:Set pointer to dynamic data area
024C	23E4	DO 03E4	:and write back new value, con-
			:vert hex to decimal
024E	A4A3	I=04A3	:Write LSD256 decimal digits into
0250	F233	MI=V2 (3DD)	:memory for display, using the
0252	A4A1	I=04A1	:CHIP-8 Interpreter. Then do MSD

ADDRESS	CODE	COMMENTS	REMARKS
			:but shifting by one digit
			:to overwrite the first digit
0256	239C	DO 039C	:Display fuel data
			DISPLAY VELOCITY
0258	23AA	DO 03AA	:Display velocity (to erase)
025A	8DE0	VD=VE	:Save sign bit of old value
025C	2320	DO 0320	:Calculate new velocity; new
			:sign bit in VE
025E	A4A8	I=04A8	:Write new value into memory
0260	F255	MI=V0:V2	
0262	8DE5	VD=VD-VE	:Look for change in sign of
0264	4D00	SKIP; VD .NE. 0	:velocity and set pointer to
0266	1270	GOTO 0270	:"minus sign" if changed
0268	A417	I=0417	
026A	73ED	V3=V3+ED	:Subtract 3 to make room for
026C	7402	V4=V4+2	:sign, place sign at proper
026E	D341	SHOW 1MI @ V3, V4	:height; display it for show
			:or erase
0270	A4B3	I=04B3	:Set pointer for velocity digits
0272	3E00	SKIP; VE=0	:Check sign of new value of velo-
0274	127C	GOTO 027C	:city for jumping complement
0276	8620	V6=V2	:Use V6 as intermediate variable
0278	6200	V2=0	:to calculate the complement for
027A	8265	V2=V2-V6	:negative velocity
027C	F233	MI=V2 (3DD)	:Put velocity digits into memory
027E	23AA	DO 03AA	:Display new velocity data
0280	8820	V8=V2	:Pass velocity value in V2 on
			:to general variable V8
			DISPLAY HEIGHT
0282	23B8	DO 03B8	:Display height (to erase)
0284	2340	DO 0340	:Calculate new height
0286	A4B8	I=04B8	:Set pointer to memory storage
0288	23E4	DO 03E4	:area, write back new height,
			:convert hex to decimal
028A	A4C3	I=04C3	:Put LSD_{256} decimal digits in
028C	F233	MI=V2 (3DD)	:memory for display
028E	A4C1	I=04C1	:Same for MSD_{256}
0290	F033	MI=V0 (3DD)	
0292	23B8	DO 03B8	:Display new height
			DISPLAY LUNAR LANDER
0294	4100	SKIP; V1 .NE. 0	:Select lander picture according
0296	12A6	GO 02A6	:to height loss or more than 256
0298	A42F	I=042F	:meters; i.e., V1=0 is checked
029A	DAB1	SHOW 1MI @ VA, VB	:Show lander "far" for V1 .GT. 0
			:(to erase)
029C	6B0F	VB=F	:Take height from top of screen
029E	8B15	VB=VB-V1	:First, invert value, then mul-
02A0	22D6	DO 02D6	:tiply by 2. Maps range of V1
			:onto screen

ADDRESS	CODE	COMMENTS	REMARKS
02A2	DAB1	SHOW 1MI @ VA, VB	:Display lander "far"
02A4	122A	GOTO 022A	:Go begin main loop
02A6	67CF	V7=CF	:For "near" lander, move ref-
02A8	8725	V7=V7-V2	:erence position from top of
02AA	3F01	SKIP; VF=1	:lander to bottom, for proper
02AC	122A	GOTO 022A	:touchdown. No display if
			:shift is incomplete
02AE	A430	I=0430	:Select "near" lander picture
02B0	DAB6	SHOW 6MI @ VA, VB	:Show lander (to erase)
02B2	6BFF	VB=FF	:Take height from top down
02B4	8B25	VB=VB-2	:First invert, then
02B6	02E0	DO MLS at 02E0	:Divide by 8. Maps range of V2
02B8	7BFB	VB=VB+FB	:Shift reference position on
			:lander by subtracting 5 from
			:VB
02BA	DAB6	SHOW 6MI @ VA,VB	:Display lander "near"
02BC	122A	GOTO 022A	:End of normal cycle
			HEIGHT SCALE AND MOON SURFACE
02BE	A479	I=0479	:Set pointer to beginning of
			:graphics
02C0	F465	V0:V4=MI	:Reads 2 pairs of coordinates
			:and a jump
02C2	D011	SHOW 1MI @ V0,V1	:Display markings
02C4	7104	V7=V7+4	:Increment vertically
02C6	311F	SKIP; V1=1F	:Stop at reaching bottom of
02C8	12C2	GOTO 02C2	:screen; else draw on
02CA	F41E	I=I+V4	:Jump to moon surface picture
02CC	D231	SHOW 1MI @ V2, V3	:Display part of moon surface
02CE	7208	V2=V2+8	:Increment horizontally
02D0	3220	SKIP; V2=20	:Stop after 4 segments (last
02D2	12CA	GO 02CA	:one is 00) if not done with
			:moon surface
02D4	00EE	RETURN	
			MULTIPLY BY 2, WITH ROUNDING
02D6	8BB4	VB=VB+VB	:Or 2 x VB
02D8	6780	V7=80	:Dummy; to detect half values
02DA	8725	V7=V7-V2	:of MSD_{256} of height
02DC	8BF4	VB=VB+VF	
02DE	00EE	RETURN	
			MLS: DIVIDE BY 8
02E0	F8 FB	LDI FB	Register 06 holds the value
02E2	A6	PLO reg 6	assigned to variable X; i.e.,
02E3	06	LDN reg 6	VB. The contents of 06FB are
02E4	F6 F6		loaded via register 6, shifted
02E6	F6	SHR (3x)	right three times and stored
02E7	56	STR reg 6	back
02E8	D4 x	SEP reg 4	Return

ADDRESS	CODE	COMMENTS	REMARKS
			SWITCH FROM "NEAR" TO "FAR"
02EA	8104	V1=V1+V0	
02EC	A430	I=0430	:Select "near" lander picture
02EE	DAB6	SHOW 6MI @ VA, VB	:and display (to erase it)
02F0	7A01	VA=VA+01	:Adjust horizontal position
02F2	6B1F	VB=1F	:Adjust vertical position
02F4	A42F	I=042F	:Select "far" lander position
02F6	DAB1	SHOW 1MI @ VA, VB	:and display it
			CONTROL DATA
02FA	05 0C	V9=5	:Moon gravity key equivalent
02FC	00 15	VA, VB	:Initial horizontal and ver-
			:tical lander coordinates
02FE	01 01	VC=15	:Delay time (1/60 second) for
			:1 second cycle
		VD, VE=1	:Sets initial sign of velocity
			:to 1 (downward towards moon
			:surface)
			CALCULATE FUEL
0300	3100	SKIP; V1=0	:Check MSD_{256} of fuel. If .GT.
0302	1318	GO 0318	:0, a carry is accounted for
0304	3200	SKIP; V2=0	:Check LSD_{256} for remaining
0306	130C	GOTO 030C	:fuel, branch if some left
0308	6800	V8=0	:If not, set general variable
030A	00EE	RETURN	:to zero and return
030C	8285	V2=V2-V8	:Subtract entered fuel rate x
030E	3F00	SKIP; VF=0	:1 second, and check for fuel
0310	00EE	RETURN	:If okay or just used up
0312	8284	V2=V2+V8	:If more fuel requested than
0314	8820	V8=V2	:left, restore previous value
			:and set general variable (V8)
			:equal to amount left; then
0316	130C	030C	:Repeat above procedure
0318	8285	V2=V2-V8	:Subtract fuel rate x 1 second
031A	3F01	SKIP; VF=1	:from V2, and subtract carry
031C	71FF	V1=V1+FF	:from V1 if applicable
031E	00EE	RETURN	
			CALCULATE VELOCITY
0320	3E01	SKIP; VE=1	:Detect negative velocity
0322	1330	GOTO 0330	:and branch if .LT. 0
0324	8294	V2=V2+V9	:First, add moon gravity ac-
			:celeration x 1 second; then
0326	3F00	SKIP; VF=0	:check if 255 m height exceeded.
0328	13D4	GOTO 03D4	:If so, go to "OVERFLOW" - halt.
032A	8285	V2=V2-V8	:If not, subtract rocket engine
032C	8EF0	VE=VF	:acceleration and store new sign
032E	00EE	RETURN	:in VE and return
0330	8294	V2=V2+V9	:Works similarly to routine at

ADDRESS	CODE	COMMENTS	REMARKS
0332	8EF0	VE=VF	:0320, but with negative
0334	8285	V2=V2-V8	:velocity
0336	3E00	SKIP; VE=0	
0338	132C	GOTO 032C	:Now treat it as positive
033A	4F00	SKIP; VF .NE. 0	:Check for velocity .LT.
033C	13D4	GOTO 03D4	:-255 m/s. If so, go to
033E	00EE	RETURN	:"OVERFLOW". If not, return
			SWITCH FROM "FAR" TO "NEAR"
0350	A42F	I=042F	:Select "far" lander picture
0352	DAB1	SHOW 1MI @ VA, VB	:Display it (to erase)
0354	7AFF	VA=VA+FF	:Subtract 1 to adjust horiz.
0356	6B00	VB=0	:Adjust vertically
0358	A430	I=430	:Select "near" lander picture
035A	DAB6	SHOW 6MI @ VA, VB	:Display it
035C	00EE	RETURN	
			DETECT TOUCHDOWN OR CRASH
035E	8285	V2=V2-V8	
0360	80F0	V0=VF	
0362	80E5	V0=V0-VE	
0364	4000	SKIP; V0 .NE. 0	:Check both height and velocity
0366	00EE	RETURN	:If no touchdown or crash
0368	4001	SKIP; V0 .NE. 1	
036A	12EA	GOTO 02EA	:If height exceeds 255 m
036C	A4A8	I=04A8	:If height is .LE. 0, read
036E	F665	V0:V6=MI	:current value of velocity :from memory
0370	610A	V1=0A	:Set velocity margin for touch- :down (10 m/s)
0372	8215	V2=V2-V1	:Check velocity against margin
0374	3F00	SKIP; VF=0	:If VF=0, it's a safe touchdown
0376	137E	GOTO 037E	:If not, go to "crash" routine
0378	6A19	VA=19	:Select horizontal touchdown :position
037A	A460	I=0460	:Set pointer to text in memory
037C	138E	GOTO 038E	:Go display final text and exit
037E	A430	I=0430	:Select "near" lander picture
0380	DAB6	SHOW 6MI @ VA, VB	:Display it (to erase)
0382	A438	I=0438	:Select "crashed" lander
0384	6A0C	VA=0C	:Position "crashed" lander
0386	6B1B	VB=1B	
0388	DAB5	SHOW 5MI @ VA, VB	:and display the first part
038A	F51E	I=I+V5	:Advance memory pointer
038C	7A08	VA=VA+8	:Advance horizontal coordinate
038E	DAB5	SHOW 5MI @ VA, VB	:Display next part of crash scene
0390	7601	V6=V6+1	:Increment frame counter (V6)
0392	3605	SKIP; V6=5	:Exit with total of 5 frames
0394	138A	GOTO 038A	

ADDRESS	CODE	COMMENTS	REMARKS
			END OF GAME
0396	EFA1	SKIP; KEY .NE. VF	:Wait for restart by key 0
0398	1200	GOTO 0200	:If no screen overwrite occurred
039A	1396	GOTO 0386	:Else restart with key 1
			FUEL DISPLAY
039C	A498	I=0498	:Read current value of variables
039E	F665	V0:V6=MI	
03A0	A4A3	I=04A3	:Position for digits display
03A2	23C6	DO 03C6	:Digit display routine; 1 digit
03A4	3603	SKIP; V6=3	:Count digits to be written
03A6	13A0	GOTO 03A0	:To repeat for next digit
03A8	00EE	RETURN	
			VELOCITY DISPLAY
03AA	A4A8	I=04A8	:Read current value of variables
03AC	F665	V0:V6=MI	
03AE	A4B3	I=04B3	:Works like FUEL DISPLAY sub-
03B0	23C6	DO 03C6	:routine
03B2	3603	SKIP; V6=3	:Counts three digits
03B4	13AE	GOTO 03AE	:To repeat
03B6	00EE	RETURN	
			HEIGHT DISPLAY
03B8	A4B8	I=04B8	:Read current values
03BA	F665	V0:V6=MI	
03BC	A4C2	I=04C2	:Works like above routines
03BE	23C6	DO 03C6	
03C0	3604	SKIP; V6=4	:Counts four digits
03C2	13BC	GOTO 03BC	
03C4	00EE	RETURN	
			DIGIT DISPLAY
03C6	F61E	I=I+V6	:Advance pointer in memory;
03C8	F065	V0:V0=MI	:V6 was read in before (as 0)
03CA	F029	I=V0 (LSDP)	:and is tested in calling routine
03CC	D345	SHOW 5MI @ V3, V4	:Display the (decimal) digit
03CE	8354	V3=V3+V5	:Advance display coordinate :(horizontally)
03D0	7601	V6=V6+1	:Increment V6 to count digits
03D2	00EE	RETURN	
			OVERFLOW
03D4	6602	V6=2	:Pre-set frame counter
03D6	6A20	VA=20	:Position for text on screen
03D8	6B81	VB=18	
03DA	A483	I=0483	:Select text in memory
03DC	1388	GOTO 0388	:Final text display section

ADDRESS	CODE	COMMENTS	REMARKS
			KEY ENTRY
03DE	A497	I=0497	:Location to store key entry
03E0	F065	V0:V0=MI	:Read previous entry into V0
03E2	1244	GOTO 0244	:If no key was pressed during
			:the cycle
			HEX TO DECIMAL CONVERSION
03E4	F255	MI=V0:V2	:Write the variables back
03E6	8114	V1=V1+V1	:Double the MSD_{256} to jump to
03E8	A4C8	I=04C8	:proper place in look-up table
03EA	F11E	I=I+V1	:Advance pointer by the doubled V1
03EC	F165	V0:V1=MI	:Read 2 numbers from table
03EE	6764	V7=64	:(i.e., 100_{10})
03F0	8275	V2=V2-V7	:Subract 100 from LSD_{256} until 0
03F2	7001	V0=V0+1	:Add the carry
03F4	3F00	SKIP; VF=0	
03F6	13F0	GOTO 03F0	
03F8	8214	V2=V2+V1	
03FA	8274	V2=V2+V7	
03FC	80F5	V0=V0-VF	:Adjust value from table with
03FE	00EE	RETURN	:carry and then return

DATA

0400	00 01 C0 28 00 05 00 00	:Initial fuel values
	00 00 00 04 04 08 00 00	
0410	00 00 93 28 08 05 00 E0	:Initial velocity values
	00 00 00 01 04 07 00 00	
0420	00 0F FF 23 10 05 00 00	:Initial height values
	00 00 04 00 09 05 00 80	
0430	10 3C FF 7E 24 42 00 00	
	03 3F 38 11 00 04 62 C9	
0440	B0 00 00 00 00 00 00 75	
	86 84 84 74 B0 09 39 28	
0450	3D E8 08 CE 29 E9 xx xx	
	xx xx xx xx xx xx xx xx	
0460	40 EE 4A 4A 6E 00 AE A8	:Graphics data
	A8 EE 80 E3 A2 A2 A3 80	
0470	BA AA AA BB 00 2E AA AA	
	EA 00 03 00 1F 01 C0 FF	
0480	F7 BF 00 00 EA AA AA E4	
	00 EA EC 88 E8 68 89 C9	
	⋮	:Dynamic memory for fuel, velo-
		:city and height

```
ADDRESS CODE                                    COMMENTS

 04C8   00 00 02 38 05 0C 07 64    :Hex to Decimal two-byte
 04D0   0A 18 0C 50 0F 24 11 5C    :conversion table
        14 30 17 04 19 3C 1C 10
 04E0   1E 48 21 1C 23 54 26 28
        28 60 2B 34 2E 08 30 40
 04F0   33 14 35 4C 38 20 3A 58
        3D 2C 40 00 42 38 45 0C
```

(Editor's note: We offer the program tape ($5.00) for those of you who don't wish to type it in. So far as we can tell, the code is accurately reproduced here - but no guarantees! If something doesn't work - and you're certain you entered the code exactly as printed - give us a call and we'll try to help you get it going. Call Rick after 1:00 EST at (215) 631-9052.)

IMPORTANT ANNOUNCEMENT!

ARESCO

is moving! As of April 1, 1979, our new address will be:

ARESCO
Box 1142
Columbia
MD 21044

If you write on or before March 31, use the Audubon address. If you write on or after 4/1, use the new address to reach us quickest.

DOUBLE-BUFFER SPEEDUP HARDWARE FOR 64 x 128 GRAPHICS WITH THE COSMAC 1802 AND THE 1861 VIDEO CHIP

by Ben Hutchinson

(This is Part 2, the conclusion of this article. The text and two pages of diagrams were published in issue #7.)

COUNTER CONTROL SIGNALS FOR 74163 ADDRESS COUNTERS

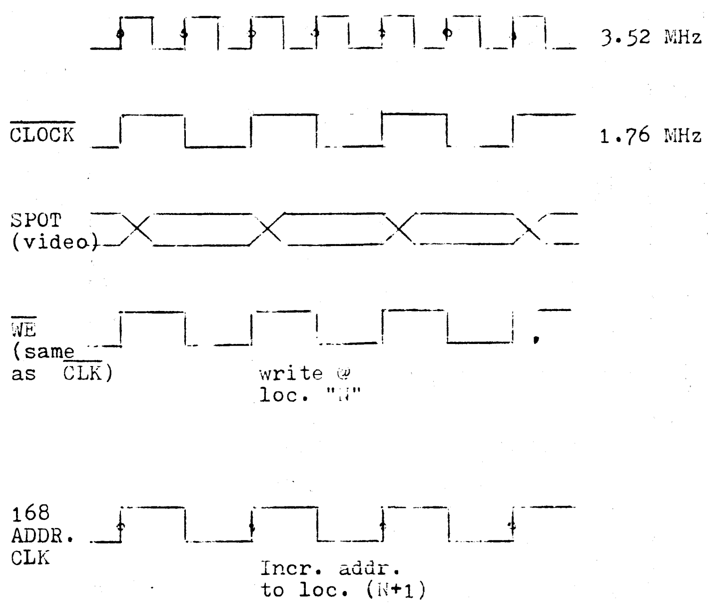
Symbol	Derivation	Function
$\overline{\text{LOAD}}$	$\overline{\text{ELS+63+INT}}$	Three functions: 1) Just before frame start, when INT=0, loads both counters to stop state (128), so both start at 00 on first occurrence of S2·TPB (start of first video line). 2) When and only when counter is in write mode (Q=1 for "left", $\overline{\text{Q}}$=1 for "right"), loads counter with stop state (192) following state 63. This ends first write line. 3) Only in write mode, loads counter with state 64 on first occurrence of S2·TPB when in state 192 (stopped). This starts 2nd write line.
$\overline{\text{OLS}}$ (odd line start) (Pins 1, both pkg.)	$\overline{\text{128}\cdot\overline{\text{64}}\cdot\text{S2}\cdot\text{TPB}}$	Sets counter to 0 on first occurrence of S2·TPB when in state 128. (Pin 1 is synchronous clear.) This starts first write line and all read lines.
ENABLE	$\overline{\text{128}}$	Stops counter whenever the MSBit with value 128, is 1. This normally means 128 or 192. Any state .GE. 128, entered at random or at power-up, will stop count to wait for ELS or OLS.
ELS	$\text{128}\cdot\overline{\text{64}}\cdot\text{S2}\cdot\text{TPB}$	See 3) under $\overline{\text{LOAD}}$ above.

LINE NUMBER	Q	$\overline{Q}$	"LEFT" MEMORY AND ADDRESS COUNTER FCN	$\overline{CS1}$	$\overline{WE}$	COUNTER (163) CLOCK	COUNTER STATES	"RIGHT" MEMORY AND ADDRESS COUNTER FCN	CS1	$\overline{WE}$	COUNTER (163) CLOCK	COUNTER STATES
1	1	0	Store 1861 output (write)	(1.7 Mhz Clk)		1.76 MHz	0-63	Supply output video (read)	0	1	3.52 MHz	0-127
2	1	0		(1.7 MHz Clk)		1.76 MHz	64-127		0	1	3.52 MHz	0-127
3	0	1	Supply output video (read)	0	1	3.52 MHz	0-127	Store 1861 output (write)	(1.7 MHz Clk)		1.76 MHz	0-63
4	0	1		0	1	3.52 MHz	0-127		(1.7 MHz Clk)		1.76 MHz	0-63

COUNTER STOP STATE	WHEN ENTERED	FOLLOWS STATE	FOLLOWED BY STATE	HOW RESTART?
192	End of odd lines In write mode only	63	64	S2•TPB (state 142)=ELS=1 makes $\overline{LOAD}$ =0 (pin 9)
128	End of even lines In write mode;	127	0	S2•TPB ($\overline{64}$•128)=OLS=1 $\overline{OLS}$ makes $\overline{CLR}$=0 (pin 1)

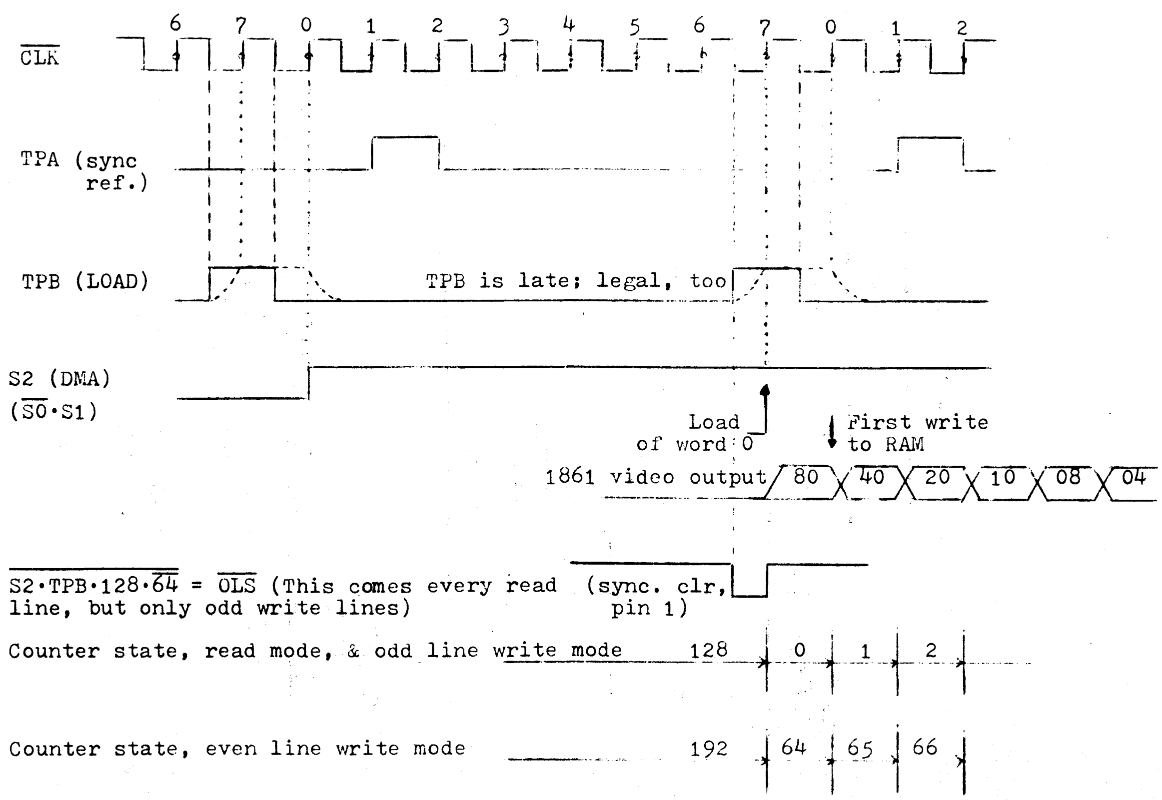
CLOCK PHASING (Taking the 1861 Data Sheet literally; i.e., shift video out on lo-to-hi trans. of $\overline{CLK}$)

From this, it looks like clearly the 93421 $\overline{WE}$ and the 74163 clock have the <u>same</u> polarity.



Best interpretation of the data <u>sheet</u> is that this is also the same as the 1861 pin (i.e., $\overline{CLR}$). If not, one inverter fixes it.

START-OF-LINE TIMING



COMMENTS

In case you hadn't noticed, folks, this is issue #8. That means there are only two more issues in this year's volume, and we have to start looking at whether we'll continue publishing the VIPER for another year. We don't plan to get rich off the VIPER, but we don't plan to lose any more than we have to, either!

As of now, we're accepting subscription orders for Volume 2. We won't cash any checks, or charge any credit cards, until we're certain we're going to publish - if we don't, you'll get your own check back, uncashed. We'll know whether we'll do this again next year by the middle of July (there's no VIPER in July, remember!), and if it looks like we have a sufficient response to indicate that we're performing a useful function, we'll be here again in August. (That means: Order your Volume 2 subscription now, so we can count noses!)

Use the order form below - but be sure to write VOLUME 2 in very large letters so our order clerk (me) won't get confused. Send renewals to our new address: PO Box 1142, Columbia, MD, 21044, along with your check or credit card number.

And keep on sending in articles - we'll need something to print in the first issue of Volume 2 - and your experiences are more interesting than you know! - Terry

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

Place
Stamp
Here

VIPER

P.O. Box 43

Audubon, PA 19407

VIPER

P.O. Box 43

Audubon, PA 19407

TO: